ITHEA

International Journal

INFORMATION THEORIES & APPLICATIONS

2009 Volume 16 Number 1

International Journal
INFORMATION THEORIES & APPLICATIONS
Volume 16 / 2009, Number 1

IJ ITA is official publisher of the scientific papers of the members of the ITHEA® International Scientific Society

IJ ITA welcomes scientific papers connected with any information theory or its application.
IJ ITA rules for preparing the manuscripts are compulsory.
The **rules for the papers** for IJ ITA as well as the **subscription fees** are given on *www.ithea.org* .
The camera-ready copy of the paper should be received by http://ij.ithea.org.
Responsibility for papers published in IJ ITA belongs to authors.
General Sponsor of IJ ITA is the **Consortium FOI Bulgaria** (www.foibg.com).

International Journal "INFORMATION THEORIES & APPLICATIONS" Vol.16, Number 1, 2009

Printed in Bulgaria

Edited by the **Institute of Information Theories and Applications FOI ITHEA®**, Bulgaria,
in collaboration with the V.M.Glushkov Institute of Cybernetics of NAS, Ukraine,
and the Institute of Mathematics and Informatics, BAS, Bulgaria.

Publisher: **ITHEA®**
Sofia, 1000, P.O.B. 775, Bulgaria. www.ithea.org, e-mail: info@foibg.com

ISSN 1310-0513 (printed) **ISSN 1313-0463 (online)** **ISSN 1313-0498 (CD/DVD)**

SOLVING LARGE SYSTEMS OF BOOLEAN EQUATIONS

Arkadij Zakrevskij

Abstract. *Systems of many Boolean equations with many variables are regarded, which have a lot of practical applications in logic design and diagnostics, pattern recognition, artificial intelligence, etc. A special attention is paid to systems of linear equations playing an important role in information security problems. A compact matrix representation is suggested for such systems. A series of original methods and algorithms for their solution is surveyed in this paper, as well as the information concerning their program implementation and experimental estimation of their efficiency.*

Keywords: *Solving Boolean equations, large systems, combinatorial search*

ACM Classification Keywords: *G.2.1 Combinatorics. I.2.8. Problem solving, G.3 Probability and Statistics*

Introduction

A special type of systems of logical equations is regarded here, which seems to be very important for applications in logic design, pattern recognition and diagnostics, artificial intelligence, information security, etc. Such systems consist of many equations and Boolean variables (up to thousand and more), but with restricted number of variables in each equation (for example, not exceeding 10). That allows one to represent every equation by a rather short Boolean vector of its roots, providing a compact description of the system as a whole and efficient use of vector logical operations.

In that case each function $\varphi_i(\boldsymbol{x})$ with k arguments from some system F can be represented by a pair of Boolean vectors: 2^k-component *vector* $\boldsymbol{v}_i$ *of function values* (using the conventional component ordering) and n-component *vector* $\boldsymbol{w}_i$ *of function arguments*.

For instance, if $\boldsymbol{x} = (a, b, c, d, e, f, g, h)$, then the pair of vectors $\boldsymbol{v}_i$ = 01101010 and $\boldsymbol{w}_i$ = 00101001 represents the function $\varphi_i(c, e, h)$ which takes value 1 on four combinations 001, 010, 100 and 110 of argument values and takes value 0 on all others.

The whole system F can be represented by a pair of corresponding Boolean matrices: $(m \times 2^k)$ *matrix* $\boldsymbol{V}$ *of functions* and $(m \times n)$ *matrix* $\boldsymbol{W}$ *of arguments*.

Example 1. The system of Boolean equations

$$\varphi_1 = a'b'cd' \vee a'bc'd \vee ab'c'd$$

$$\varphi_2 = c'd'e'f' \vee c'd'e'f \vee cd'e'f' \vee cd'ef \vee cde'f \vee cdef'$$

$$\varphi_3 = e'fgh' \vee ef'g'h' \vee ef'gh \vee efgh'$$

is represented in matrix form as follows:

$$\boldsymbol{V} = \begin{matrix} 0010\ 0100\ 0100\ 0000 & v_1 \\ 1100\ 0000\ 1001\ 0110 & v_2 \\ 0000\ 0010\ 1001\ 0010 & v_3 \end{matrix} \qquad \boldsymbol{W} = \begin{matrix} a\,b\,c\,d\,e\,f\,g\,h & \\ 11110000 & w_1 \\ 00111100 & w_2 \\ 00001111 & w_3 \end{matrix}$$

Let us designate these systems as *large SLE*. It is supposed that in many applications these systems usually have few roots or none at all.

A series of original methods and algorithms for solving large SLE is presented in this survey, together with the results of their program implementation. They were published in various papers (see *References*).

Search Tree Minimization

Two combinatorial methods using tree searching technique could be applied to solve large SLE: the *equation scanning method* and the *argument scanning method*. The first method is implementing consecutive multiplication of orthogonal DNFs of the equations from a considered system and uses the search tree T_e which levels correspond to equations. The second method realizes a scanning procedure over arguments corresponding to levels of the search tree T_a. In both cases the run-time is roughly proportional to the size of the tree, i.e. to the number of its nodes. Two original algorithms were worked out that considerably reduce that number in trees T_e and T_a.

Solving large SLE can be considerably accelerating by the described below methods taking into account only the matrix of arguments $\boldsymbol{W}$ [1, 2].

Raising efficiency of the equation scanning method. In that method the nodes of i-th level of the search tree T_e represent the roots of the subsystem, constructed from the first i equations. Let us consider the set of variables, on which this subsystem depends, as $\boldsymbol{U}_i = \boldsymbol{u}_1 \cup \boldsymbol{u}_2 \cup \ldots \cup \boldsymbol{u}_i$ and denote the number of elements in $\boldsymbol{U}_i$ (in other words, the variables included in the first i equations) as $r(i)$. Then roots of the subsystem under review are the elements of the $r(i)$-dimensional Boolean space. Suppose, the functions are random, taking value 1 with probability p on every combination of argument values, independently of each other.

Affirmation 1. The expected value $M_e(i)$ of the number of nodes on the i-th level of tree T_e can be calculated as $M_e(i) = p^i 2^{r(i)}$.

In particular, the number of nodes on the last level is estimated as $M_e(m) = p^m 2^n$. (These nodes represent the solutions of the whole system)

When we include the next equation (given by function $f_{i+1}(\boldsymbol{u}_{i+1})$) into the subsystem, the set of considered variables will expand by the arguments, which are included in $f_{i+1}(\boldsymbol{u}_{i+1})$ but were not presented in any previous function. Thus, the number of possible solutions $M_e(i+1)$ can increase compared to $M_e(i)$. On the other hand, since each new equation represents a new restriction on the set of solutions, $M_e(i+1)$ may be also smaller than $M_e(i)$. The total effect of both tendencies can be represented by the following formula:

Affirmation 2. $M_e(i+1) = M_e(i)\, p\, 2^{r(i+1) - r(i)}$.

This formula shows that the increase in the number of nodes by the transition to the next level depends on the number of new arguments in $f_{i+1}(\boldsymbol{u}_{i+1})$. This number is usually much smaller than the total number of variables in $f_{i+1}(\boldsymbol{u}_{i+1})$.

The algorithm complexity for finding all solutions of the considered system is proportional to the total number of nodes in the tree T_e: $M_e = M_e(1) + M_e(2) + \ldots + M_e(m)$. The number of nodes at the last level can be determined unambiguously: $M_e(m) = p^m 2^n$. However, numbers of nodes on other levels and the total number of nodes M_e depend on the order, in which equations are considered.

We suggest the following method to decrease the algorithmical complexity. All equations are ordered by the following rule: the next equation must contain the minimum number of new variables. At the first step, the equation depending on the minimum number of arguments is selected.

Example 2. Suppose, that $p = 0.5$ and the distribution of the variables by the equations is given by the matrix ***W*** shown below. Note, that the case $p = 0.5$ corresponds to the often encountered in practice situation when characteristic Boolean functions are completely random. Considering the equations in the natural order (according to the rows of matrix ***W***), we get: $M_e(1) = 4$, $M_e(2) = 16$, $M_e(3) = 16$, etc., with the total estimated number of the nodes in the tree $M_e = 67$.

i	***W***	$M_e(i)$
1	0 1 0 0 0 1 0 1	4
2	1 0 0 0 1 1 1 1	16
3	0 1 1 0 1 0 0 1	16
4	1 0 0 1 0 0 1 0	16
5	0 1 1 0 1 0 0 0	8
6	0 1 0 0 0 1 1 0	4
7	1 0 0 1 0 1 1 0	2
8	0 0 1 0 1 0 0 0	1
	$M_e =$	67

But if we will reorder equations according to the proposed method (using the substitution (8, 5, 3, 1, 6, 2, 4, 7) on the set of rows), we will considerably decrease the computational complexity: $M_e(1) = 2$, $M_e(2) = 2$, $M_e(3) = 2$, etc., with the total estimated number of the nodes in the tree $M_e = 15$.

i	***W***	$M_e(i)$
1	0 0 1 0 1 0 0 0	2
2	0 1 1 0 1 0 0 0	2
3	0 1 1 0 1 0 0 0	2
4	0 1 0 0 0 1 0 1	2
5	0 1 0 0 0 1 1 0	2
6	1 0 0 0 1 1 1 1	2
7	1 0 0 1 0 0 1 0	2
8	1 0 0 1 0 1 1 0	1
	$M_e =$	15

Affirmation 3. Suppose that $p = 0.5$; $m = n$; $w_i^j = 1$ if $i \le j$, and $w_i^j = 0$ otherwise. In this case the search tree will contain 2^n nodes for the initial order of the equations, and only n nodes for the optimal order.

Raising efficiency of the argument scanning method. In this method we construct the search tree T_a, which shows the bifurcation hierarchy by the values of the Boolean arguments $x_1, x_2, \ldots, x_n$. Each x_j corresponds to one (and only to one) level of the tree, there are n levels in the tree T_a. The nodes on j–th level represent all input

vectors of the variables $x_1, x_2, \ldots, x_i$, for which no function of the initial system will have a zero value. Let us denote by $M_a(j)$ the expected number of nodes on level j.

Affirmation 4. $M_a(j) = 2^j \prod_{i=1}^{m} S(p, q(i, j))$,

where $S(p, r) = 1 - (1-p)^{2^r}$ is the probability that the random function with r arguments (having parameter p) is not equal to 0, and $q(i, j)$ is the number of ones in the i–th row of the matrix $\boldsymbol{W}$, located to the right from the component j.

In particular, the number of solutions of the system equals the number of nodes on the last level, which can be estimated as $M_a(n) = 2^n p^m$. The total number of nodes in tree T_a is given by the formula

$$M_a = M_a(1) + M_a(2) + \ldots + M_a(n).$$

When the number r of the arguments of a random Boolean function is increasing, the probability $S(p, r)$ that this function is not constant zero is swiftly going to 1. For example, if $p = 0.5$, then $S(p, r) = 1 - 2^{-2^r}$ ($S(0) = 1/2$, $S(1) = 3/4$, $S(2) = 15/16$, $S(3) = 255/256$, $S(4) = 65535/65536$, etc.) In practice, we can take $S(r) = 1$ if $r > 3$.

In the proposed algorithm we are optimizing the order, in which variables are selected. As the criteria of minimization, the expected number of nodes in the sequentially considered tree levels is used.

When the next level j is considered and the corresponding argument is selected, the effect of this choice is estimated in advance. Whenever some specific value of some argument is selected and substituted into the equation depending on this variable, the number u of the free variables in this equation decreases by one. As a result, the probability $S(u)$ that the equation can be satisfied, is changed for $S(u-1)$, i.e. decreases in $S(u)/S(u-1)$ times. We will use the notation $R(u)=S(u)/S(u-1)$, for example, $R(1) = 3/2$, $R(2) = 15/12$, $R(3) = 255/240$, $R(4) = 65535/65280$. In practice, we can assume that $R(u) = 1$ if $u > 4$.

Affirmation 5. During the transition from level $j - 1$ to level j the mathematical expectation of the number of nodes in the level is increasing in $2 / \prod R(q(i, j))$ times, where the product is taken by all i, for which $w_i^j = 1$.

In the proposed algorithm at each step an argument is selected such that the number of nodes in corresponding tree level is minimized. The procedure works differently, depending on whether there exists a row in the argument matrix $\boldsymbol{W}$ containing not more than 4 ones. If all rows in this matrix contain more than 4 ones, we choose rows with the minimum number of ones, and select the column j having the maximal number of ones in the chosen rows. The argument x_j is taken as the next one, and the j-th column is deleted from the further consideration. The procedure is repeated until a row will appear which contains not more than 4 ones.

To choose the next argument, we calculate the value $\prod R(q(i, j))$, using the already known values of $R(1)$, $R(2)$, $R(3)$, $R(4)$. The variable j with the maximal value of $\prod R(q(i, j))$ is selected.

Example 3. Let us consider the system from the example 1, considering the arguments in the order ($x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$). Taking into account the number of ones to the right from the position i, we obtain:

$$M_a(1) = 2 \cdot S(3) \cdot S(4) \cdot S(4) \cdot S(2) \cdot S(3) \cdot S(3) \cdot S(3) \cdot S(2) = 1.731,$$
$$M_a(2) = 4 \cdot S(2) \cdot S(4) \cdot S(3) \cdot S(2) \cdot S(2) \cdot S(2) \cdot S(3) \cdot S(2) = 2.874.$$

For other j we calculate the following values of $M_a(j)$:

$$M_a(3) = 3.463;\ M_a(4)= 5.214;\ M_a(5)= 3.693;$$
$$M_a(6) = 3.560;\ M_a(7) = 1.687;\ M_a(8)= 1.000.$$

The total number of nodes in the tree T_a is calculated as

$$M_a = M_a(1) + M_a(2) + \ldots + M_a(8) = 23.223.$$

Now, we will use another ordering of the arguments, applying the proposed algorithm. First, we will select the input variable x_5, included into equations 2, 3, 5 and 8, since for $j = 5$ the value of $\prod R(q(i, j))$ for $w_i^j = 1$ is maximal (equal to 1.333). We will delete the corresponding (x_5) column from the further consideration. At the second step, variable x_3 will be selected. The final optimized order (x_5, x_3, x_2, x_8, x_6, x_7, x_1, x_4) is represented by the following column transfer in the matrix ***W***:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8		x_5	x_3	x_2	x_8	x_6	x_7	x_1	x_4
1	0	1	0	0	0	1	0	1		0	0	1	1	1	0	0	0
2	1	0	0	0	1	1	1	1		1	0	0	1	1	1	1	0
3	0	1	1	0	1	0	0	1		1	1	1	1	0	0	0	0
4	1	0	0	1	0	0	1	0		0	0	0	0	0	1	1	1
5	0	1	1	0	1	0	0	0		1	1	1	0	0	0	0	0
6	0	1	0	0	0	1	1	0		0	0	1	0	1	1	0	0
7	1	0	0	1	0	1	1	0		0	0	0	0	1	1	1	1
8	0	0	1	0	1	0	0	0		1	1	0	0	0	0	0	0

Let us estimate the complexity of the search tree for the new argument order:

$$M_a(1) = 2 \cdot S(3) \cdot S(4) \cdot S(3) \cdot S(3) \cdot S(2) \cdot S(3) \cdot S(4) \cdot S(1) = 1.384,$$
$$M_a(2) = 4 \cdot S(3) \cdot S(4) \cdot S(2) \cdot S(3) \cdot S(1) \cdot S(3) \cdot S(4) \cdot S(0) = 1.390.$$

Similarly we calculate the next values $M_a(j)$: $M_a(3) = 1.313$; $M_a(4) = 1.395$; $M_a(5) = = 1.395$; $M_a(6) = 1.318$; $M_a(7) = 1.125$; $M_a(8) = 1.000$. Thus, we see that the expected value of the number of nodes in the tree T_a equals 10.620, which is more than twice less than it was for the initial order of arguments.

The program implementation and computer experiments confirm the high efficiency of the both methods. They show also that the argument scanning method greatly surpasses in efficiency the other one.

***

The search for solutions can be greatly facilitated by preliminary reducing the number of roots in separate equations, which, in its turn, could lead to decreasing the number of variables in a considered system and the number of equations. Three reduction methods are suggested for that, called *local reduction*, *spreading of constants* and *technique of syllogisms* [3].

The main idea of these methods consists in analyzing one by one equations of the system F, revealing there so called *k-bans* (affirmations about existence of some empty interval of the rank k in the Boolean space over the equation variables – were the equation has no root), and using them for reducing the sets of roots in other equations which, in its turn, contributes to finding new bans. That process has the chain character and can result in reducing the number of equations and variables in the system F. The method of constants spreading deals with 1-bans, the technique of syllogisms operates with 2-bans (using original deduction procedures for solving polysyllogisms), and the method of local reduction is using bans of arbitrary rank. Each of them has its own area of preferable application.

Local Reduction

This method suggested in [4] has the local nature. That means that the possibility of reduction is looked for when examining various pairs of functions $\varphi_i(\boldsymbol{u}_i)$ and $\varphi_j(\boldsymbol{u}_j)$ from the system F with intersecting sets of arguments: $\boldsymbol{u}_{i,j} = \boldsymbol{u}_i \cap \boldsymbol{u}_j \neq \varnothing$.

Let us introduce some denotations. Consider the characteristic set M_i of function $\varphi_i(\boldsymbol{u}_i)$ in the space of arguments from the set $\boldsymbol{u}_i$, and let $\boldsymbol{a}$ be its arbitrary element: $\boldsymbol{a} \in M_i$. The latter is a k-component Boolean vector, where k is the number of arguments of function $\varphi_i(\boldsymbol{u}_i)$: $k = |\boldsymbol{u}_i|$. Let $\boldsymbol{v}$ be an arbitrary subset from $\boldsymbol{u}_i$ ($\boldsymbol{v} \subseteq \boldsymbol{u}_i$) and $\boldsymbol{a}/\boldsymbol{v}$ – the *projection of element* $\boldsymbol{a}$ *onto* $\boldsymbol{v}$, i.e. the vector composed of those components of vector $\boldsymbol{a}$ which correspond to variables included in set $\boldsymbol{v}$.

The set of all different projections of elements from M_i on $\boldsymbol{v}$ is named the *projection of set* M_i *on* $\boldsymbol{v}$ and designated as $M_i/\boldsymbol{v}$. Let $M_{i,j}$ be the intersection of sets $M_i/\boldsymbol{u}_{i,j}$ and $M_j/\boldsymbol{u}_{i,j}$, and $M_{i/j}$ – the set of all such elements from M_i which projections on $\boldsymbol{u}_{i,j}$ belong to the set $M_{i,j}$.

For example, if $\boldsymbol{u}_i = (a, b, c, d, e)$, $\boldsymbol{u}_j = (c, d, e, f, g, h)$, $M_i = (01101, 11010, 10011)$ and $M_j = (101110, 001101, 010010)$, then $\boldsymbol{u}_{i,j} = \boldsymbol{u}_{j,i} = (c, d, e)$, $M_{i,j} = M_{j,i} = (101, 010)$, $M_{i/j} = (01101, 11010)$ and $M_{j/i} = (101110, 010010)$.

Let us introduce the operation $M_i := M_{i/j}$ of changing M_i for $M_{i/j}$.

Affirmation 6. For any $i, j = 1, 2, ..., m$ the operation $M_i := M_{i/j}$ is an equivalence transformation of system F, preserving the set of its roots.

Note that the application of this operation to the shown above example reduces each set M_i and M_j by one element.

Let us say that operation $M_i := M_{i/j}$ is *applicable* to an ordered pair of functions (φ_i, φ_j) if $M_i \neq M_{i/j}$. The probability of its applicability rises with increasing of the cardinality $|\boldsymbol{u}_{i,j}|$ of set $\boldsymbol{u}_{i,j}$ and goes down when $|\boldsymbol{u}_{i,j}|$ decreases. For instance, it is rather high when $|M_j| < 2^s$, where $s = |\boldsymbol{u}_{i,j}|$.

Consider now the procedure of sequential execution of this operation on pairs where it can be applied. It could terminate with reducing some of the sets M_i down to the empty set, which will mean that system F is inconsistent, or some set of reduced functions will be found where the given operation cannot be applied to any pair. This procedure is called the *local reduction* of system F.

Let us demonstrate the described algorithm of local reduction using the following example of system F.

$$\boldsymbol{V} = \begin{matrix} 0\,0\,1\,0\;\;0\,1\,0\,0\;\;0\,1\,0\,0\;\;0\,0\,0\,0 & \boldsymbol{v}_1 \\ 1\,1\,0\,0\;\;0\,0\,0\,0\;\;1\,0\,0\,1\;\;0\,1\,1\,0 & \boldsymbol{v}_2 \\ 0\,0\,0\,0\;\;0\,0\,1\,0\;\;1\,0\,0\,1\;\;0\,0\,1\,0 & \boldsymbol{v}_3 \end{matrix} \qquad \begin{matrix} a\,b\,c\,d\,e\,f\,g\,h \\ \boldsymbol{W} = \end{matrix} \begin{matrix} 1\,1\,1\,1\,0\,0\,0\,0 & \boldsymbol{w}_1 \\ 0\,0\,1\,1\,1\,1\,0\,0 & \boldsymbol{w}_2 \\ 0\,0\,0\,0\,1\,1\,1\,1 & \boldsymbol{w}_3 \end{matrix}$$

Regard in succession pairs of functions, beginning with the first one: (φ_1, φ_2). Using the operation of component-wise conjunction of corresponding rows of matrix ***W,*** we find for this pair common arguments c and d. Going through all combinations of values of these variables, we examine defined by them intervals in the space of arguments of function φ_1 (this space is presented by vector $\boldsymbol{v}_1$) and find between them intervals free of values 1 of this function. Then we delete all 1s in corresponding intervals of vector $\boldsymbol{v}_2$.

Vector representation of intervals and component-wise logical operations are used during this procedure. For example, considering combination 00 of values of variables c and d, we construct vector 1000 1000 1000 1000 which marks with 1s the corresponding interval in the space of variables a, b, c, d. Its

conjunction with vector $\boldsymbol{v}_1$ does not contain ones, therefore equation $\varphi_1 = 1$ has no roots in this interval. The respective interval in the space of arguments of function φ_2 is represented by vector 1111 0000 0000 0000, inasmuch as variables c and d take now left positions. All ones contained in this interval are deleted from vector $\boldsymbol{v}_2$, so the latter receives the value 0000 0000 1001 0110.

These operations could be presented in a more compact form, by the formula

$$C'\,d'\,\varphi_1 = 0 \;\rightarrow\; \boldsymbol{v}_2 := 0000\ 0000\ 1001\ 0110\ .$$

Continuing the reduction algorithm, we perform one by one the following operations presented similarly:

$$\begin{aligned}
c\,d\,\varphi_1 = 0 &\;\rightarrow\; \boldsymbol{v}_2 := 0000\ 0000\ 1001\ 0000\ ,\\
c'\,d\,\varphi_2 = 0 &\;\rightarrow\; \boldsymbol{v}_1 := 0010\ 0000\ 0000\ 0000\\
e'\,f\,\varphi_2 = 0 &\;\rightarrow\; \boldsymbol{v}_3 := 0000\ 0000\ 1001\ 0010\\
e\,f'\,\varphi_2 = 0 &\;\rightarrow\; \boldsymbol{v}_3 := 0000\ 0000\ 0000\ 0010\\
e'\,f'\,\varphi_3 = 0 &\;\rightarrow\; \boldsymbol{v}_2 := 0000\ 0000\ 0001\ 0000
\end{aligned}$$

As a result, the initial system of Boolean functions is reduced to the following one:

$$\boldsymbol{V} = \begin{matrix} 0\,0\,1\,0\ \ 0\,0\,0\,0\ \ 0\,0\,0\,0\ \ 0\,0\,0\,0 & \boldsymbol{v}_1\\ 0\,0\,0\,0\ \ 0\,0\,0\,0\ \ 0\,0\,0\,1\ \ 0\,0\,0\,0 & \boldsymbol{v}_2\\ 0\,0\,0\,0\ \ 0\,0\,0\,0\ \ 0\,0\,0\,0\ \ 0\,0\,1\,0 & \boldsymbol{v}_3 \end{matrix}$$

from where the unique root of the system is easily obtained: 00101110.

Spreading of Constants

This method can be regarded as a simplified version of the local reduction. It can be efficiently used when the number of roots in some equations $\varphi_k = 1$ is very small. In that case it is enough to look only for 1-bans regarding separate literals x_i and x_i' and checking them consecutively for satisfying equations.

When $x_i \wedge \varphi_k = 0$ for function φ_k in some equation of the system, 1-ban $x_i = 0$ is found. In that case value 0 is assigned to variable x_i (value 1 in the case $x_i' \wedge \varphi_i = 0$), and the latter is changed for constant 0 (or 1) in all other equations. In such a way finding constants is followed by their spreading over the whole system F. Replacing some variables by constants usually decreases the number of roots in regarded equations which, in its turn, helps to discover new constants. So, the process of constants spreading has the cyclic chain nature. As a result, the dimension of processed equations is decreased, sometimes down to zero – when all variables of the regarded equation receive definite values. If function φ_k turns into 1, the corresponding equation is deleted from the system; if φ_k turns into 0, it becomes evident that the system is inconsistent.

Simple enough, this method turned out to be very efficient, being applied to some problems of cryptology. A special problem of cryptanalysis of the mechanical rotor encryption machine Hagelin M-209-B, which was used in several forms by Germans during the second world war, was investigated in [5]. It was shown that its

cryptanalysis can be reduced to solving a definite system of many Boolean equations (about five hundred) each of which contains six Boolean variables, meanwhile the general number of variables equals 131 – the set of their values constitutes the sought-for key. To solve this system a method was proposed in [5] based on using Reduced Ordered Binary Decision Diagrams (ROBDDs) for representation of the regarded functions. Its computer implementation on Pentium Pro 200 showed that under some suppositions it enables to find the key in several minutes.

Application of the method of spreading of constants using vector representation of the considered Boolean functions and taking into account the specific of the regarded system of logical equations turned out to be considerably more efficient. It accelerates the search for the key more than in thousand times [6].

Technique of Syllogisms

Here 2-bans are looked-for and used in the reduction procedure. Besides, the latter takes into account all logical consequences deduced from the set of found 2-bans by syllogisms [7]. An improved technique of polisyllogisms is applied for that [8].

Let us regard equation $\varphi(z_1, z_2, \ldots, z_k) = 1$ with function φ taking value 1 on s randomly selected inputs. When s is small, it is possible to find some constant, which prohibits the value of some variable (1-ban). But it is more probable to reveal a prohibition on some combination of values of two variables (2-ban), which determines the corresponding *implicative regularity*, or connection between these variables. For example, connection "if *a*, then not *b*" prohibits combination of values $a = 1$, $b = 1$. It could be revealed in φ if $ab\varphi = 0$. For convenience, represent this ban by product ab (having in mind equation $ab = 0$).

In a similar way, 2-bans ab', $a'b$, $a'b'$ are defined. They are interpreted easily as general affirmation and negation category statements. By that besides three such statements of Aristotle syllogistic (ab' – all *A* are *B*, $a'b$ – all *B* are *A*, ab – none of *A* is *B*) the fourth is also used: $a'b'$ – none of objects is *A* and is not *B*. Such a statement was not considered by Aristotle, inasmuch as he did not regard empty classes [9].

Suppose, that by examining equations of the system F one by one, we have found a set P of 2-bans. Let us consider the task of *closing* it, i. e. adding to it all other 2-bans which logically follow from P (so called *resolvents* of P). This task is equivalent to the polysyllogistic problem. Denote the resulting *closed set of 2-bans* as $Cl(P)$. A method to find it is suggested below. It differs from the well-known method of resolution and its graphical version by application of vector-matrix operations which speed up the logical inference.

Let X_t^1 and X_t^0 be the sets of all literals that enter 2-bans contained in F together with literal x_t or x_t', correspondingly. We introduce operator Cl_t of *partial closing* of set P in regard to variable x_t, extending this set by uniting it with direct product $X_t^1 \times X_t^0$ containing results of all possible resolutions by this variable.

Affirmation 7. $Cl_t(P) = P \cup X_t^1 \times X_t^0 \subseteq Cl(P)$.

Affirmation 8. $Cl(P) = Cl_1 Cl_2 \ldots Cl_n (P)$.

In such a way, the set P can be closed by separate variables, one by one.

The set P can be represented by a square Boolean matrix $\boldsymbol{P}$ of the size $2n$ by $2n$, with rows $\boldsymbol{p}_{t1}$, $\boldsymbol{p}_{t0}$ and columns $\boldsymbol{p}^{t1}$, $\boldsymbol{p}^{t0}$ corresponding to literals x_t, x_t', $t = 1, 2, \ldots, n$. Elements of matrix $\boldsymbol{P}$ correspond to pairs of literals, and non-diagonal elements having value 1 represent discovered 2-bans. So, the totality of 1s in row $\boldsymbol{p}_{t1}$ (as well as in column $\boldsymbol{p}^{t1}$) indicates set X_t^1, and the totality of 1s in row $\boldsymbol{p}_{t0}$ (column $\boldsymbol{p}^{t0}$) indicates set X_t^0. Using vector operations, we can construct the matrix $\boldsymbol{P}^+$, presenting the result of closing operation: $\boldsymbol{P}^+ = Cl(P)$.

For example, if $\boldsymbol{x} = (a, b, c, d)$ and 2-bans ab', ac, $a'd'$, bc' are found forming set P, then

```
a a' b b' c c' d d'          a a' b b' c c' d d'
0 0  0 1  1 0  0 0  a                 c 0  c 1  1 b  0 c    a
0 0  0 0  0 0  0 1  a'          0 0  0 0  0 0  0 1   a'
0 0  0 0  0 1  0 0  b                 c 0  0 0  0 1  0 c    b
  P =  1 0  0 0  0 0  0 0  b'   P+ = 1 0  0 0  0 0  0 a    b'
1 0  0 0  0 0  0 0  c                 1 0  0 0  0 0  0 a    c
0 0  1 0  0 0  0 0  c'          b 0  1 0  0 0  0 b    c'
0 0  0 0  0 0  0 0  d                 0 0  0 0  0 0  0 0    d
0 1  0 0  0 0  0 0  d'          c 1  c a  a b  0 c    d'
```

– the bans-consequences are marked in matrix **P+** by symbols of variables by which the corresponding resolutions were executed.

The closed set *Cl*(*P*) could be found also by the *increment algorithm of expansion* of *P* : every time when a new 2-ban *p* is added by a special operation *ins*(*p, P*) all resolvents are included in *P,* too. In that case after each step the set *P* will remain closed: *P* = *Cl*(*P*).

Operation *ins*(*p, P*) is defined as follows.

Affirmation 9. If $P = Cl(P)$, than $Cl(P \cup \{p\}) = P \cup D$, where

$D = (\{x\} \cup X0) \times (\{y\} \cup Y0)$, if $p = xy$,

$D = (\{x\} \cup X0) \times (\{y'\} \cup Y1)$, if $p = xy'$,

$D = (\{x'\} \cup X1) \times (\{y\} \cup Y0)$, if $p = x'y$,

$D = (\{x'\} \cup X1) \times (\{y'\} \cup Y1)$, if $p = x'y'$.

Consider now the problem of finding all *prime bans* (which do not follow from one another) deduced from system *P*. It is known that no set of 2-bans can produce any bans of higher rank. But it can produce some 1-bans, prohibiting definite values of separate variables.

Affirmation 10. All 1-bans deduced from set *P* are represented by 1-elements of the main diagonal of matrix $\boldsymbol{P}^+$. In the regarded example 1-bans *a* and *d*' are presented in such a way.

Affirmation 11. If the pair of 1-bans *x* and *x'* is found for some variable *x*, the system *F* is inconsistent.

Note that inconsistency of *F* follows from inconsistency of *P*, but not vice versa.

Based on the technique of syllogisms an efficient reduction method was developed, dealing with a set of logical equations *F*, empty at the beginning. It examines the equations in cyclic order, reduces the set of roots of the current equation $f_j = 1$ by considering bans enumerated in *P* (prohibited roots are deleted) and looks there for new 2-bans not existing in *P*. These bans are added to *P*, at the same time operation of closing *P* is performed. By that some variables can receive unique values – when 1s appear on the main diagonal of matrix **P** (1-bans are found). The procedure comes to the end when inconsistency is revealed (0-ban is found represented by a pair of 1s on the main diagonal of **P**) or when processing *m* equations one by one turns out to be unsuccessful. In that case we have as a result a reduced system of equations equivalent to the initial one.

Computer Experiments

Extensive computer experiments were conducted on PC Pentium 100 to evaluate the efficiency and applicability of the suggested reduction methods [10, 11]. A series of pseudo-random consistent (having at least one root) systems of Boolean equations with given parameters (m – the number of equations, n – the total number of variables, k – the number of variables in each equation and p – the relative number of roots in equations) was generated [12] and subjected to the reduction procedures programmed in C++. Two important results were obtained by that.

First, the *avalanche* effect of reduction was revealed experimentally, both for the local reduction and the technique of syllogisms. When we conduct experiments for fixed values of p, n and k, gradually increasing m, it turns out that for some crucial value of m an avalanche occurs. It means that the number of roots in the equations dramatically decreases in such a high degree that it could be easy to find the complete solution of the regarded system. This effect is well shown on Table 1, where partial results of some experiments are presented. Note that q s the average number of remaining roots in one equation after the reduction. Evidently, if q = 1, the system has only one root, and it is found.

Table 1. Examples illustrating avalanche effect of the reduction procedures

Local reduction

Experiment 1. p = 1/2, n = 50, k = 5. (m: q) = 113: 8.19, 114: 8.21, 115: 1.

Experiment 2. p = 1/2, n = 50, k = 6. (m: q) = 298: 30.02, 299: 1.

Experiment 3. p = 1/4, n = 100, k = 6. (m: q) = 167: 13.88, 168: 1.

Experiment 4. p = 1/4, n = 100, k = 7. (m: q) = 390: 28.53, 391: 1.

Experiment 5. p = 1/4, n = 200, k = 6. (m: q) = 384: 14.29, 385: 1.

Experiment 6. p = 1/8, n = 200, k = 6. (m: q) = 72: 7.93, 73: 1.09.

Experiment 7. p = 1/8, n = 200, k = 7. (m: q) = 196: 13.25, 197: 1.

Technique of syllogisms

Experiment 8. p = 1/2, n = 50, k = 5. (m: q) = 74: 13.32, 75: 1.07.

Experiment 9. p = 1/4, n = 100, k = 6. (m: q) = 85: 14.25, 86: 1.05.

Experiment 10. p = 1/8, n = 200, k = 7. (m: q) = 128: 15.76, 129: 1.02.

Second, the existence of avalanche effect enables practically for every combination of values of parameters p, n, k to find the crucial value m_c indicating the number of equations m at which the system collapses under the influence of the reduction procedure. Assume that such a collapse occurs when q becomes less than 1.1.

That crucial value m_c is shown below (the first number in a pair playing the role of the table element) both for local reduction and technique of syllogisms, for p = 1/2, 1/4 and 1/8, as the function of n and k. The run-time t in seconds is presented by the second number in the pair. For instance, if n = 80 and k = 7, then for local reduction and p = 1/4 it follows that m_c = 245 and t = 54 s. These results show that the area of applicability of the suggested method is rather broad, up to thousand variables under certain conditions.

Table 2. Dependence of the crucial value m_c of the number of equations m and the run-time t on the total number of variables n, the number of variables k and the density of roots p in separate equations

Local reduction

	n	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
	20	38-1	30-2	34-6	41-10	37-39	64-173	60-431
	40	67-3	74-8	195-57	426-404			
	80	144-10	273-47	1355-989				
$p=1/2$	160	372-33	1094-973					
	320	655-96						
	640	1245-358						
	1280	1500-647						
	20	14-0	13-0	10-0	13-0	13-2	13-5	17-5
	40	20-0	26-1	26-1	45-4	113-41	204-230	431-2493
	80	40-1	46-1	78-5	245-54	1031-1368		
$p=1/4$	160	85-2	123-4	314-36	1226-648			
	320	250-10	250-17	821-174				
	640	550-42	475-60	2265-1146				
	1280	1100-179	1000-251					
	2560	2200-995						
	20	10-0	10-0	10-0	10-0	10-0	10-1	10-3
	40	25-0	21-0	15-1	17-1	22-2	36-8	52-27
	80	35-1	30-1	25-1	43-2	123-18	256-130	1008-2598
$p=1/8$	160	82-1	67-1	74-2	134-9	662-267	2287-4000	
	320	199-4	166-5	158-6	346-42			
	640	415-19	329-15	352-29	705-129			
	1280	749-74	736-96	742-137	1568-673			
	2560	1965-633	1802-626	1423-565				

Technique of syllogisms

	n	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$
	20	30-0	39-1			
	40	42-0	73-2			
	80	124-2	217-7			
$p=1/2$	160	217-5	615-23			
	320	573-25	1446-61			
	640	1191-166				
	1280	2438-1371				
	20	14-0	13-0	9-1	471-33	
	40	28-0	20-2	24-2	321-25	
	80	50-1	47-1	39-4	620-53	
$p=1/4$	160	129-3	109-5	166-12	1825-142	
	320	296-24	333-26	452-44		
	640	797-201	663-186	894-220		
	1280	1371-1585				
	20	10-0	10-0	10-0	10-1	86-11
	40	18-0	21-1	14-1	27-3	177-23
	80	40-0	31-1	32-2	46-5	357-52
$p=1/8$	160	111-4	78-4	79-7	101-15	917-125
	320	271-26	267-29	212-32	227-46	
	640	596-226	537-223	409-166	413-187	
	1280	1027-1701				

Systems of Linear Logical Equations – Finding Shortest Solutions

In general case any system of linear logical equations (SLLE) can be presented as

$$a_1^1x_1 \oplus a_1^2x_2 \oplus \ldots \oplus a_1^nx_n = y_1 ,$$
$$a_2^1x_1 \oplus a_2^2x_2 \oplus \ldots \oplus a_2^nx_n = y_2 ,$$
$$a_m^1x_1 \oplus a_m^2x_2 \oplus \ldots \oplus a_m^nx_n = y_m ,$$

or in a more compact form of one matrix equation

$$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{y}.$$

Here $\boldsymbol{A}$ is a Boolean $(m\times n)$-matrix of coefficients, $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ – a Boolean vector of unknowns and $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$ – a Boolean vector of free terms. The operation of multiplying matrix $\boldsymbol{A}$ by vector $\boldsymbol{x}$ is defined as follows:

$$\bigoplus_{j=1}^{n} a_i^j x_j = y_i , \quad i = 1, 2, \ldots, m.$$

Suppose $\boldsymbol{A}$ and $\boldsymbol{y}$ are known and vector $\boldsymbol{x}$ is to be found. It is accepted usually, that the problem consists in finding a *root* of the system – a value of vector $\boldsymbol{x}$, satisfying all equations, i.e. turning them into identities. However, when several roots exist, a problem arises to choose one of them, optimal in some sense.

Alongside with two parameters m and n the third parameter of an SLLE is important – the *rank* r, i.e. the maximal number of linearly independent columns in matrix $\boldsymbol{A}$. Remind, that a set of Boolean vectors is called *linearly independent* if the component-wise sum (modulo 2) of any of its elements differs from zero. It is known that the rank equals as well the maximal number of linearly independent rows in the same matrix. The relations between parameters m, n and r determine if the system has some roots and how many of them.

In case $n = m = r$ the system has exactly one root and is called *defined*, or *deterministic*. When $n < m$, the system could have no roots and is called in this case *over-defined*, or *inconsistent*, or *contradictory*. When $n > m$, the system has 2^{n-r} roots and is called *undefined*, or *non-deterministic*.

In this section the last case is considered, and the optimization task of finding a *shortest* solution (with minimum number of ones in vector $\boldsymbol{x}$) is to be solved. That task has important application at design of linear finite automata [13] and logic circuits synthesized in Zhegalkin basis and possessing such attractive properties, as good testability and compactness at implementation of arithmetic operations [14]. It is useful also when solving information security problems [15, 16].

A simplest algorithm. Let us suppose that a regarded system is undefined and $m = r$, i.e. all rows of matrix of coefficients $\boldsymbol{A}$ are linearly independent. In that case a shortest solution could be found by means of selecting from matrix $\boldsymbol{A}$ one by one all different combinations of columns, consisting first of 1 column, then of 2 columns, etc. and examining them to see if their sum equals vector $\boldsymbol{y}$. As soon as it happens, the current combination is accepted as the sought-for solution.

That moment could be forecasted. If the *weight* of the shortest solution (the number of 1s in vector ***x***) is *w*, the number *N* of checked combinations is defined approximately by the formula

$$N = \sum_{i=0}^{w} C_n^i$$

and could be very large, as is demonstrated below.

It was shown in [17] that the expected weight γ of the shortest solution of an SLLE with parameters *m* and *n* can be estimated before finding the solution itself. We represent this weight as the function $\gamma(m, n)$. First we find the mathematical expectation $\alpha(m, n, k)$ of the number of solutions with weight *k*. We assume that the considered system was randomly generated, which means that each element of ***A*** takes value 1 with the probability 0.5 and any two elements are independent of each other. Then the probability that a randomly selected column subset in matrix ***A*** is a solution equals 2^{-m} (probability that two randomly generated Boolean vectors of size *m* are equal). Since the number of all such subsets having *k* elements equals C_n^k, we get:

$$\alpha(m, n, k) = C_n^k 2^{-m}, \text{ where } C_n^k = n! / ((n-k)!\, k!).$$

Similarly, we denote as $\beta(m, n, k)$ the expected number of the solutions with weight not greater than *k*:

$$\beta(m, n, k) = \sum_{i=0}^{k} C_n^i 2^{-m}.$$

Now, the expected weight γ of the shortest solution can be estimated well enough by the maximal value of *k*, for which $\beta(m, n, k) < 1$:

$$\gamma(m, n) = k,$$

where $\beta(m, n, k) < 1 \le \beta(m, n, k+1)$

For example, the values of α and β for the system of 40 equations with 70 variables and the values of *k* from 7 to 13 are shown in Table 3.

k	α	β
7	0.001	0.001
8	0.009	0.010
9	0.059	0.069
10	0.361	0.430
11	1.968	2.398
12	9.676	12.074
13	43.170	55.244

It is clear enough that the weight of the shortest solution for this system will be probably equal to 10.

Unfortunately, the described above simple algorithm could appear too difficult to implement. Regarding another example with $m = 100$ and $n = 130$ we find that $\gamma = 31$, and the number N of checked combinations is about 10^{30}. Examining them with the speed of one million combinations per second we need about **30 000 000 000 000 000 years** to find the solution. Too much!

Gaussian method. The well-known Gaussian method of variables exclusion [18] was developed for solving systems of linear equations with real variables, and is adjusted here for Boolean variables. It enables to avoid checking all 2^n subsets of columns from Boolean matrix $\boldsymbol{A}$ which have up to w columns, when only one of 2^{n-m} regarded combinations presents some root of the system.

Its main idea consists in transforming the extended matrix of the system (matrix $\boldsymbol{A}$ with the added column $\boldsymbol{y}$) to the *canonical form*. A maximal subset of m linear independent columns (does not matter which one) is selected from $\boldsymbol{A}$ and by means of equivalent matrix transformations (adding one row to another) is transformed to **I**-matrix, with 1s only on the main diagonal. That part is called a *basic*, the rest $n - m$ columns of the transformed matrix $\boldsymbol{A}$ constitute a *remainder*. The column $\boldsymbol{y}$ is changed by that, too.

According to this method the subsets of the remainder are regarded, i.e. combinations selected from the set which has only $n - m$ columns (not all n columns!). It is easy to show that every of these combinations enables to get a solution of the considered system. Indeed, any sum (modulo 2) of its elements can be supplemented with some columns from **I** to make it equal to $\boldsymbol{y}$.

When we are looking for a shortest solution (solution with the minimum weight) using this method, described in detail in [19], we have to consider different subsets of columns from the remainder, find the solution for each such subset and select a subset, which generates the shortest solution. If it is known that the weight of the shortest solution is not greater than w, then the level of search (the cardinality of inspected subsets) is restricted by w. Note that if $w \geq n - m$, then all 2^{n-m} subsets must be searched through.

For the same example ($m = 100$ and $n = 130$) $N \cong 10^9$, which means that the run-time of Gaussian method is about **17 minutes**.

Decomposition method. An additional gain can be received by decomposition of the process of solution, at which instead of one canonical form of matrix $\boldsymbol{A}$ several canonical forms are considered. That idea was realized before by the author who suggested a decomposition method for finding shortest solutions [17]. That method is based on constructing a set of different but equivalent canonical forms of the regarded SLLE and solving them in parallel until a shortest solution is found. The run-time of the implementation program depends much on the level of combinatorial search, and the lowering of this level can greatly accelerate the search process.

Let us assume that we can find q maximal subsets of linear independent columns in matrix $\boldsymbol{A}$, such that the corresponding remainders do not intersect. In this case $n \geq q(n - m)$. The following method can be used, which was called the *method of non-intersecting remainders*.

Let us construct the set Q, consisting of q canonical forms, such that the basics of these forms are obtained using the considered subsets. We will search for the optimal solution within the set Q, with the subsequent increase in the level of search up to some value.

Affirmation 12. A canonical form always exists in Q, such that a shortest solution can be found on the search level not greater than $\lfloor \gamma/q \rfloor$.

Affirmation 13. A shortest solution for the given system can be found by the subsequent consideration of the remainders of the canonical forms from the set *Q*, restricting the search level by the value $\lfloor (w-1)/q \rfloor$, where w is the weight of the shortest already found solution.

Based on these statements the decomposition method was proposed to find a shortest solution of a system of linear logical equations. Using this method we search through the subsets in all remainders first on level 0, then on level 1, etc. until the level of search reaches $\lfloor (w-1)/q \rfloor$(the nearest integer from below).

Affirmation 14. The number $N_r(m, n)$ of the subsets of the columns, which are considered using the not-intersecting remainders method, is defined by the formula:

$$N_r(m, n) = q \sum_{i=0}^{p} C_{n-m}^{i},$$

where $p = \lfloor \gamma (m, n) / q \rfloor$.

For the same example (m = 100 and n = 130), q = 4, γ = 31 and $p = \lfloor 31/4 \rfloor = 7$. In this case $N_r \cong 10^7$, that means that the run-time of this method is about **ten seconds**.

Recognizing short solutions. A much bigger progress in the run-time saving can be achieved in the case when some short solution exists, with weight w perceptibly smaller than γ [19].

Such a solution which satisfies the relation $w < \gamma$ or even $w < \gamma - 1$ could be immediately recognized and accepted without any additional proof. That enables to increase considerably the size of regarded and solved systems, which is measured in number of equations and variables (m and n).

Consider, for example, a random SLLE with m = 300 and n = 350 with expected weight of a shortest solution γ = 101. In general case such solution could be found on the level of search 21, and we should spend about **61 years** to find it by the decomposition method (examining one million combinations per second). However, when a solution with weight 70 exists, it can be found and recognized on the level of search 7 in 7 minutes, and a solution with weight 35 can be found on level 2 in **only 0.5 seconds**.

Randomized parallelization. A new version of the decomposition method was suggested in [20, 21], in which a set of canonical forms is prepared beforehand, all different but equivalent to the given one. They have various basics specified by some maximal linearly independent subsets of columns of matrix A, selected *at random*, independently of each other. In such a way the process of looking for a shortest solution is randomized. The number q of used canonical forms could be arbitrary, being chosen by some additional considerations.

A solution is searched in parallel over all these forms, first at the level 0 of exhaustive search, then at the level 1, etc., until at the current level k a solution with weight w, satisfying condition $w < \gamma - 1$ will be found. With raising q this level k can be reduced, which reduces the run-time as well.

Suppose there exists a solution with weight w. The chances to detect it at level k of exhaustive search can be estimated as follows. Consider an n-component Boolean vector $\boldsymbol{a}$, with a randomly selected $(n-m)$-component sub-vector $\boldsymbol{a}'$. There exist C_n^w (the number of different combinations from n by w) values of vector $\boldsymbol{a}$ each of which has exactly w ones. Let us assume that all of them are equiprobable. The number of those of them, which have

exactly k ones in vector $\boldsymbol{a}'$ ($k \leq n - m$ by that), is evaluated by the formula $C_{n-m}^{k} C_{m}^{w-k}$, and the number N_k of those which have no more than k ones in vector $\boldsymbol{a}'$ is evaluated by the formula

$$N_k = \sum_{j=0}^{k} C_{n-m}^{j} C_{m}^{w-j}$$

Evidently,

$$C_n^w = \sum_{i=0}^{\min(w,n)} C_{n-m}^{i} C_{m}^{w-i}$$

and the formula

$$P = 100\, N_k / C_n^w$$

shows the percentage of situations wgere a short solution with weight w can be found at the level of search k.

For example, in Table 4 are shown the calculated values received by P for different levels of search k at $m = 420$, $n = 500$ and $w = 75$.

The following conclusion could be deduced from that table. Preparing beforehand $q = 100$ random canonical forms of the considered SLLE with given parameters, we could hope to find the solution on level 5 or 6. In that case about 25 or 300 million combinations should be checked for every of 100 canonical forms.

Table 4. Evaluation of chances to detect a shortest solution at given level of search k ($m = 420$, $n = 500$ and $w = 75$).

k	1	2	3	4	5	6	7	8	9	10	11	12	13
P	0.00	0.01	0.06	0.23	0.85	2.41	5.62	11.27	19.84	23.12	36.36	50.00	62.34

Programming and experiments. The suggested randomized parallel algorithm was programmed and verified (C++, PC COMPAC Presario – processor Intel Pentium III, 1000 MH). The dependence of the run-time T on the number q of randomly selected canonical forms was investigated for different systems of linear equations. Some results are presented below.

Thirty different random SLLEs with $m = 900$ and $n = 1000$ were generated (each having a solution with weight $w = 100$) and solved, using q randomly chosen canonical forms for every system, for different q: 1, 10, 30 and 300. The following parameters of the solution process were measured and shown in Table 5:

N – the ordinal number of the solved SLLE, L – the level of search at which the solution was found, F – the number of canonical form where the solution was found, T – the time spent for finding the solution (measured in seconds (s), minutes (m), hours (h), days (d) and years (y).

For instance, the short solution with $w = 100$ was found for SLLE number 22 in 6 minutes, at the level of search 3 while solving the canonical form number 190.

Note that the last parameter T was found not immediately but forecasted according to the method described in [22], which changes the real solution process for a virtual one. That saves much time spent for the experiment.

The positive result of increasing the number q of canonical forms is evident: at $q = 300$ every of the considered 30 examples is solved in several minutes – instead of many (thousands sometimes) years at $q = 1$.

Solving Over-defined Systems

It can appear, that the regarded SLLE has no root – when it is over-defined (inconsistent, or contradictory, when usually $m > n$). In that case it is possible to put the task of finding a value of vector x, fitting to maximum number of the equations and accepted therefore as a solution of the system. Such task arises at development of information security systems and can be interpreted as follows. Suppose, the appropriate value of vector y is received for given A and x, and then distorted (in components marked by ones in the vector of distortions e). As a result a vector $z = y \oplus e$ appears, whereas x and y are "forgotten". It is required to restore the initial situation on known now values A and z.

This task was solved in [23], where it was reduced to finding a shortest root of an undefined SLLE obtained from the initial over-defined SLLE by appropriate transformation of matrix A and vector y. The boundaries of correct setting of the task were defined in [16], within which the values of x and y can be restored practically uniquely. A new method of solution of the given task was offered in [24], based on compact representation of the processed information and usage of the procedure of random sampling. The given over-defined SLLE is converted by that to other over-defined SLLE equivalent to initial one but solved more easily.

Table 5. The results of solving undefined SLLEs with parameters $n = 1000$, $m = 900$, $w = 100$.

	q=1			q=10			q=30			q=300		
N	L	F	T	L	F	T	L	F	T	L	F	T
1	1	8	7d	9	6	16h	9	6	18h	33	4	19m
2	1	9	69d	8	6	14h	16	5	2h	254	2	4m
3	1	10	2y	7	7	7d	28	6	2d	234	2	4m
4	1	13	776y	3	6	5h	3	6	8h	117	3	5m
5	1	3	4s	1	3	4s	1	3	5s	1	3	4m
6	1	10	2y	5	7	5d	26	5	3h	56	2	4m
7	1	12	112y	9	4	3m	9	4	3m	57	3	5m
8	1	8	7d	10	5	1h	10	5	1h	106	3	5m
9	1	12	112y	10	5	1h	28	4	10m	141	3	6m
10	1	9	69d	2	6	4h	2	6	6h	95	2	4m
11	1	13	776y	7	8	82d	15	4	5m	50	2	4m
12	1	13	776y	9	8	104d	14	5	2h	117	3	5m
13	1	10	2y	8	6	14h	8	6	16h	35	3	4m
14	1	6	58m	1	6	2h	1	6	4h	39	3	4m
15	1	6	58m	1	6	2h	11	5	1h	134	3	6m
16	1	14	4954y	2	8	27d	28	4	10m	205	3	7m
17	1	6	58m	1	6	2h	14	5	2h	49	3	4m
18	1	10	2y	2	7	2d	27	5	3h	285	2	4m
19	1	13	776y	8	7	8d	8	7	9d	84	3	5m
20	1	10	2y	2	6	4h	2	6	6h	203	4	1,3h
21	1	8	7d	7	5	45m	7	5	52m	93	4	39m
22	1	7	13h	10	6	17h	27	4	9m	190	3	6m
23	1	12	112y	2	5	13m	16	2	4s	16	2	4m
24	1	15	29185y	4	7	4d	24	6	2d	226	2	4m
25	1	10	2y	2	6	4h	2	6	6h	46	3	4m
26	1	13	776y	9	7	9d	16	5	2h	112	4	45m
27	1	10	2y	6	8	71d	16	4	6m	281	3	8m
28	1	12	112y	7	8	82d	18	6	1d	87	3	5m
29	1	12	112y	6	4	2m	6	4	2m	254	2	4m
30	1	12	112y	7	8	82d	22	6	2d	31	4	18m
Sum:			38704y			1,3y			20d			5,3h

Bibliography

1. Zakrevskij A., Zakrevski L. Solving systems of logical equations using search tree minimization technique. – Proceedings of the PDPTA'02 International Conference, June 24-27, 2002, Las Vegas, USA. – pp. 1145-1150.
2. Zakrevskij A., Vasilkova I. Reducing search trees to accelerate solving large systems of Boolean equations. – Boolean Problems // 5-th International Workshop, Sept. 19-20, 2002, Freiberg (Sachsen). – pp. 71-76.
3. Zakrevskij A. Reduction algorithms for solving large systems of logical equations. – Computer Science Journal of Moldova, 2000, v. 8, No 1. – pp. 3-15.
4. Zakrevskij A.D. Solving systems of logical equations by the method of local reduction. – Doklady NAN B, 1999, v. 43, No 5, pp. 5-8. (in Russian).
5. Baumann M., Rohde R., Barthel R. Cryptanalysis of the Hagelin M-209 Machine. – 3rd International Workshop on Boolean Problems, Sept. 17-18, 1998, Freiberg (Sachsen), pp. 109-116.
6. Zakrevskij A.D., Vasilkova I.V. Cryptanalysis of the Hagelin machine by the method of spreading of constants. – Proceedings of the Third International Conference on Computer-Aided Design of Discrete Devices (CAD DD'99), Minsk, November 10-12, 1999, vol. 1. – pp. 140-147.
7. Zakrevskij A.D. Solving large systems of logical equations by syllogisms. – Doklady NAN B, 2000, v. 44, No 3, pp. 40-42 (in Russian).
8. Zakrevskij A.D. To formalization of polysyllogistic. – Logical Inference, Moscow: Nauka, 1979, pp.300-309 (in Russian).
9. Lukasiewich J. Aristotle syllogistic from the point of view of modern formal logic. – Moscow, 1959 (in Russian).
10. Zakrevskij A., Vasilkova I. Reducing large systems of Boolean equations. – 4-th International Workshop on Boolean Problems, September 21-22, 2000, Freiberg, Germany. – pp. 21-28.
11. Zakrevskij A. D. Solving large systems of logical equations. – Sixth ISTC Scientific Advisory Committee Seminar "Science and Computing", Moscow, Russia, 15-17 September 2003. – Proceedings, volume 2, pp. 528-533.
12. Zakrevskij A.D, Toropov N.R. Generators of pseudo-random logical-combinatorial objects in C++. - Logical Design, No 4, 1999, Minsk, Institute of Engineering Cybernetics, pp. 49-63 (in Russian).
13. Gill A. Linear sequential circuits. McGraw-Hill Book Co., New York, 1966.
14. Zakrevskij A.D., Toropov N.R. Polynomial implementation of partial Boolean functions and systems. – Moscow, URSS, 2003 (in Russian).
15. Balakin G.V. Introduction into the theory of random systems of equations. – Proceedings on discrete mathematics, Moscow, TVP, 1997, vol. 1, pp. 1-18 (in Russian).
16. Zakrevskij A. Solution of a system of linear logical equations with distorted right members – when it could be found. – New Information Technologies. Proceedings of the Fifth International Conference NITe'2002, Minsk, BSEU. – Vol. 1, pp. 54-58.
17. Zakrevskij A. D., Zakrevski L. Optimizing solutions in a linear Boolean space – a decomposition method // Proc. of STI '2003, Orlando, Florida, USA, July 2003, pp. 276-280.
18. Gauss C.F. Beitrage zur Theorie der algebraischen Gleichungen. – Gött., 1849
19. Zakrevskij A.D. Looking for shortest solutions of systems of linear logical equations: theory and applications in logic design. – 2. Workshop "Boolesche Probleme", 19./20. September 1996, Freiberg/Sachsen, pp. 63-69.
20. Zakrevskij A.D. Randomization of a parallel algorithm for solving undefined systems of linear logical equations. – Proceedings of the International Workshop on Discrete-Event System Design – DESDes'04. – University of Zielona Gora Press, Poland, 2004, pp. 97-102.
21. Zakrevskij A.D. Raising efficiency of combinatorial algorithms by randomized parallelization. – XI-th International Conference "Knowledge-Dialogue-Solution" KDS – 2005, June 20-30, 2005, Varna, Bulgaria, pp. 491-496.
22. Zakrevskij A.D., Vasilkova I.V. Forecasting the run-time of combinatorial algorithms implementation. – Methods of logical design, 2003, issue 2. Minsk: UIIP of NAS of Belarus, pp. 26-32 (in Russian).
23. Zakrevskij A.D. Solving inconsistent systems of linear logical equations. – 6-th International Workshop on Boolean Problems, September 23-24, 2004, Freiberg (Sachsen), pp. 183-190.
24. Zakrevskij A.D. A new algorithm to solve overdefined systems of linear logical equations. – Computer-Aided Design of Discrete Devices (CAD DD'04). Proceedings of the Fifth International Conference, 16-17 November 2004, Minsk, vol. 1, pp. 154-161.

Authors' Information

Arkadij Zakrevskij - *United Institute of Informatics Problems of the NAS of Belarus, Surganov Str. 6, 220012 Minsk, Belarus; e-mail: zakrevskij@tut.by*