Solving of logic functions systems using genetic algorithm

V G Kurbanov[1,2] **and M V Burakov**[1]

[1] Chair of control system of Saint-Petersburg State University of Aerospace Instrumentation, Bolshaya Morskaya, 67, St. Petersburg, Russia
[2] Institute of Problems in Mechanical Engineering, Russian Academy of Sciences, 61 Bol'shoi pr., Petersburg, Russia

Abstract. The problem of solving a system of logical functions that occurs in the analysis and synthesis of intelligent control systems for nonlinear dynamic plants is considered. The arithmetic of logical functions is proposed by using the Zhegalkin basis and the evolutionary solution of the system of logical functions by means of a genetic algorithm. The given examples show that the offered algorithm can be successfully used in practice.

1. Introduction

When analyzing and synthesizing intelligent control systems for nonlinear dynamic plant that function in unsteady, extreme conditions with given quality indicators, one of the most important tasks is the problem of logical inference and decision making for a system of logical functions written in symbolic form. The solution of such a problem in a symbolic form encounters fundamental difficulties and only in the simplest cases is achieved by methods of directed search. In more complex cases, the main methods for solving it are heuristic methods, in which intuition and knowledge in solving similar problems and previous practical experience play a big role.

This knowledge in expert systems is used by the so-called logical inference machine, i.e. a program that operates with a knowledge base and draws conclusions. One of the variants of the inference engine is the system of production rules (product system), which is most often used in expert systems [1].

The work of the production system can be considered as the following sequence of operations: finding all the rules, the conditions of which are met; the choice of one of these rules and the implementation of the actions prescribed by it. The disadvantages of heuristic methods of obtaining solutions in logical inference systems are: the lack of guarantee of a solution for a finite number of steps in the algorithm, the randomness in choosing the most rational way of finding a solution, the low speed of the number of logical inferences per unit time, the excessive complexity of interpreting the structural properties of the solver. Therefore, interest in the problems of equivalent symbolic transformations of logical functions (LF) and the systems of logical equations (SLE) that they form is steadily growing, they underlie the solution of such problems of artificial intelligence as logical inference and decision making [2], regardless of to which application area the data decisions apply.
In this paper, we propose a version of the description and solving of the SLE

2. Arithmetization of systems of logical equation

The symbol representation of LF and SLE for large dimensions of logical problems is cumbersome and inconvenient, therefore it is actual to search for other forms of representation of LF and SLE, for example, those in which their arithmetic properties are used.

There are methods when for an ordered set a Cartesian product is constructed, which elements are lexicographically ordered and there is no need to write explicitly all its terms, but it is sufficient to know how any one of them is calculated. In this case, due to the arithmetic properties of LF and SLE, which they display when they are presented in the form of algebraic structures according to mod2, that is, in the algebra of Zhegalkin [3], it turns out possible to reduce logical problems to arithmetic or similar arithmetic ones. In general, this allows us to represent logical systems as linear structures which equations do not contain conjunctive elements, and to analyze and synthesize their structural properties, use the mathematical apparatus of vector-matrix algebra.

Then for such systems, the logical inference is represented as a procedure for inversion of [0, 1] - matrices, and the decision making is a repetition of this procedure after changing the initial conditions.

The *basic vector of a logical system* is an ordered set of logical variables that cannot be decomposed into simpler ones, that is, they cannot be produced by other variables:

$$\mathbf{x}^{\mathrm{T}} = < x_1, x_2, \ldots, x_n >,$$

where symbol $^{\mathrm{T}}$ means transposing of vector **x**.

The *fundamental vector of a logical system* is the ordered set of elements of the Cartesian product of the basis vector **x**, augmented by **1** in place of the last element:

$$\mathbf{S}^{\mathrm{T}} = <x_1, x_2, ..., x_n, x_1x_2, x_1x_3, ..., x_1x_2x_3, ..., x_1x_2 \cdot, ..., \cdot x_n, 1>.$$

The dimension of this vector is $n_{\mathbf{s}} = 2^n$.

The *identification line of the LF* is a line consisting of zeros and ones of the form: $\mathbf{C}_i$ = <1, 1, 0, 1, 1, 0, 1, ..., 1>. Units in this line identify the elements of the vector S, which in the *i*-type LF add up according to **mod2**. If the last element of the identification line is 1, then it means the negation of f_i. This form of the LF task (representation) allows you not to enter additional identifiers, for example, strikethroughs, to denote the logical connective of negation NOT.

The idea of the arithmetic symbolic logics belongs to I.Zhegalkin; based on J. Boole's algebra, he simplified the laws of operating with logical addition and multiplication and reduced these operations to actions that are subject to the arithmetic laws of associativity, commutativity and distributivity. In this case, according to I. Zhegalkin, the logical connection "or" can be used only in a strictly separative sense, as the connection "either." Obviously, this restriction puts forward an additional requirement to the process of forming knowledge bases of the subject domain.

One of the most important properties of the algebra of logics introduced by I. Zhegalkin is that in it the logical content of the symbols "1" and "0" corresponds always to the true or false function, and the negation operation is replaced by the operation of adding one to the logical variable:

$$\bar{x} = x \otimes 1.$$

A distinctive feature of the LF representation in the Zhegalkin (ZhB) Basis is that when performing their transformations, LFs can be considered as algebraic expressions in symbolic form, and manipulation operations with them are algebra operations. Such operations are: addition, multiplication, transfer of expressions from the left to the right, and vice versa, the compilation and solution of systems of equations, including matrix methods, substitution of some functions into others, reduction of similar terms. And a t the same time, they do not contradict the axioms of ZhB.

Thus, when solving systems of logical equations, their arithmetization is easily accomplished by transferring the LF from the Boolean Basis to the ZhB, and when interpreting the solutions obtained, it is back to the Boolean Basis. Therefore, the solution and analysis of logical problems are reduced to the analysis and solution of the system:

$$\mathbf{AS} = \mathbf{b} \qquad (1)$$

where **A** is a rectangular binary matrix of dimension [*n, m*], *n*> *m*; **S** is the fundamental vector of a logical system of dimension *n*; **b** is a binary vector of dimension *n*.

The main task is to find a matrix that is pseudo-inverse to matrix **A**. The solution of system (1), in which the variables are greater than the equations, leads to a set of solutions. Therefore, the pseudo-inverse matrix is not unique. Their complete set defines all admissible solution vectors.

3. Solution of the system of logical equations

One of the methods of solution is a method analogous to the method of elimination of K. Gauss when solving linear systems of algebraic equations with real numbers. The essence of the method consists in adding equations to exclude the variables entering into these equations. The effect of absorption of the terms of the equations as a result of their addition according to mod 2 is connected with the fulfillment of one of the axioms of Zhegalkin's algebra:

$$x \oplus x = 0.$$

The procedure consists in transforming the matrix **A** so that as a result of adding rows of the matrix according to the rules of elementary transformations according to mod 2, the resulting matrix would have a minimal number of non-zero elements. This is the solution.

If by the condition of the problem it is required to find among the possible admissible solutions one that is the best, then its solution, as a rule, is connected with optimization problems. Optimization tasks are possible only when there is some measure for the vector **S**. This measure is hidden in the attributes of **S**, since the vector itself is logical. Therefore, it is necessary to introduce a definition of its measure (norm), which requires the metrizability of a set of attributes, which unfortunately, is not always easy, and sometimes impossible to get. The optimization problem is most easily solved if the indicated measure has a scalar type.

In the simplest case, the scalar measure, for example, can be the Euclidean norm, that is, the sum of the non-zero elements of the vector **S** which weight is conditionally taken to be unity. Here we can see an analogy with the code distance in the R. Hemming theory. Finding the minimum of this sum is a task of combinatorial optimization. It is known that minimization of the Euclidean norm of a linear system with real numbers

$$\mathbf{A}_a\mathbf{S}_a = \mathbf{b}_a$$

is approachable if the following condition is met:

$$\underset{\min}{\mathbf{S}_\mathbf{a}} = \left(\mathbf{A}_\mathbf{a}^\mathbf{T}\mathbf{A}_\mathbf{a}\right)^{-1}\mathbf{A}_\mathbf{a}^\mathbf{T}\mathbf{b}_\mathbf{a}. \quad (2)$$

A solution is similar in which S contains the minimal number of non-zero components, that is, if A ~ is a pseudo-inverse matrix, then the solution of equations (1) has the form:

$$\mathbf{A}^{\sim}\mathbf{A}\mathbf{S} = \mathbf{A}^{\sim}\mathbf{b}.$$

If $\mathbf{A}^{\sim}\mathbf{A} = \mathbf{E}$ – a single matrix then

$$\mathbf{S} = \mathbf{A}^{\sim}\mathbf{b}.$$

In system (1), matrices and vectors have a binary data type, and in expression (2), the elements are assigned a real type, and the optimization problem goes to the attribute area. If, by the condition of the problem, only admissible solutions are required, that is, the entire set of pseudo-inverse matrices, then the solution of the optimization problem is not required. However, the most studied and often encountered optimization problem is finding an admissible solution with the maximum value of its probability, which is typical for fuzzy problems of making a decision of the logical-probabilistic type.

Suppose that the following rules for working with data are given, for example from a sensor of some dynamic plant:

$$\text{If } (x_{ij} = 1 \wedge y_{ij} = 1 \wedge z_{ij} = 1 \wedge u_{ij} = 1 \wedge v_{ij} = 1 \wedge w_{ij} = 1), \text{ then } q_{ij} = 1 \quad (3)$$

The rules of the form (3) are implicit in the language of the algebra of logic or Boolean algebra:

$$x_{ij} \wedge y_{ij} \wedge z_{ij} \wedge u_{ij} \wedge v_{ij} \wedge w_{ij} \rightarrow q_{ij} \quad (4)$$

The expressions of the form (4) can be transformed into the form of the Zhegalkin algebra or into equivalent algebraic equations according to mod 2 [2].

$$s_{ij} \oplus s_{ij} * q_{ij} \oplus 1 = b_{ij},$$

where: $\oplus$ - a sign of addition according to mod 2, $*$ - a sign of multiplication according to mod 2, b_{ij} – 0 or 1 ($b_{ij} \in (0,1)$ and

$$s_{ij} = x_{ij} * y_{ij} * z_{ij} * u_{ij} * v_{ij} * w_{ij}.$$

Then the resulting system of logical equations can be written in the matrix form according to mod 2 [3, 4]:

$$\mathbf{AF = B} \quad (5),$$

where: **B** is a binary vector of dimension n, **F** is a fundamental vector of a logical system of dimension n, constructed from combinations of logical variables x_{ii}, y_{ii}, z_{ii}, u_{ii}, v_{ii}, w_{ii}, q_{ij}, supplemented by 1 in place of the last element, **A** is a rectangular binary matrix of dimension [n, m].

The procedure for obtaining the system of equations (5) is easy to formalize. For this, we can first construct a fundamental vector **F** of the system:

$$\mathbf{F}^{\mathbf{T}} = [s_{11}, s_{12}, \ldots, s_{1n}, s_{21}, \ldots, s_{mn}, s_{11}*s_{12}, \ldots, s_{11}*s_{12}*\ldots*s_{mn}, s_{11}*q_{11}, \ldots, s_{11}*s_{12}*\ldots*s_{mn}*q_{11}*\ldots*q_{mn}, 1]$$

The following algorithm is used:

- all the logical variables of sensory data are listed;
- after, all combinations of two of the logical variables of sensory data are listed;
- then, all combinations of three of the logical variables of the sensory data are listed;
- then, all combinations of four of the logical variables of the sensory data are listed,
- and so on, the product of all the logical variables of the sensory data is placed at the end.

After, the fundamental matrix A of the system is constructed:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 1 \\ 1 & 1 & 0 & \ldots & 0 \\ 1 & 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 & 1 \\ 1 & 1 & 1 & \ldots & 0 \\ 1 & 1 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 & 0 \\ 1 & 1 & 1 & \ldots & 1 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The algorithm for constructing the matrix **A** is as follows:

- the first line in the first column is set to 1, and in the remaining 0
- in the second row, the second column is set to 1, and in the remaining 0
- and so on, until there is 1 in the last column,
- then put 1 in the first two columns, and in the remaining 0,
- - then put 1 in the first and third columns, and in the remaining 0,
- - then put 1 in the first and fourth columns, and in the remaining 0,
- - and so on, until there are two in the last two columns,
- - then put 1 in the first three columns, and in the remaining 0,
- - then put 1 in the first, third and fourth columns, and in the remaining 0,
- - then put 1 in the first, fourth and fifth columns, and in the remaining 0,
- - and so on, until there are three in the last three columns,
- - and so on, until there are 1 in all columns

Naturally, the matrix system of equations according to mod 2 of the form (4) obtained in this way will have a large dimension. However, in real problems, not all components of this equation (not all combinations of logical variables) are physically realizable and can be discarded. As a result of this reduction, we obtain a matrix system of equations according to mod 2 of smaller dimension [3]:

$$\mathbf{CR = G.}$$

Thus, an optimization problem arises, in which it is necessary to choose the most acceptable or suboptimal solution from a variety of solutions. This problem can be attributed to the class of global optimization problems, for which currently popular methods are widely used, such as the genetic algorithm implemented in the Optimtool Simulink MatLab package [5].

4. Using the genetic algorithm

The genetic algorithm (GA) is a subject-independent mechanism of global optimization, modeling the basic principles of evolution [6]. For example, in [7, 8] are considered the questions of genetic training of nonlinear PID regulators.

To use GA in a specific task, it is necessary to solve two questions:

- Describe the fitness function *F*, which will characterize each solution of the problem. The optimization problem is considered as the minimization of *F*.
- Select the method for coding the solution, i.e. describe the chromosomes which population will be modified by the action of genetic operators.

In the problem under consideration, the chromosome is a bit line of length *n* equal to the number of logical equations. Elements of the line are interpreted as logical variables.

The fitness function can be described as follows:

$$F = \begin{cases} Fb, & if \quad \mathbf{CR} \neq \mathbf{G}, \\ \dfrac{1}{1+\sum\limits_{i=1}^{n} r_i}, & if \quad \mathbf{CR} = \mathbf{G}. \end{cases} \tag{6}$$

where $\mathbf{R} = [r_1, r_2, r_3, \ldots, r_n]^T$.

The value of *Fb* in (6) is a large positive constant, which serves to reject false decisions.

Thus, when the function (6) is used, the solutions with the greatest number of non-zero components have the best suitability.

5. Modeling example

Consider a system of 5 logical equations from 8 variables:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

To find the solution, the following *MatLab*-function is used:

```
function z=kurbanov1(X)
A=[false false true false false true true false;
    true true false true false false true true;
    false true false false true false false false;
    false true true false false true true false;
    true false true false true false false true];

B = [true false true true false];

for i = 1:5
P(i)=and(A(i,1),X(1));
 for j =2:8
   P(i)=xor(P(i),and(A(i,j),X(j)));
 end;
end;
S=0;
for i=1:5
 if (B(i))==(P(i)) S=S+1; end;
end;
 if S~=5 z=200; end;
 if S==5
    z=0;
    for i=1:8
     if X(i)~=0 z=z+1; end;
    end;
    z=1/(1+z);
 end;
end
```

Table 1 shows the state of the population at the second iteration.

Table 1. Status of the population

chromosome									Fitness function
1	1	0	1	1	1	1	1	1	0,125
2	1	0	0	1	1	1	0	0	0,2
…	…								…
41	1	0	1	1	1	1	1	1	0.125
42	1	1	0	0	1	0	1	0	200
43	0	0	0	0	1	0	1	1	0,25
44	0	0	1	1	1	1	1	0	0,166
…	…								…
100	0	0	1	0	1	0	0	0	0.33

It takes only a few dozen iterations to find a solution by used Optimization Toolbox Graphical User Interface MatLab (Figure 1).

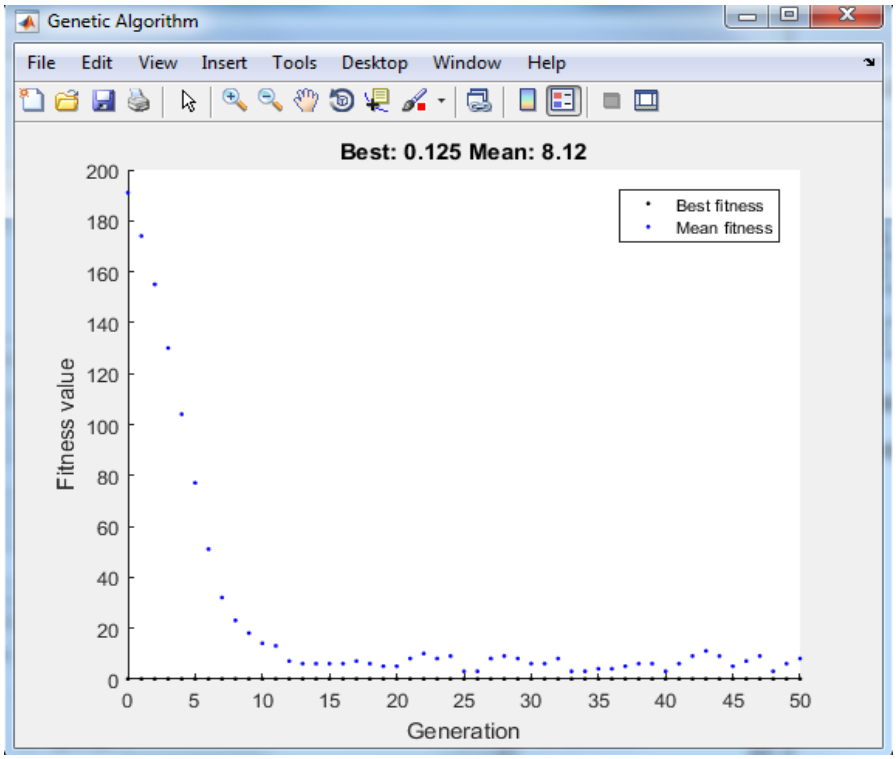

Figure 1. Illustration of GA work

6. Conclusion

The practical use of logical inference systems in intelligent systems is associated with the solution of systems of logical equations that can have a large dimension. The number of variables in this case usually exceeds the number of equations, which leads to the non-uniqueness of the solution. The use of the Zhegalkin algebra makes it possible to perform the algebraization of the problem, so that the scalar measure of the quality of the solution can be the Euclidean norm. The genetic algorithm is in this case an effective tool for finding a suboptimal solution.

7. References

[1] Jackson P 1998 *Introduction To Expert Systems* (Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA) p 542

[2] Gorodetskiy A E, Kurbanov V G and Tarasova I L 2017 Behavioral Decisions Of A Robot Based On Solving Of Systems Of Logical Equations *Smart Electromechanical Systems: The Central Nervous System*, ed A E Gorodetskiy and V G Kurbanov (Springer International Publishing AG) p 270

[3] Zhegalkin I I Arifmetizatsiia simvolicheskoi logiki [Arithmetization symbolic logic] *Matematicheskii sbornik* 1928 **35(3-4)** p 35 (in Russian)

[4] Dubarenko V V and Kurbanov V G The Method of Bringing the Systems of Logical Equations in the Form of Linear Sequential Machines *Informatsionno – izmeritel'nye i upravliaiushchie sistemy* 2009 **4(7)** pp 37-40 (in Russian).

[5] Burakov M V 2008 *Genetic algorithm: theory and practice* (SPb: GUAP) p 164 (in Russian).

[6] Goldberg D E 1989 *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison-Wesley, Reading, MA) p 412

[7] Burakov M V and Kurbanov V G Fuzzy PID Controller for Nonlinear Plant *ARPN J. of Eng. and Appl. Sci.* 2016 **11(9)** pp 5745-5748

[8] Burakov M V and Kurbanov V G Neuro-PID control for nonlinear plants with variable parameters *ARPN J. of Eng. and Appl. Sci.* 2017 **12(4)** pp 1226-1229

Acknowledgments

This research is supported by RFBR (grant 16-29-04424, 18-01-00076, 15-07-04760 and 18-51-06003).