

Combinatorial methods of solving Boolean problems

Arkadij Zakrevskij

United Institute of Informatics Problems
Of NAN of Belarus

Contents

- 1 Boolean problems.** Application area. Mathematical base. Combinatorial tasks over Boolean matrices. Constructing efficient means.
- 2 Methods of combinatorial search.** Reduction. Parallelism. Randomization. Recognition.
- 3 Experimental system.** Generating examples. Parametric control. Cuts. Predicting run-time.
- 4 Selected tasks.** Pattern recognition by logical inference. Boolean equation systems. Linear logical equation systems. Decomposition of Boolean functions

Boolean problems - application area

- Testing and verification.
- High-level design of devices for computation and logical control.
- Information security.
- Data mining and pattern recognition.

Examples of tasks

- Search for shortest covers of Boolean matrices.
- Checking the CNF of a Boolean function for **satisfiability** – the SAT problem.
- Finding minimal diagnostic test.
- Minimization of Boolean functions in the class of DNF.
- And many other.

The author entered this field about fifty years ago, after graduating from Tomsk State University.

Boolean problems – mathematical base

Discrete mathematics:

- the theory of finite sets and relations,
- the theory of graphs,
- the theory of algorithms,
- the theory of automata,
- the theory of Boolean functions,
- the theory of predicates,
- the theory of logical inference,
- other branches of the modern algebra of logic.

Combinatorial tasks over Boolean matrices

- Many combinatorial problems of logical design and artificial intelligence can be stated in terms of logical matrices, Boolean and ternary, usually as optimization tasks.
- They are solved by methods of **combinatorial search**, connected with regarding and checking subsets of some large sets. The main efforts should be directed towards essential **reducing** that search.

Examples

- *minimization of disjunctive basis*

- **0 1 0 1 0 0 1** **1 1 0 0 1 0 0** **1 0 1 1**
- **0 0 1 1 0 0 0** **1 0 0 1 1 0 1** **1 0 0 1**
- **1 1 1 1 1 0 0** **0 0 1 1 0 0 0** **0 1 1 1**
- **0 1 1 0 0 0 1**

- *optimization of packing*

- **1 0 0 1 0 0 0** **1 0 0**
- **0 0 1 1 0 0 1** **0 1 0**
- **0 1 0 0 1 1 0** **1 0 0**
- **1 0 0 0 1 0 1** **0 0 1**
- **0 1 0 0 0 1 0** **0 1 0**

Constructing efficient means

- Using computers for solving hard combinatorial problems.
- Development of programming languages and systems oriented towards logical and combinatorial problems – LYaPAS, XBOOLE.
- The chain: mathematical models – formal tasks – methods – algorithms – programs – computer experiments. Regarding this chain as a whole and maximizing its **practical efficiency**.

Methods of combinatorial search

Reducing the search.

- Optimal ordering when using ordered search trees – *solving large system of Boolean equations.*
- Changing the search space – *minimizing weakly specified Boolean functions.*

Parallelism.

- Component-wise operations over long Boolean vectors.
- Decomposing a problem into several subproblems solved in parallel.

Methods of combinatorial search

- Changing a problem for selection of other equivalent problems and organizing a competition – in case of great dispersion of unpredicted run-times.
- About true parallelism and quasi-parallelism.

Randomization.

- The selection is formed by random. The bigger is dispersion of solution costs – the bigger is the gain.

Recognition.

- A good solution can be recognized more quickly, the time for search is cut down.

Experimental system

Generating examples

A generator of a flow of random examples of initial data enables **statistical investigation** of regarded algorithms.

A set of parameters control the generator of examples. For example, the following parameters were used for generating pseudo-random undefined systems of linear logical equations (SLLE):

- 1) m - the number of equations,
- 2) n - the number of variables,
- 3) r - the rank of an existing short solution,
- 4) g - the ordinary number of example.

Experimental system

Technique of cuts

is used in experiments for estimating the efficiency of regarded programs.

A cut presents the results of solving a series of examples defined by fixed values of all the parameters except one.

The values of the latter are changed from one example to another in accordance with three additional parameters:

v - the ordinary number of the selected parameter,

s - the number of steps in the series,

h - the size of a step (the increment of the selected parameter).

Experimental system

Predicting run-time.

The expected run-time of the solving algorithms can be calculated during their **virtual fulfillment**, on the base of evaluation of some statistical parameters of objects.

For example, the following ones can be calculated rather easily for random undefined SLLEs:

mathematical expectation of the number of roots with weight k ,

the upper boundary of the weight of the shortest root.

Examples for demonstration

The following problems are regarded below:

- Pattern recognition by logical inference.
- Solving systems of Boolean equation.
- Solving systems of linear logical equation.
- Decomposition of Boolean functions.

Pattern recognition by logical inference

A set X of attributes (binary or multi-valued)

$X = \{x_1, x_2, \dots, x_n\}$, V_i - the set of values for x_i

The space M over X

$$M = V_1 \times V_2 \times \dots \times V_n$$

The world model (a subject area) $W \subseteq M$

The data (a selection from W) $F \subseteq W$

$$|W| \ll |M|, |F| \ll |W|$$

Pattern recognition by logical inference

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>	<i>h</i>
$F =$	1	0	0	1	1	0	0	1
	0	0	1	1	1	0	0	0
	1	0	0	0	0	1	0	0
	0	1	1	1	0	0	0	1
	1	1	1	0	0	0	0	0

Question: 1 0 ? 1 0

Pattern recognition by logical inference

Using logic for solving recognition problems

Two main stages

Inductive inference

Data mining, search for regularities,
extraction of knowledge

Deductive inference

Using knowledge for recognition

Solving a system of logical equations

The knowledge - concept and format

The difference between data and knowledge

A set of implicative regularities, connecting some attributes = bans on some combinations

The less attributes connected, the stronger is the tie

Selected attributes: a, b, c

Prohibited combination: $a = 1, b = 1, c = 0$

Conjunction form: $a b c' = 0$

Disjunction form: $a' \vee b' \vee c = 1$

A ternary disjunctive knowledge matrix

$$D = \begin{array}{cccccccc} & a & b & c & d & e & f & g & h \\ 1 & - & - & 0 & - & - & 0 & - \\ - & - & - & 1 & - & 1 & - & - \\ 0 & 1 & - & - & - & - & - & - \end{array}$$

The following equation should be satisfied in W :

$$(a \vee d' \vee g') (d \vee f) (a' \vee f) = 1$$

Data mining

The main idea:

finding empty intervals in space M ,
presenting them by conjunctive terms,
putting forward corresponding hypotheses
about regularities,
evaluating their plausibilities,
accepting the plausible enough ones,
filling the knowledge base with them.

Data mining

Evaluating a hypothesis plausibility, the binary case.

$n = |X|$, $m = |F|$, $r =$ the rank of a found empty interval

The probability that such an interval could appear accidentally, is approximated by $E(n, m, r)$ – the expected number of empty intervals of the rank r for a random selection F from M

$$E(n, m, r) = C_n^r 2^{-r} (1 - 2^{-r})^m$$

Data mining

The dependence of E on r under $n=100, m=200$

$r:$	1	2	3	4	5	6
$E:$	10^{-58}	$2 \cdot 10^{-21}$	$3 \cdot 10^{-6}$	156	$4 \cdot 10^6$	$3 \cdot 10^9$

Conclusion:

The search for empty intervals should be restricted in this case by the relation

$$r < 4$$

Data mining

The dependence of r_{\max} on parameters n and m
(for $E < 0.01$)

$n \backslash m$	20	50	100	200	500	1000
10	1	2	3	4	5	6
30	1	2	2	3	4	5
100	1	1	2	3	4	5

Finding syllogistic regularities

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>		<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>
0	1	0	0	1		0	0	-	-	-
1	1	0	1	1		0	-	-	1	-
1	0	0	1	1		1	-	1	-	-
0	1	1	0	0		1	-	-	-	0
1	0	0	1	1	$D =$	-	0	1	-	-
0	1	1	0	0		-	0	-	0	-
						-	0	-	-	0
						-	-	1	1	-
						-	-	1	-	1
						-	-	-	1	0

Deductive inference

The problem: there are given

the knowledge matrix \mathbf{D} , obtained from the data presented by the selection F from W ,

the values of some attributes of a new object

$$W \in W \setminus F.$$

To find the value of some other (goal) attribute

Deductive inference

The complete solution – compressing matrix D

Suppose we know that $a = 1$ and $c = 1$. Then

a	b	c	d	e	f	b	d	e	f	b	d	e	f
0	-	0	-	-	1	-	-	-	1	-	-	-	1
-	0	-	-	1	0	0	-	1	0	0	-	1	-
1	-	-	1	0	-	1	0	-	0	1	0	-	-
-	1	-	0	-	0								
-	1	1	1	-	-								

Hence, $f = 1$. As to the remaining attributes, $b = d = 0$ or $b = e = 1$ are possible.

Solving Large Systems of Boolean Equations

m – the number of equations,

n - the general number of Boolean variables,

k - the number of variables in one equation.

m and *n* could be large (hundreds),

k is **small** (up to ten).

Reduction of large SBE

- Testing pairs of equations for finding *bans*
 - logical conclusions in form $\mathbf{k} = 0$, where \mathbf{k} is an elementary conjunction of $r(\mathbf{k})$ variables.
- Using bans for reducing the number of solutions in other equations.
- Repeating these steps by and by.

Reduction of large SBE

- Three reduction methods:
- *spreading of constants* – when $r(k) = 1$ and some variables receive definite values,
- *the method of syllogisms* – when $r = 2$ and the set of found bans is closed by deductive inference (syllogisms),
- *the method of local reduction* – where r may be arbitrary.

Solving undefined systems of linear logical equations

SLLE – a system of m linear logical equations with n Boolean variables :

$$a_{11}x_1 \oplus a_{12}x_2 \oplus \dots \oplus a_{1n}x_n = y_1 ,$$

$$a_{21}x_1 \oplus a_{22}x_2 \oplus \dots \oplus a_{2n}x_n = y_2 ,$$

...

$$a_{m1}x_1 \oplus a_{m2}x_2 \oplus \dots \oplus a_{mn}x_n = y_m .$$

As a rule, all quantities are known except x_j

.

The problem about the *shortest root*.

11..11..11....1..111	0
1111.11.111111.1.1.1	0
1....1..111..1111.11	0
1....111111.11.111.	1
.1.....11.1...11.1	0
.1...11....1.1.1..1.	0
11.1.11.1..1.111.11.	0
.1.1..11.11.11.11111	1
111..1.11....111...1	1
1.11...11..1.11....1	0
.1.11.1...11.11.1..1	1
.111..111.1111.1111.	1
10000100000000100010	Selected columns

Estimation of the weight γ of the shortest root in a random SLLE

The weight γ is defined by the formula

$$\beta(m, n, \gamma) < 1 \leq \beta(m, n, \gamma + 1),$$

where $\beta(m, n, k)$ is math. expectation of the number of roots with weight not more than k in a random SLLE with parameters m and n :

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

Combinatorial search for roots

- *The trivial algorithm **Simple** :*
- Exhaustive search for column combinations (from n by k), where *search level* k takes values $0, 1, 2, \dots, K$
- The process terminates as soon as the current combination (having K columns) is a root. Evidently, it is the shortest, with weight $w = K$.
- The algorithm is very time-consuming.

Canonization and recognition

- The search is considerably accelerated by **Gauss** method applied to Boolean case.
- By operations of summing rows (by modulo 2) the system is transformed to *canonical form*, with *basis* (m columns) and *remainder* ($n-m$).
- Every combination of columns from the remainder defines a root, they are examined one by one.
- The immediate root is regarded as the solution as soon as its weight w satisfies inequality $w < \gamma$ - the principle of *recognition*.

Solving SLLE by Gauss method

1.....1...1...1..1..	0
.1.....11..	0
..1.....	0
.....11.....1.1..1	0
....1.....1...1.1..1	1
.....1.....11..1.	0
...1..1.....111.11	0
.....1.1...111.11	0
.....1..1.111	1
.....11...11.1..	1
.....1...111.	1
.....1111..1	1
.....*...*...*****	<i>remainder</i>
1.....1.....1...1.	<i>solution</i>

Randomized parallelism

Algorithm **RandoPar** q :

- q different random linearly independent subsets of columns are selected.
- q canonical forms are constructed.
- A short solution is looked for in parallel over all forms sequentially on search levels 0, 1, 2,...
- The search terminates as soon as the weight w of immediate root satisfies relation $w < \gamma$.

Efficiency of different methods

Search of the shortest root with weight 75 in SLLE
with parameters $n = 500$ and $m = 420$

Algorithm

*The run time (1000 MH),
averaged over 20 examples*

Simple

~ 1078 years

Gauss

5 000 years

RandoPar 10

5 months

RandoPar 250

1 hour

Solving inconsistent (overdefined) SLLE

When $m > n$ no root exist usually. So the problem is reformulated.

To solve an inconsistent SLLE $\mathbf{A} \mathbf{x} = \mathbf{y}$ with given \mathbf{A} and \mathbf{y} means to find such a value of vector \mathbf{x} , which satisfies the maximum number of equations.

In other words, to find a shortest *correction vector* \mathbf{c} transforming \mathbf{y} to a nearest to it *suitable vector* \mathbf{y}^* , for which the system becomes consistent:

$$\mathbf{y}^* = \mathbf{y} \oplus \mathbf{c}.$$

Solving inconsistent (overdefined) SLLE

Solving of *inconsistent* SLLE is *combinatorially reduced* to the previous problem – to finding a shortest solution of the corresponding *undefined* SLLE.

Sequential two-block decomposition of Boolean function

$$f(\mathbf{x}) = f(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$$

disjunctive

$$f(\mathbf{x}) = g(h(\mathbf{u}), \mathbf{v})$$

non-disjunctive

$$f(\mathbf{x}) = g(h(\mathbf{u}, \mathbf{w}), \mathbf{w}, \mathbf{v})$$

Strong partition \mathbf{u}/\mathbf{v}

$$\mathbf{u} \cup \mathbf{v} = \mathbf{x}$$

$$(x_2, x_3, x_4, x_7) / (x_1, x_5, x_6)$$

Weak partition \mathbf{u}/\mathbf{v}

$$\mathbf{u} \cup \mathbf{v} \subseteq \mathbf{x}$$

$$(x_2, x_4, x_7) / (x_1, x_5)$$

$$\mathbf{w} = (x_3, x_6)$$

Two main tasks

Note. Only non-trivial compositions are regarded below, for which $|u| > 1$ and $|v| > 0$.

The task 1. For a given function $f(x)$ and partition u/v (strong either weak) to find out, whether $f(x)$ is decomposable at u/v .

If such a composition does exist, we shall term partition u/v as *appropriate*.

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

The second task is more difficult. It is regarded below.

Correct reformulation of Task 2

The probability $p(n)$ of decomposability of *random* Boolean function of n variables is very low.

n	=	5	6	7	8	9
$p(n)$	=	0.105	0.0029	$2.49 \cdot 10^{-7}$	$1.36 \cdot 10^{-16}$	$8.03 \cdot 10^{-33}$

So, *random* Boolean functions of n variables are *practically non-decomposable*, if $n > 6$.

The problem should be reformulated:

A Boolean function $f(\mathbf{x})$ is given, obtained by an unknown composition $g(h(\mathbf{u}, \mathbf{w}), \mathbf{w}, \mathbf{v})$. To find the weak partition \mathbf{u}/\mathbf{v} .

That is the problem of *recognition*.

Relation of submission on the set of partitions

A partition $\mathbf{u}^*/\mathbf{v}^*$ is a *trace* of partition \mathbf{u}/\mathbf{v} , i. e. *submits to* \mathbf{u}/\mathbf{v} , if

$$\mathbf{u}^* \subseteq \mathbf{u} \text{ и } \mathbf{v}^* \subseteq \mathbf{v}.$$

Assertion. If function $f(\mathbf{x})$ is decomposable at partition \mathbf{u}/\mathbf{v} , then it is decomposable at submitted partition $\mathbf{u}^*/\mathbf{v}^*$.

The simplest from non-trivial partitions have $k = |\mathbf{u}| = 2$ and $m = |\mathbf{v}| = 1$. They are called *triads*.

A triad is a trace of partition \mathbf{u}/\mathbf{v} if it submits to \mathbf{u}/\mathbf{v} .

Assertion. Function $f(\mathbf{x})$ is not decomposable, if it is not decomposable at any of triads.

Solving task 2

The suggested heuristic algorithm for finding existing partition u/v is based on the following

Statistical conclusion. If $n > 9$, practically every appropriate triad u^*/v^* a trace of the partition u/v . The latter could be found by successive expansion of sets u^* and v^* .

Algorithm

1. By random checking triads one by one, to find an appropriate triad u^*/v^* .
2. Supposing that u^*/v^* is a trace of the sought-for partition u/v , to expand it by adding new elements in its left and right parts and checking the results.

To the run-time estimation

The expectation of the number q of checked triads before finding a trace of partition \mathbf{u}/\mathbf{v} is

$$M(q) = C_n^2(n-2) / C_k^2 m.$$

Approximation:

$$\text{if } k = m = n/t \quad M(q) = t^3 ,$$

$$\text{if } k = m = n/2 \quad M(q) = 2^3 = 8 ,$$

$$\text{if } k = m = n/3 \quad M(q) = 3^3 = 27 .$$

Note. It does not depend on n .

Experiments (C++, 2.8 GHz)

Preparing an example:

- parameters n, k, m are fixed;
- random partition \mathbf{u}/\mathbf{v} on set \mathbf{x} and two random Boolean functions g and h (of $m+1$ and k variables) are generated;
- composition $f(\mathbf{x}) = g(h(\mathbf{u}), \mathbf{v})$ is calculated.

Finding the solution:

- the partition \mathbf{u}/\mathbf{v} is found by the given algorithm,
- the number q of triads checked when looking for a track is defined,
- the whole time t spent for finding \mathbf{u}/\mathbf{v} is measured (in seconds).

Results of experiments

n	k/m	q	t	k/m	q	t
14	7/7	3	0.00	5/5	30	0.00
15	8/7	2	0.00	5/5	78	0.02
16	8/8	1	0.00	6/5	18	0.01
17	9/8	11	0.02	6/6	34	0.05
18	9/9	4	0.02	6/6	42	0.09
19	10/9	3	0.03	7/6	39	0.17
20	10/10	9	0.11	7/7	3	0.16
21	11/10	1	0.11	7/7	3	0.39
22	11/11	3	0.48	8/7	6	2.41
23	12/11	9	2.17	8/8	16	7.05
24	12/12	3	2.48	8/8	26	18.17
25	13/12	4	5.61	9/8	19	33.16
26	13/13	4	11.64	9/9	3	52.11
27	14/13	19	60.13	9/9	36	187.55
28	14/14	19	1280.67	10/9	3	1585.03