
Minimization of Boolean Functions of Many Variables – Iterative Algorithm

Arkadij Zakrevskij, Nikolaj Toropov

United Institute of Informatics Problems of the National Academy
of Sciences, Minsk, Belarus

e-mail: zakrevskij@tut.by toropov@newman.bas-net.by

1. The task formulation

To find a rather good DNF (disjunctive normal form) for an arbitrary Boolean functions of many variables ($20 < n < 30$). Here the number of conjunctions in the *perfect* DNF amounts to many millions. The *resulting* DNF with minimal number of conjunctions (a shortest DNF) is considered as an optimal one.

A practically efficient heuristic method for solving this task is suggested.

The initial information is given by a Boolean vector \mathbf{f} with 2^n components presenting an arbitrary Boolean function f of n variables.

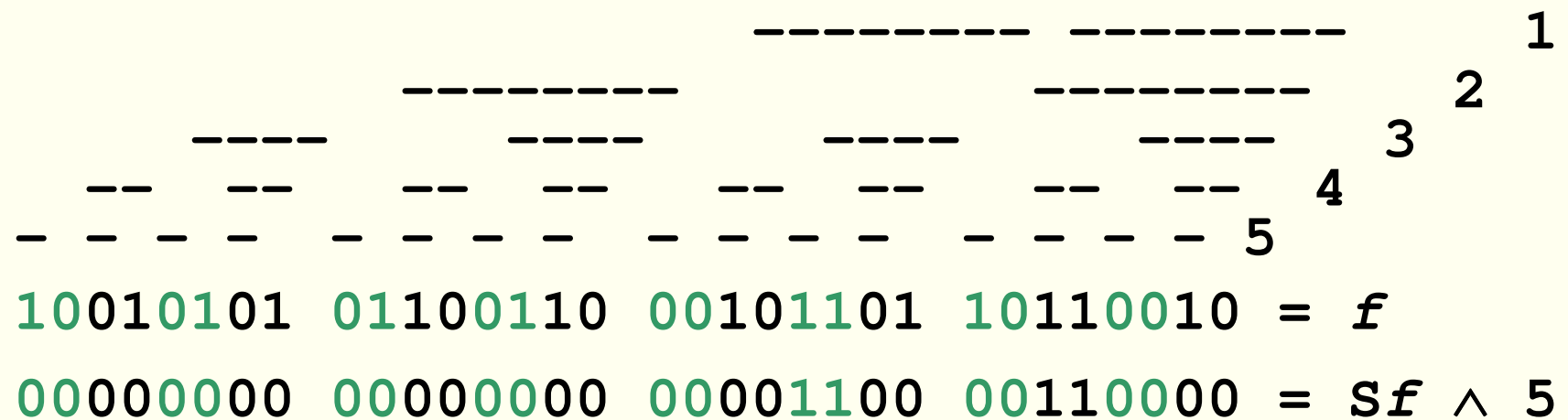
Note, that at $n = 25$ the length of notation such a vector exceeds 130 kilometers. The known classical methods of Boolean function minimization cannot deal with such functions.

3. Using efficient operations over adjacent elements of the Boolean space

Boolean n -vectors representing different elements of the Boolean space are **neighbour** if they differ by their values only in one component.

A basic set of efficient parallel operations executed **simultaneously** over 2^{n-1} pairs of elements of the n -dimensional Boolean space, adjacent by some variable x_i ($i = 1, 2, \dots, n$) was developed.

One of such operations is used in the method: the **operation of symmetrization** $Sf \wedge k$, that provides both neighbour by variable x_k elements in each couple gain an identical value.



4. Calculation of the matrix of neighborhood

The operation of symmetrization is used for finding all elements of the characteristic set M^1 which have neighbor regarding the variable x_i :

$$S\mathbf{f} \wedge i$$

In its turn, that enables to get the matrix of neighborhood \mathbf{N} , where $n_i^k = 1$ iff both the element m_k of the Boolean space M and its neighbor by the variable x_i belong to the characteristic set M^1 .

$$\begin{array}{r}
 \mathbf{f} = 10010101 \quad 01100110 \quad 00101101 \quad 10110010 \\
 S\mathbf{f} \wedge 1 = 00000101 \quad 00100010 \quad 00000101 \quad 00100010 \\
 S\mathbf{f} \wedge 2 = 00000100 \quad 00000100 \quad 00100000 \quad 00100000 \\
 S\mathbf{f} \wedge 3 = 00010001 \quad 01100110 \quad 00000000 \quad 00100010 \\
 S\mathbf{f} \wedge 4 = 00000101 \quad 00000000 \quad 00000101 \quad 10100000 \\
 S\mathbf{f} \wedge 5 = 00000000 \quad 00000000 \quad 00001100 \quad 00110000
 \end{array}
 \left. \vphantom{\begin{array}{r} \mathbf{f} \\ S\mathbf{f} \wedge 1 \\ S\mathbf{f} \wedge 2 \\ S\mathbf{f} \wedge 3 \\ S\mathbf{f} \wedge 4 \\ S\mathbf{f} \wedge 5 \end{array}} \right\} = \mathbf{N}$$

The columns of matrix \mathbf{N} corresponding to elements of vector \mathbf{f} show by which variables the regarded elements of the characteristic set M^1 have neighbors.

5. Selecting elements with minor number of neighbors

$$\begin{aligned}
 \mathcal{F} &= 10010101 \ 01100110 \ 00101101 \ 10110010 \\
 &\quad 00000101 \ 00100010 \ 00000101 \ 00100010 \\
 &\quad 00000100 \ 00000100 \ 00100000 \ 00100000 \\
 \mathcal{N} &= 00010001 \ 01100110 \ 00000000 \ 00100010 \\
 &\quad 00000101 \ 00000000 \ 00000101 \ 10100000 \\
 &\quad 00000000 \ 00000000 \ 00001100 \ 00110000
 \end{aligned}$$

Estimating the numbers of neighbors for elements of the characteristic set M^1

$$00010303 \ 01200220 \ 00101302 \ 10510020$$

Finding the subsets of elements with minor number of neighbors (0, 1, 2 and 3) and presenting them by the following Boolean vectors:

$$\left. \begin{aligned}
 m_0 &= 10000000 \ 00000000 \ 00000000 \ 00000000 \\
 m_1 &= 00010000 \ 01000000 \ 00101000 \ 10010000 \\
 m_2 &= 00000000 \ 00100110 \ 00000001 \ 00000010 \\
 m_3 &= 00000101 \ 00000000 \ 00000100 \ 00000000
 \end{aligned} \right\} = M$$

6. Presentation of the sought DNF

The required DNF will be presented by a pair: Boolean solution vector g and matrix D obtained from f and N by deleting some ones.

For the same Boolean vector

$f = 10010101 \ 01100110 \ 00101101 \ 10110010$ – initial perfect DNF

$00000101 \ 00100010 \ 00000101 \ 00100010 \ S f \wedge 1$

$00000100 \ 00000100 \ 00100000 \ 00100000 \ S f \wedge 2$

$N = 00010001 \ 01100110 \ 00000000 \ 00100010 \ S f \wedge 3$

$00000101 \ 00000000 \ 00000101 \ 10100000 \ S f \wedge 4$

$00000000 \ 00000000 \ 00001100 \ 00110000 \ S f \wedge 5$

$g = 10010000 \ 01100000 \ 00101001 \ 10010000$ – found DNF

$00000000 \ 00100000 \ 00000001 \ 00000000 \ x_1$

$00000000 \ 00000000 \ 00100000 \ 00000000 \ x_2$

$D = 00010000 \ 01100000 \ 00000000 \ 00000000 \ x_3$

$00000000 \ 00000000 \ 00000001 \ 10000000 \ x_4$

$00000000 \ 00000000 \ 00001000 \ 00010000 \ x_5$

$x_1 \ 0 \ 0 \ 0 \ - \ 1 \ 1 \ - \ 1 \ 1$

$x_2 \ 0 \ 0 \ 1 \ 1 \ - \ 0 \ 0 \ 1 \ 1$

$x_3 \ 0 \ - \ - \ - \ 0 \ 1 \ 1 \ 0 \ 0$

$x_4 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ - \ - \ 1$

$x_5 \ 0 \ 1 \ 1 \ 0 \ 0 \ - \ 1 \ 0 \ -$

Matrix form the found DNF

7. Looking for obligatory prime implicants

The construction of a DNF implementing the given function f begins with the search for **obligatory prime implicants** of higher ranks, which belong to anyone shortest DNF of this function.

The search is facilitated by the described sorting of units of characteristic set M^1 by the number of their neighbors.

Let Int_i represented by the ternary vector t^i be the minimal interval of the Boolean space M , covering some element m_i together with all its neighbors. Then vector t^i represents an **obligatory prime implicant** of the f if and only if $Int_i \subseteq M^1$.

The probability of satisfying that condition depends on the number k of neighbors of an element m_i . It equals to 1 for $k = 0$ or $k = 1$ and then quickly decreases, obtaining values $1/2$, $1/16$, $1/2048$, $1/67\ 208\ 864$ for $k = 2, 3, 4, 5$, correspondingly, for completely random Boolean function.

That is why only elements enumerated in vectors m_0 , m_1 , m_2 and m_3 are checked during the search for obligatory prime implicants. Vectors t^i are found for them rather easily, and then checked for meeting condition $Int_i \subseteq M^1$.

8. Continuation of the algorithm. Heuristics

When a regarded element m_i does not satisfy the condition $Int_i \subseteq M^1$, it is covered by some interval which covers not all neighbors of m_i but only some of uncovered elements of M^1 . This operation is **heuristic**.

1's in vector \mathbf{f} that mark already covered elements of N^1 are changed for zeros, and as a result vector \mathbf{f} is changed for vector \mathbf{f}^* .

It is possible that even after this procedure \mathbf{f}^* has 1's. Then the described procedure is **iterated**. New values of matrices \mathbf{N} and \mathbf{M} are calculated for vector \mathbf{f}^* each time and used in company with vector \mathbf{f} for finding additional implicants of the function f , which should be properly corrected to make them prime and added to the solution. If necessary, this iteration could be repeated many times.

9. A program implementation of the algorithm

The described algorithm was programmed in C++ and tested on computer (Pentium IV, 2.8 GHz).

A generator of a pseudo-random Boolean vector \mathbf{f} was developed. A notion of **density** of values 1 r was introduced, that defines the relative quantity of ones in vector \mathbf{f} .

The program is **iterative**:

1. At first only such columns of the matrix \mathbf{N} of neighbourhood are processed, each of which contains no 1s, one 1, two 1s or three 1s. Obligatory and some other prime implicants are found.
2. If $\mathbf{f}^* \neq \mathbf{0}$, the second iteration is fulfilled: a new matrix \mathbf{N} is constructed presenting relation of neighborhood between the elements enumerated in vector \mathbf{f}^* , its columns containing not more than three 1s are regarded and some new implicants are found in the same way, vectors \mathbf{g} and \mathbf{f}^* take new values.
3. If after that again $\mathbf{f}^* \neq \mathbf{0}$, the third iteration is fulfilled, etc.
4. Not all of found implicants are prime, hence two additional procedures are applied to them to make the solution good enough: making them prime and eliminating doubles.

10. Example

Let $n = 19$ and $r = 1/4$. A random vector f is generated with 147 232 ones and a matrix of neighborhood N with $2^{19} = 534\ 288$ columns is constructed, characterized by the following distribution of numbers $N(i)$ of columns having i 1s.

$i =$	0	1	2	3	4	5	6	7										
$N(i) =$	296	2087	7180	16352	25357	29868	26913	19406										
$i =$	8	9	10	11	12	13	14	15	16	17	18	19						
$N(i) =$	11379	5401	2087	690	172	35	8	1	0	0	0	0						

The following quantities of obligatory prime implicants were found for

$i = 0, 1, 2, 3$: 296, 2082, 1974, 80. Together with non-obligatory members, 24 379 prime implicants were found at the first iteration. The number of 1s in vector f^* now equals 81 910. The next iteration is fulfilled, because $f^* \neq \mathbf{0}$.

In such a way four iterations are fulfilled before vector f^* becomes equal to $\mathbf{0}$ (in 10.5 seconds). 61 477 implicants are found by that, covering all 1-elements of vector f , with the following distribution by ranks:

19 – 2 458, 18 – 33 668, 17 – 25 237, 16 – 114.

Not all of these implicants are prime, than one procedure eliminates, if possible, some letters from regarded implicants and makes them prime. The new distribution:

19 – 296, 18 – 20 933, 17 – 38 617, 16 – 1 631.

The other procedure compares implicants by pairs and removes doubling. As a result, 60 972 prime implicants of 4 different ranks are found (in 80 sec), with the following distribution:

19 – 296, 18 – 20 933, 17 – 38 353, 16 – 1 390.

11. Some results of computer experiments

n	r	N	C	It	T
7	1/2	62	31	1	0.00
8	1/2	138	58	2	0.00
9	1/2	274	107	3	0.00
10	1/2	556	194	3	0.00
11	1/2	1083	379	4	0.01
12	1/2	2203	735	5	0.03
13	1/2	4337	1414	7	0.09
14	1/2	8734	2780	12	0.35
15	15/32	16404	5313	13	1.01

n – the number of variables,

r – the density of values 1,

C – the number of conjunctions in the obtained DNF,

It – the number of iterations,

n	r	N	C	It	T
16	14/32	31021	10181	13	3.12
17	14/32	61150	19811	90	24.37
18	13/32	114347	38007	21	42.26
19	12/32	212620	72343	12	148
20	11/32	392995	137215	10	610
21	11/32	786345	270642	18	2499
22	10/32	442274	512529	10	8858
23	9/32	2620069	966357	8	31064

N – the number of values 1,

T – the time spent in seconds.

12. Conclusion

A new practically efficient algorithm is developed and implemented on computer.

It minimizes arbitrary Boolean functions of many variables (up to 24) in acceptable time.

Thank you for your attention