IMPLEMENTATION OF A HEURISTIC METHOD OF DECOMPOSITION OF PARTIAL BOOLEAN FUNCTIONS

Arkadij Zakrevskij, Nikolai Toropov

Abstract: An original heuristic algorithm of sequential two-block decomposition of partial Boolean functions is researched. The key combinatorial task is considered: finding of suitable partition on the set of arguments, i. e. such one, on which the function is separable. The search for suitable partition is essentially accelerated by preliminary detection of its traces. Within the framework of the experimental system the efficiency of the algorithm is evaluated, the boundaries of its practical application are determined.

Keywords: Partial Boolean Functions, Heuristic Method of Decomposition

ACM Classification Keywords: B.6. Combinational logic, switching theory, automatic synthesis, optimization.

Conference: The paper is selected from XIVth International Conference "Knowledge-Dialogue-Solution" KDS 2008, Varna, Bulgaria, June-July 2008

Introduction

It is generally accepted to understand decomposition of a Boolean function as presenting it by a composition of functions of smaller number of variables. This task is diversiform - enough to note, that any nontrivial logic circuit, implementing some Boolean function of many variables, can be considered as a composition of functions realizable by separate units. We shall consider a private but important case of the task becoming classical, as not one hundred publications is devoted to it [1], namely *sequential two-block decomposition*.

In this case the problem is stated as follows. On the set of arguments $x = (x_1, x_2, \ldots, x_n)$ a Boolean function $f(x)$ is preset. It is necessary to replace $f(x)$ by an equivalent composition $g(h(u, w), w, v)$ of Boolean functions $g(h, w, v)$ and $h(u, w)$ of smaller number of variables. By that the conditions $x = u \cup w \cup v$ and $u \cap w = u \cap v = w \cap v = \varnothing$ should be fulfilled, generating a *weak partition* u/v on the set of arguments x. This replacement is named as decomposition of the function $f(x)$ at the partition u/v ($f(x)$ is *decomposable* at u/v) and can simplify a logic circuit implementing function $f(x)$ (for example, at logical synthesis in the basis of units LUT (look up tables)). It is meaningful under the condition ($|u| > 1$) & ($|v| > 0$), otherwise decomposition is trivial – the circuit does not become simpler. If $|w| = \varnothing$, the composition is named *disjunct*, otherwise – *non-disjunct*. The task becomes essentially complicated, if the Boolean function $f(x)$ appears to be *partial*, being defined not on all elements of the Boolean space $M = \{0, 1\}^n$. The decomposition of a Boolean function is a difficult combinatorial task, which complexity fast grows with increase of the number of variables n. An original heuristic decomposition technique is considered below, which efficiency is provided practically with acceptable compromise between speed of finding of a solution and its reliability. The program implementation of that method is based on usage of the set of special macro operations above long (2^n-component) Boolean vectors $\boldsymbol{f}$, with which it is possible to represent arbitrary Boolean functions $f(x)$ of n variables.

In paper [2] was proved

The assertion 1. At equiprobable sampling of the function $f(x)$ from the set of all Boolean functions of n variables the probability of decomposability of this function is bounded above by the value

$$C_n^2 (n - 2) \gamma 2^{n-3}, \text{ where } \gamma = 88/256.$$

This value fast tends to zero with growth of n (for example, at n = 6, 12, 18 it receives accordingly values 0,012, 10^{-234}, 10^{-15193}), whence follows, that decomposability of arbitrary Boolean functions of many variables is rather improbable. Therefore, a practical sense the task of decomposition has only for such function, which, as a priory

is known, is presented by some composition, which is required to be found. The task in this setting is considered below.

Preparation of input data

An approach is widely spread, according to which concrete examples of input data for programs of solution of the diversiform tasks of logical design are selected from special libraries shaped in view of some practical reasons. In the given paper another way is offered based on generation of random examples with set values of some parameters.

In the considered task of decomposition such parameters are: n – the number of arguments of the function $f(x) = f(x_1, x_2, \ldots, x_n)$, $p = |u|$ - (the power of set u), $q = |v|$ and r - the degree of uncertainty of the function $f(x)$ (more exactly this value will be defined below).

During preparation of an example there are selected by random from the set x p variables forming the set u, and q variables forming the set v. Remaining variables of the set x will form the set w. Then random Boolean functions $h(u, w)$ and $g(x, w, v)$ are generated. These operations are simple enough and are fulfilled with usage of generators of random evenly distributed Boolean vectors.

More composite appears the composition of functions $h(u, w)$ and $g(x, w, v)$, resulting in obtaining the function $f(x)$. At presentation of the offered below method of solving this task it is convenient to use the language of macro operations above long 2^n-component Boolean vectors f. Along with standard component-wise operations above vectors of identical dimension ($a \vee b$, $a \oplus b$, etc) it contains high-performance operations above adjacent elements of Boolean space, fulfilled in parallel in all 2^{n-1} couples of adjacent elements [3]. While we shall take advantage from following of such operations:

$f - k$ – assignment of value 0 to argument x_k (obtaining of the function $f(x_k = 0)$),

and also generalizing it operation $f - u$, at which execution the value 0 is assigned to all arguments, marked with 1 in n-component vector u.

We shall use also the operation $h \times u$ of mapping the function $h(u)$ onto the interval of space M, corresponding to the conjunction of inversions of the variables not marked in u. All elements of remaining intervals with the same outer variables gain by that the value 0.

In terms of these operations the program of calculation of vector f, representing the required function $f(x)$, looks so:

$$a := h \times (u, w) - v,$$
$$b := g_0 \times (w, v) - u,$$
$$c := g_1 \times (w, v) - u,$$
$$f := \bar{a}\, b \vee a\, c,$$

where the vectors g_0 and g_1 are represented with two halves of vector g, specifying coefficients of decomposition of the function $g(x, w, v)$ by variable x.

Let's illustrate this program on an example (fig. 1), where $n = 6$, and the sets $u = (x_1, x_2)$, $w = (x_3, x_4)$ and $v = (x_5, x_6)$ are preset by appropriate Boolean vectors $u = 110000$, $w = 001100$ and $v = 000011$.

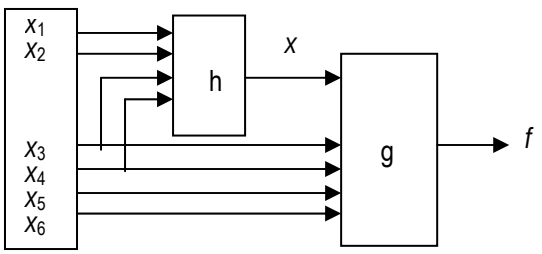

Fig.1

We admit, that the generated Boolean functions $h(u, w)$ and $g(x, w, v)$ are represented accordingly by 2^4-component vector h and 2^5-component vector g with the generally accepted order of following of components:

h = 11010010 01101100,

g = 00110100 11001001 10100101 10101011.

Then the calculations are reduced to obtaining the shown below sequence of Boolean vectors. At calculation of vectors $a^* = h \times (u, w)$, $b^* = g_0 \times (w, v)$ and $c^* = g_1 \times (w, v)$ the appropriate interval of space M is found at first (it is marked by the bold font). Then the vector of the considered function (h, g_0 or g_1) is brought into it, with saving the order of components following.

```
                                 -------- -------- -------- -------- x1
                 -------- --------                  -------- --------   x2
          -------- -------- --------          -------- -------- -------- x3
     ----    ----    ----    ----    ----    ----    ----    ----    ----   x4
   --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  -- x5
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -   x6

10001000 00001000 00000000 10000000 00001000 10000000 10001000 00000000  a*
11111111 00001111 00000000 11110000 00001111 11110000 11111111 00000000  a
00110100 11001001 00000000 00000000 00000000 00000000 00000000 00000000  b*
00110100 11001001 00110100 11001001 00110100 11001001 00110100 11001001  b
10100101 10101011 00000000 00000000 00000000 00000000 00000000 00000000  c*
10100101 10101011 10100101 10101011 10100101 10101011 10100101 10101011  c
00000000 11000000 00110100 00001001 00110000 00001001 00000000 11001001  ab
10100101 00001011 00000000 10100000 00000101 10100000 10100101 00000000  ac
10100101 11001011 00110100 10101001 00110101 10101001 10100101 11001001  f
```

After obtaining the function $f(x)$ an uncertainty is brought into it by replacement of some of its values (0 or 1) with the symbol of uncertainty "–". The result of this conversion is represented by one ternary vector f^- or the couple of Boolean vectors f^0 and f^1, in which unities mark values 0 and 1 of function $f(x)$. For example (n = 4),

f^- = 011–0001 –101–110,

f^0 = 10001110 00100001,

f^1 = 01100001 01010110.

The degree of uncertainty of the function is set by the parameter r, receiving values from set {0, 1, 2, …, 31} and defining probability r/32 that the arbitrary selected component of vector f^- will receive value "–".

Depositing of uncertainty in the function $f(x)$ is carried out on the basis of results obtained in [4], which essence can be caught from the following example. In order to receive a random Boolean vector with probability 19/32 of appearance of unity in an arbitrary selected component, it is enough to present number 19 by its binary code 10011, then to put in correspondence to values 0 and 1 Boolean operators $\wedge$ and $\vee$, and, sorting out components of the code from the right to the left, to fulfill the following sequence of operations above completely random (with probability 0,5 of appearances of 1 in any component) independent Boolean vectors c_1, c_2, c_3, c_4, c_5 :

$$((0 \vee c_1 \vee c_2) \wedge c_3 \wedge c_4) \vee c_5$$

where 0 is a vector, which all components are equal to zero.

Method of search for a suitable partition by traces

In the basis of Boolean functions decomposition techniques the solution of the following tasks lays.

The task 1. For a given function $f(x)$ and partition u/v to clarify, whether $f(x)$ is decomposable at u/v, i. e. whether there exists a composition $g(h(u, w), w, v)$, equivalent to function $f(x)$, and, maybe, to find functions g and h.

If such a composition exists, we shall term partition u/v *suitable*.

The task 2. For a given function $f(x)$ to find a suitable partition.

The second task is more difficult. It is obvious, that the exhaustive search of all partitions with the purpose of checking them on fitness practically is unrealizable at major n, as their number is approximated by the value 3^n. Much more efficient is the method of search of suitable partition by traces, offered in [2]. It is based on the following definitions and assertions.

Consider two partitions u/v and u^*/v^*, bound with relations $u^* \subseteq u$ and $v^* \subseteq v$. Let's speak, that the partition u^*/v^* *submits* to partition u/v, or is its *trace*.

The assertion 2. If the Boolean function $f(x)$ is decomposable at partition u/v, it is decomposable also at its trace u^*/v^*.

Corollary. If the function $f(x)$ is not decomposable at partition u^*/v^*, it is not decomposable also at u/v.

Suppose, that $|u| = k$ and $|v| = m$. A partition with $k = 2$ and $m = 1$ we shall term as a *triad*. It is the simplest of partitions, on which the nontrivial decomposition can take place.

The assertion 3. The Boolean function $f(x)$ is not decomposable, if it is not decomposable at any of triads.

Let's start therefore the search for suitable partition u/v from the search for its traces on the set of triads, i.e. from finding of a suitable triad.

Assume, that the search consists in random sampling from a series of q triads completing by finding a suitable one. It is shown in [2], that practically (at $n > 10$) any suitable triad submits to the required partition u/v, in other words, accessory solutions miss. At this supposition it is fair

The assertion 4. The expectation $M(q)$ of the value q is equal to $C_n^2(n-2) / C_k^2\, m$.

Thus, the search of a suitable triad is performed rather fast. For example, the formula representing the value $M(q)$ can be approximated by more simple formula $3^3 = 27$ (at $k = m = n/3$) and $2^3 = 8$ (at $k = m = n/2$).

Having found a suitable triad, it is possible to extend sequentially sets u and v, testing different variables on possibility of their inclusion into one of these sets (the inclusion is possible, if the extended partition remains suitable).

Programming in macro operations

Fragments of partition u/v. Any weak partition u/v divides a 2^n-component ternary vector f^- into 2^{n-p-q} parts representing coefficients of disjunctive decomposition of the function $f(x)$ by variables of set w. Each of these parts can be presented by an appropriate *fragment* – so we shall term a ternary matrix the size $2^p \times 2^q$, which rows correspond to different sets of values of variables from u, and columns – to sets of values of variables from v.

A partition u/v appears *suitable*, if each its fragment is suitable. And a fragment is *suitable*, if the partial Boolean function $f(x)$ can be predetermined in such a way, that the fragment will contain no more than two values of Boolean rows. In other words, the graph of orthogonality of rows of each fragment should be bichromatic [5]. The

check of this condition becomes simpler for a special case of partition – a triad (a, b)/c. In this case the size of the graph equals 4×2.

Checking a triad on fitness. The graph of orthogonality of rows of a triad fragment contains four nodes, therefore, it is bichromatic, if there will be no cycle of length three in it. Consider two rows of a triad fragment corresponding to values (a, b) = (0, 0) and (a, b) = (1, 1).

If such a cycle exists, then at least one of selected nodes will belong to it. Therefore, it is enough to test each of these two nodes on belonging to some cycle of length three. If such belonging will not be revealed, the graph is bichromatic and the triad is suitable. Necessary and sufficient condition of belonging of a node, i.e. the row corresponding to it, to a cycle of length three could be formulated so: among rows orthogonal to the given one, there are mutually orthogonal rows.

The offered way is implemented by the following algorithm, which is remarkable by that it checks on fitness simultaneously all 2^{n-3} fragments corresponding to a given triad, and by that checks fitness of the triad as a whole. The considered partial Boolean function $f(x)$ is set by a couple of Boolean vectors f^0 and f^1.

In this algorithm alongside with introduced earlier operation $f - k$ the operation $f + k$ is used, defined similarly: it means assignment of value 1 to argument x_k. Both operations are easily implemented in Boolean space on couples of elements, adjacent by a variable x_i. The algorithm contains also the operation

$$S_k^* f = (f - k)^*(f + k)$$

of symmetrizations of a vector-function f by variable x_k and Boolean operation $^* \in \{\vee, \wedge, \oplus\}$. In further presentation the generalized operation $S_t^* f$ is used, the equivalent to operation $S_k^* f$, which is fulfilled for all variables of a subset $t = (t_1, t_2, \ldots, t_m) \subseteq x$.

Let's illustrate introduced operations by the following examples, in which $u = (x_2, x_5)$:

$$\begin{aligned}
f &= 10010010\ 01111000\ 01100001\ 11100011,\\
f-2 &= 10010010\ 10010010\ 01100001\ 01100001,\\
f+2 &= 01111000\ 01111000\ 11100011\ 11100011,\\
f-u &= 11000011\ 11000011\ 11110000\ 11110000,\\
S_2^{\vee} f &= 11111010\ 11111010\ 11100011\ 11100011,\\
S_2^{\oplus} f &= 11101010\ 11101010\ 10000010\ 10000010,\\
S_u^{\oplus} f &= 00111111\ 00111111\ 11000011\ 11000011.
\end{aligned}$$

The check of a triad is performed at first on initial rows of fragments constituting together the initial coefficient f^- of decomposition of function $f(x)$ by variables a and b. The rows, orthogonal to the initial row, are marked with value 1 in a computed vector g and are checked further on compatibility (non-orthogonality). With this purpose the couple of vectors h^0 and h^1 is evaluated. The same as initial vectors f^0 and f^1, they are Boolean vectors with 2^n components.

$h^0 := (f^0 - a) - b$	Obtaining of initial coefficient f^-
$h^1 := (f^1 - a) - b$	
$g := S_c^{\vee}(h^0 f^1 \vee h^1 f^0)$	Selection of rows, orthogonal to the initial one
$h^0 := S_a^{\vee}(S_b^{\vee}(f^0 g)$	Checking them on compatibility
$h^1 := S_a^{\vee}(S_b^{\vee}(f^1 g)$	

If it appears, that $h^0 h^1 \neq 0$, the triad admits unsuitable. Otherwise the last rows of the fragment are checked, which constitute the final coefficient f^+ (by that the symbol "–" in first two strings of the algorithm is changed for "+"). If it appears, that the triad all the same is unsuitable, other triads are tested, until a suitable one will not be found.

Search of the partition by a trace. If the considered triad (a, b)/c has appeared suitable, it is possible to assume, that it is a trace of the required partition. In this case the latter can be found, moving by the track, i. e. using the value of vector g obtained at the previous stage, and sequentially extending the sets u and v with initial values $u = (a, b)$ and $v = (c)$.

Let's begin from the set v. Sorting out sequentially all elements s from the set $x \setminus (u \cup v)$, we shall discover among them such ones, at which inclusion in set v the partition u/v remains suitable. With this purpose three operations are fulfilled for each element s :

$$e := S_s^{\vee} g$$
$$h^0 := S_u^{\vee} (f^0 e)$$
$$h^1 := S_u^{\vee} (f^1 e)$$

And if $h^0 h^1 = 0$, the element s is included into v, which is implemented by the operations

$$v := v \cup \{s\}, \quad g := e.$$

Then the maximum extension of the set u is looked for. If it is known, that the required partition is disjoint, it is possible to put $u = x \setminus v$. Otherwise, it is necessary to test all variables from the current value of the set $x \setminus (u \cup v)$ and, if it is possible, to include them in set u.

Check of the current element s we shall perform by an heuristic algorithm which operations are limited. It considers the initial coefficient f^- of decomposition of the function f by the current value of set u, discloses orthogonal to it coefficients, checks them for compatibility and, in case of compatibility, includes element s into set u without further checking.

$$e := u \cup \{s\}$$
$$h^0 := f^0 - e$$
$$h^1 := f^1 - e$$
$$g := S_v^{\vee} (h^0 f^1 \vee h^1 f^0)$$
$$h^0 := S_u^{\vee} (f^0 g)$$
$$h^1 := S_u^{\vee} (f^1 g)$$

If $h^0 h^1 = 0$, element s is included into u, that is implemented by the operation $u := e$.

So the set u and, therefore, the required partition u/v as a whole are found.

Experimental system

For the estimation of efficiency of the designed heuristic program of decomposition of Boolean functions and delimitation of its practical application an experimental system of generation of random examples and their solutions were designed.

The generation of examples is controlled by the following data-ins:

n – the number of arguments of the generated function $f(x)$,

p and q – the numbers of variables in sets u and v of partition u/v on the set of variables x,

r – the degree of uncertainty of the function $f(x)$,

g – the number of an example, i. e. the serial number of the unit of a quasi-random sequence used at generation of the current example.

The type of partition is determined also: D (disjunctive) or ND (non-disjunctive).

The results of solution of the regarded examples are fixed by values of the following output parameters:

Nt – the number of surveyed triads when searching suitable ones,

Tc, Tt, Tv, Tu and Tw – time (in seconds), expended accordingly on preparation of an example, on finding of a suitable triad, on the extension of set *v*, on the extension of set *u* and on solution of an example as a whole (excluding Tc).

Besides the quality of the solution is evaluated and it is decided, whether the calculated partition *u*/*v* and functions *h*(*u*, *w*) and *g*(*x*, *w*, *v*) coincide with given ones.

For simplification of the experimental research of the program a special mode is introduced into the system, at which a series of examples is generated and solved with coincident values of all parameters except one, for which a difference between adjacent values and their number are defined.

Experimental estimations of the program efficiency and the boundary of its practical application

A series of carried out experiments displays, that the circumscribed heuristic program operates safely enough, discovering exact solutions, if 8 < *n* < 28 and *r* < 30. The errors can arise outside this range and, maybe, on the boundaries. The results of solution of some concrete examples and conclusions following from them are shown below.

Estimation of the upper bound by the number of variables n. Let's cite the results of an experiment, in which the number of variables *n* varies from 22 up to 28, and the values of remaining parameters are fixed.

Type	n	p	q	r	g	Nt	Tc	Tt	Tv	Tu	Tw	Result
D	22	11	11	20	1	1	0.34	0.16	1.56	0.00	2.08	OK
D	23	12	11	20	1	1	0.69	0.33	3.33	0.00	4.45	OK
D	24	12	12	20	1	5	1.47	1.90	7.20	0.00	10.79	OK
D	25	13	12	20	1	2	3.12	1.99	15.19	0.00	20.70	OK
D	26	13	13	20	1	2	6.18	4.04	34.07	0.00	45.20	OK
D	27	14	13	20	1	2	12.52	8.43	74.24	0.00	99.01	OK
D	28	14	14	20	1	2	24.97	31.22	1938.61	0.00	2096.32	OK

The results of the experiment are positive (OK – a retrieved partition coincides with the initial one) and display, that at given parameters the program discovers solution rather fast (in limits of a minute), except for the case *n* = = 28, when the length of vector ***f*** becomes too big for the used RAM, that results in sharp increase of time Tv. They show also, that at *n* < 28 the time of solving grows twice at each increase of the number of variables *n* by unity. Let's remark, that Tu = 0, as an example of the task with disjoint partition is considered.

Estimation of the time of searching for a suitable triad. The results of the previous experiment display that at major *p* and *q* this time (Tt) is relatively small. The position varies at small values of parameters *p* and *q*. In that case, the number Nt of triads which are looked through in searches of a suitable one strongly increases and, as the corollary, grows the time of solution of the task as a whole, which is almost completely spent for this search.

Type	n	p	q	r	g	Nt	Tc	Tt	Tv	Tu	Tw	Result
ND	14	2	1	0	1	491	0.00	0.04	0.00	0.00	0.04	OK
ND	15	2	1	0	1	603	0.00	0.09	0.00	0.00	0.09	OK
ND	16	2	1	0	1	1250	0.01	0.36	0.01	0.00	0.37	OK
ND	17	2	1	0	1	1618	0.00	0.93	0.00	0.02	0.95	OK
ND	18	2	1	0	1	2055	0.02	2.37	0.01	0.05	2.43	OK
ND	19	2	1	0	1	2978	0.04	9.82	0.05	0.11	9.98	OK
……												
ND	27	2	2	0	1	2649	10.28	6812.28	48.64	95.09	6956.09	OK

Let's remark, that the value Nt is characterized by a major dispersion. For example, for a series of 12 random examples at fixed values of parameters (n, p, q, r) = (16, 2, 1, 0) the following results are obtained:

Nt =	1250	596	2799	498	2315	1598	4834	3797	2362	1637	3990	700
Tt =	0.36	0.18	0.80	0.14	0.66	0.46	1.39	1.08	0.67	0.47	1.15	0.21

With the purpose of specification of these data an experiment above ten random examples with parameters (Type, n, p, q) = (D, 24, 12, 12) was carried out. At r = 28 all ten examples were solved correctly, at r = 29 exact solutions were obtained only for seven examples, and at r = 30 all solutions have appeared false.

Estimation of the upper bound by the degree of uncertainty r. With growth of the degree of uncertainty of a Boolean function the probability of obtaining erratic solution at its decomposition grows also. The experiment over a series of random Boolean functions obtained as a result of composition at a strong partition with equal (whenever possible) values of parameters p and q was carried out. The maximum degree of uncertainty r_{max} of the function of n variables was defined, at which there was an exact solution. The results are shown in the following table:

n =	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
r_{max} =	0	1	6	8	14	19	19	21	22	23	25	25	26	26	28	27	29	29	29	29	30	30

Estimation of the lower bound by the number of variables n. At small number of variables many from suitable triads are not true traces of the partition, as it follows from the following experimental data received at a strong partition with equal (whenever possible) values of parameters p and q and following values of other parameters: n = 5, 6, … 16 and r = 15.

Number of variables	5	6	7	8	9	10	11	12	13	14	15	16
Total number of triads	30	60	105	168	252	360	495	660	858	1092	1365	1680
Number of suitable triads	28	44	80	78	54	90	77	90	126	147	196	224
Number of partition traces	6	9	18	24	40	50	75	90	126	147	196	224

The following table displays, how frequently it leads to a false solution. In conducted experiments series of 20 random examples with fixed values of parameters Type, n, p, q, r were solved and for each series the number of false solutions was counted up.

Type	n	p	q	r	Falsh	Type	n	p	q	r	Falsh
D	4	2	2	0	62	ND	4	2	2	0	62
D	5	3	2	0	76	ND	5	2	2	0	82
D	6	3	3	0	64	ND	6	2	2	0	55
D	7	4	3	0	50	ND	7	3	2	0	21
D	8	4	4	0	35	ND	8	3	3	0	6
D	9	5	4	0	6	ND	9	3	3	0	0
D	10	5	4	0	0	ND	10	4	3	0	0

About finding functions h(u, w) and g(x, w, v). The main objective of decomposition of a Boolean function is finding a suitable partition u/v, which permits to reduce the number of entry poles in blocks of the composition $g(h(u, w), w, v)$. Some interest represents finding functions $h(u, w)$ and $g(x, w, v)$. It is obvious, that, having found these functions, we uniquely determinate also the partition u/v. However, from finding the partition it does not follow, that we have found also these functions.

That is confirmed by the following experimental data, received by solution of the task of decomposition on hundred random examples with parameters (Type, n, p, q) = (D, 10, 5, 5) and r taking values from 0 up to 26 including. The number of guessed right partitions is designated as Nr, the number of guessed functions – as Nf.

r	0	1	…	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Nr	100	98	…	98	98	98	97	95	92	86	75	63	47	28	16	13	10	6	6	3	1	0
Nf	100	98	…	98	97	97	96	94	88	82	64	52	33	15	7	4	2	1	1	0	0	0

Conclusion

An original heuristic algorithm of sequential two-block decomposition of partial Boolean functions of n variables on weak partitions on the set of arguments is implemented by a program and experimentally researched. It is shown, that the implementing program operates safely enough, discovering exact solutions if $8 < n < 28$ and if the function is defined not less than on 29/32 parts of Boolean space. The results of more detailed research of the program on boundaries of this range and outside it are cited.

Acknowledgement

The work was partially supported by Belarus Republican Fund of Fundamental Research {Project Ф07МС-034).

Bibliography

1. Perkowski M. A., Grigiel. A survey of literature on function decomposition, Version IV. November 20, 1995. Portland State University.

2. Zakrevskij A.D. Sequential decomposition of a Boolean function – the search for an appropriate partition on the set of arguments. - Reports of NAS of Belarus, 2007, v. 51, No 1, pp. 7-11 (in Russian).

3. Zakrevskij A.D. Parallel operations over neighbors in Boolean space. - Proceedings of the Sixth International Conference CAD DD-07, - Minsk 2007, vol. 2, pp. 6-13.

4. Zakrevskij A.D. Implementation of random events with a given probability. - Transactions of SFTI, Tomsk, 1965. - Is. 47, pp. 56-59 (in Russian).

5. Arkadij Zakrevskij. Decomposition of Boolean functions – recognizing a good solution by traces. – International Journal “Information Theories and Applications”, vol. 14, 2007, pp. 359-365.

Author information

Arkadij Zakrevskij - United Institute of Informatics Problems of the NAS of Belarus, Surganov Str. 6, 220012 Minsk, Belarus; e-mail: zakr@newman.bas-net.by

Nikolai R. Toropov - United Institute of Informatics Problems of the NAS of Belarus, Surganov Str. 6, 220012 Minsk, Belarus; e-mail: toropov@newman.bas-net.by