Chapter 2 Solving Large Systems of Boolean Equations

Arkadij Zakrevskij
National Academy of Science, Belarus

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, et cetera. 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 chapter, as well as the information concerning their program implementation and experimental estimation of their efficiency.

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 k 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 $\varphi_i(\boldsymbol{x})$ of 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*

DOI: 10.4018/978-1-4666-1900-5.ch002

arguments, where m is the number of equations and n is the total number 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:

$$\boldsymbol{V} = \begin{array}{ccccc} 0010 & 0100 & 0100 & 0000 & v_1 \\ 1100 & 0000 & 1001 & 0110 & v_2 \\ 0000 & 0010 & 1001 & 0010 & v_3 \end{array}$$

$$\boldsymbol{W} = \begin{array}{ccccccccc} a & b & c & d & e & f & g & h & \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & w_1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & w_2 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & w_3 \end{array}$$

Let us name these systems as *large SLEs*. 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 SLEs is presented in this survey, together with the results of their software implementation. They were published in various papers (see, please, *References*).

SEARCH TREE MINIMIZATION

Two combinatorial methods using tree searching technique could be applied to solve large SLEs: the *equation scanning method* and the *argument scanning method*. The first method is implementing consecutive multiplication of orthogonal DNFs of the equations from the considered system and uses the search tree T_e the levels of which 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 accelerated by the described below methods taking into account only the matrix of arguments $\boldsymbol{W}$ (Zakrevskij A. & Zakrevski L., 2002; Zakrevskij & Vasilkova, 2002).

Raising Efficiency of the Equation Scanning Method

In that method, the search tree T_e is regarded i-th level of which corresponds to some equation $\varphi_i(\boldsymbol{u}_i)$ and its nodes represent the roots of the subsystem constructed of 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 be expanded 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