Implicit Boolean Network for Planning Actions

G.A. Oparin*, V.G. Bogdanova * and A.A. Pashinin*
* Matrosov Institute for System Dynamics and Control Theory of SB RAS, Irkutsk, Russia
prn51@icc.ru, bvg@icc.ru, apcrol@gmail.com

***Abstract* - It is usual to classify the automated searching of a plan as an artificial intelligence problem. In planning, the environment is defined as a set of both states and transitions between them. Planning consists in searching for a finite transitions sequence (actions) that transfers the initial state to one of the states, satisfying the objective. As a model of the planning environment, we propose to use implicit Boolean networks – binary dynamic systems, the transition function of which is defined by the implicit Boolean equation relative to the current state variables vector and some subsequent variables vectors. For this model, the forming of a plan is reduced to qualitative analysis of dynamic properties of the implicit binary dynamic system. In particular, these are reachability or cyclicity properties over a given discrete time interval. At the final stage of this method, the verification of the Boolean formula satisfiability is performed. This formula includes the equation of the dynamics of the implicit Boolean network and the specification of the dynamical property being verified. The considered approach is demonstrated on the problem of planning a river crossing and constructing a Hamiltonian cycle in a graph. A specialized parallel algorithm for solving deeply structured Boolean satisfiability problems is proposed, which provides scalability with an increase in their dimension.**

Keywords – implicit Boolean networks; planning actions; Boolean satisfiability problem; parallel SAT solver.

I. INTRODUCTION

It is known that action planning is a classic problem of artificial intelligence. The planning environment is defined as a state set and transitions between them. The planning process consists in searching for a final transitions sequence (actions) that transfer the initial state to one of the states that satisfy the goal.

Thus, according to this classical definition of the planning action problem, the environment model is a dynamic system. The desired plan is a solution to this system that satisfies the given initial and goal conditions. Using dynamic models is a fundamentally new approach to solving the planning action problem. This approach differs significantly from static planning formalizations such as deduction or satisfiability [1].

Many logical puzzles can be considered a specific case of the planning problem. These include puzzles related to river crossings (the farmer-wolf-goat-cabbage problem, the missionary-cannibal problem), the Knight's tour problem, the famous Game of Fifteen, and many other problems. Such problems specificity is that the state space represents a set of Boolean vectors of the corresponding dimension. Our research is focused on this problems class.

The main feature of the planning problem is that the environment (exactly, transition diagram) is non-deterministic. The transition is allowed only in one state of the possible states set. Solving this problem within the framework of a Boolean formalism requires the involvement of new models of binary dynamical systems (BDS), which we define as implicit Boolean networks.

The article is organized as follows. Section II provides a brief overview of using implicit Boolean networks for solving the class of problems under consideration. Section III presents the problem statement and solution method. Section IV and V consider the application of implicit Boolean networks to solving the problems of crossing a river and finding a Hamiltonian cycle in a graph, respectively. Sections VI and VII present a parallel algorithm for solving these problems and the results of computational experiments. Section VIII contains a conclusion about the results achieved in the study course.

II. RELATED WORK

In [2], the simple implicit Boolean networks (of the first order) motivated by the well-known river crossing problem were proposed. In [3], for the first time, we considered high-order networks (greater than one) and their application to solving a specific class of combinatorial problems. In particular, based on such network use, the well-known minimum set cover problem, which has many applications, was solved.

In the authors' work [4], the method of Boolean constraints was developed. Its possibilities were demonstrated on qualitative analysis problems of the dynamic properties of classical Boolean networks operating on a limited time interval.

Notably, this method is a sufficiently general tool for qualitative analysis of various types of BDS and a wide range of dynamic properties over a finite time interval.

The purpose of this paper is to apply this method for a qualitative study of implicit Boolean networks whose transition function is subject to an implicit Boolean equation $F(x^t, x^{t+1}, \ldots, x^{t+p}) = 0$ that is unsolvable relative to the state x^{t+p}.

III. PROBLEM STATEMENT AND SOLUTION METHOD

Implicit Boolean networks are binary dynamic systems, the transition function of which is defined by an implicit Boolean equation relative to the vector of variables of the current state and the vectors of variables of several subsequent states.

Let the initial state has index zero, and all subsequent states are assigned indices in the range from one to the value *p*. The value *p* (by analogy with implicit differential equations) is called the order of the dynamic implicit equation of the Boolean network.

Thus, we consider a BDS whose behavior in time is given by an implicit Boolean equation of the *p*-th order [5] of the following form:

$$F(x^t, x^{t+1},\ldots,x^{t+p}) = 0, \qquad (1)$$

where $x^{t+\tau} = x(t+\tau) = (x_1(t+\tau), x_2(t+\tau),\ldots,x_n(t+\tau))$ – is a binary state vector in the time moment $t+\tau$, ($x^{t+\tau} \in B^n$, $B=\{0,1\}$, $\tau \in [0, p]$ – is a local time); $t = 0,1,2,\ldots,k$ – is a discrete time, F – is a scalar logic algebra function of their arguments ($F : B^{n(k+1)} \to B$). It is assumed that the function F does not have the unique solvability property relative to the state vector variables x^{t+p}. In other words, the behavior of BDS (1) is typically non-deterministic. Equation (1) may hold for suitably chosen time sequences of states $X^0 = (x^0, x^1,\ldots,x^p)$ of the length p+1, starting at the time $t = 0$. Such sequences are called local solutions of (1) (or local trajectories).

The topology of the set of local solutions (as admissible initial conditions of (1)) in the state space B^n may include branching states, equilibrium states, and various closed configurations types of the state sequence. Topology analysis of the local solutions set is an independent problem, which is not considered here.

With regard to the dynamics of an implicit Boolean network, the main property is the extensibility of its local solutions. A local solution is extendable by one step if its last *p* states coincide with the first *p* states of at least one local solution from the set of all local solutions of (1). All extendable local solutions satisfy the following equation of one-step transitions of the implicit Boolean network for $t = 0$ and $t = 1$:

$$F(x^0, x^1,\ldots,x^p) \vee F(x^1, x^2,\ldots,x^{p+1}) = 0. \qquad (2)$$

Solutions to this equation define a directed graph (extensibility graph) with vertices corresponding to local solutions and arcs directed from the local solution $X^0 = (x^0, x^1,\ldots,x^p)$ to the local solution $X^1 = (x^1, x^2,\ldots,x^{p+1})$. The solution X^1 will be called the successor of the solution X^0, and X^0 - the predecessor of X^1.

A local solution is extendable if at least one value of x^{p+1} satisfies equation (2), i.e., (2) is solvable for x^{p+1}. Thus, three different situations are possible relative to a given local solution $X^0 = (x^0, x^1,\ldots,x^p)$:

1. If equation (2) has a unique solution, then the local solution X^0 is called deterministic (i.e., X^0 has one successor $X^1 = (x^1, x^2,\ldots,x^{p+1})$).
2. If equation (2) has several solutions, then X^0 is called a local branching solution (X^0 has several successors, their number is equal to the number of solutions according to the state x^{p+1} of equation (2)).
3. If equation (2) has no solutions, then the local solution X^0 is called dead-end (X^0 has no successors).

The extensibility property is illustrated for the following simple example from [5] of an implicit Boolean network of the second-order (*p*=2, *n*=1):

$$x^{t+2} \wedge x^t \oplus x^{t+1} = 0, \quad x^{t+\tau} \in B, \quad \tau = 0,1,2. \qquad (3)$$

This equation has four local solutions $X^0 = (x^0, x^1, x^2)$: (000), (001), (100), (111). The successors of each of these solutions are found from the following equation of one-step transitions:

$$(x^2 \wedge x^0 \oplus x^1) \vee (x^3 \wedge x^1 \oplus x^2) = 0. \qquad (4)$$

The extensibility graph of local solutions is shown in Fig. 1.

In case the implicit Boolean equation (1) has solutions on the time interval $t \in T = \{0,1,2,\ldots,k\}$, each of its fragments of length p+1 must be an extendable local solution, that is, satisfy the condition

$$\Phi(x^0, x^1,\ldots,x^p, x^{p+1}, x^{p+2},\ldots,x^{p+k}) = \bigvee_{t=o}^{k} F(x^t, x^{t+1},\ldots,x^{t+p}) = 0. \qquad (5)$$

The main difference between an implicit Boolean network and a classical (automata) [4] or singular [6] network is that the initial condition in an implicit network is specified not as a state $x^0 \in B^n$ but as a finite sequence of states $X^0 = (x^0, x^1,\ldots,x^p)$, that satisfies the condition

$$F(x^0, x^1,\ldots,x^p) = 0.$$

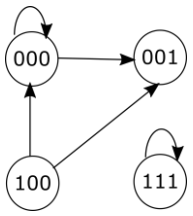

Figure 1. Extensibility graph of local solutions for equation (4). Here (000), (100) – branching solutions, (111) – deterministic

If BDS (2) has local branching solutions, its behavior is typically non-deterministic. Namely, some initial conditions X^0 may correspond to a set of solutions to (5). This solution set is called a funnel of trajectories. This factor must be considered when formulating the necessary dynamic properties of an implicit Boolean network within the framework of the Boolean constraint method we use. A similar situation arises in the qualitative analysis of singular Boolean networks [6]. For equation (5), as a model of action planning, proofs of the two following properties are most interested. Firstly, the reachability property of a target state from a given initial state in *k* steps. Secondly, the trajectory periodicity. In addition, these proofs should be constructive.

IV. River Crossing Planning Problem

Different approaches to solving this problem are described in publications. For example, in [5], for constructing a model, Boolean differential equations and methods of their study are used. In [7], this problem is solved using the fundamental ideas of the model-checking algorithm. The semi-tensor product method is used in [2].

We use the Boolean constraints method [4], which allows, in addition to the reachability property, to investigate a wide range of other dynamic properties.

The essence of the problem is as follows. The farmer must transfer a wolf, a goat, and a cabbage bag across the river from the left bank to the right bank. Only a farmer can drive the boat, only one "passenger" can fit in the boat with the farmer, or the farmer crosses the river alone. An additional restriction is that the goat with the wolf, and the cabbage bag with the goat, cannot be left unattended by the farmer.

Let us introduce the Boolean state vector $x=(x_1,x_2,x_3,x_4)$, where the variables x_1,x_2,x_3,x_4 describe the states of the farmer, wolf, goat, and cabbage, respectively. The initial state $x_1=x_2=x_3=x_4=0$ means that all participants of the crossing river are on the left bank. The goal state $x_1=x_2=x_3=x_4=1$ means that all participants are on the right bank.

At any moment time $t\in T=\{0,1,2,\ldots,k\}$, one of four actions can be performed:

- A_1 – the farmer crosses alone;
- A_2 – the farmer transports the wolf;
- A_3 – the farmer transports the goat;
- A_4 – the farmer transports the cabbage.

Following the Boolean formalism, an action is understood as a transition from one state from B^4 to another state from B^4. Therefore, for each action, we assign a Boolean dynamic system that implements this transition:

$$\begin{array}{cccc} A_1: & A_2: & A_3: & A_4: \\ x_1^{t+1}=\bar{x}_1^t & x_1^{t+1}=\bar{x}_1^t & x_1^{t+1}=\bar{x}_1^t & x_1^{t+1}=\bar{x}_1^t \\ x_2^{t+1}=x_2^t & x_2^{t+1}=\bar{x}_1^t & x_2^{t+1}=x_2^t & x_2^{t+1}=x_2^t \\ x_3^{t+1}=x_3^t & x_3^{t+1}=x_3^t & x_3^{t+1}=\bar{x}_1^t & x_3^{t+1}=x_3^t \\ x_4^{t+1}=x_4^t & x_4^{t+1}=x_4^t & x_4^{t+1}=x_4^t & x_4^{t+1}=\bar{x}_1^t \end{array}$$

The Boolean equations equivalent to these dynamical systems are:

$$\begin{aligned} &A_1(x^{t+1},x^t)=(x_1^{t+1}\oplus\bar{x}_1^t)\vee(x_2^{t+1}\oplus x_2^t)\vee \\ &\vee(x_3^{t+1}\oplus x_3^t)\vee(x_4^{t+1}\oplus x_4^t)=0 \\ &A_2(x^{t+1},x^t)=(x_1^{t+1}\oplus\bar{x}_1^t)\vee(x_2^{t+1}\oplus\bar{x}_1^t)\vee \\ &\vee(x_3^{t+1}\oplus x_3^t)\vee(x_4^{t+1}\oplus x_4^t)=0 \\ &A_3(x^{t+1},x^t)=(x_1^{t+1}\oplus\bar{x}_1^t)\vee(x_2^{t+1}\oplus x_2^t)\vee \\ &\vee(x_3^{t+1}\oplus\bar{x}_1^t)\vee(x_4^{t+1}\oplus x_4^t)=0 \\ &A_4(x^{t+1},x^t)=(x_1^{t+1}\oplus\bar{x}_1^t)\vee(x_2^{t+1}\oplus x_2^t)\vee \\ &\vee(x_3^{t+1}\oplus x_3^t)\vee(x_4^{t+1}\oplus\bar{x}_1^t)=0 \end{aligned}$$

An implicit Boolean equation whose solutions represent all possible actions at time *t* is written as

$$\begin{aligned} G(x^{t+1},x^t)=A_1(x^{t+1},x^t)\wedge A_2(x^{t+1},x^t)\wedge \\ \wedge A_3(x^{t+1},x^t)\wedge A_4(x^{t+1},x^t)=0 \end{aligned}. \quad (6)$$

Let us follow the conditions of the problem. The goat is on the same bank as the wolf or the cabbage. Therefore, the farmer is on the same bank. These constraints are written as the following equations:

$$\begin{aligned} H_1(t)&=[(x_3^t\equiv x_2^t)\vee(x_3^t\equiv x_4^t)]\wedge(x_3^t\oplus x_1^t)=0 \\ H_2(t+1)&=[(x_3^{t+1}\equiv x_2^{t+1})\vee(x_3^{t+1}\equiv x_4^{t+1})]\wedge \\ &\wedge(x_3^{t+1}\oplus x_1^{t+1})=0 \end{aligned} \quad (7)$$

Thus, the plan for the crossing river for passengers is a solution to the following implicit Boolean equation:

$$\begin{aligned} \Phi(x^0,x^1,\ldots,x^{k+1})=\bigvee_{t=0}^{k}(G(x^t,x^{t+1})\vee \\ \vee H_1(x^t)\vee H_2(x^{t+1}))=0 \end{aligned} \quad (8)$$

for $x^0=(0,0,0,0)$, $x^k=(1,1,1,1)$, and $k=7$.

For $k=0$, all local solutions are found. For $k=1$, all local solutions extendable by one step are found. Local solutions are shown in Table 1. In Fig. 2, a graph of crossing plans corresponding to these local solutions is shown.

V. THE PROBLEM ON THE HAMILTONIAN CYCLE IN A GRAPH

Let us follow the problem statement from [8]. Let $V=\{v_1,v_2,...,v_n\}$ be a set of graph vertices, and $A=\|a_{ij}\|$ be an adjacency matrix of vertices of dimension $n\times n$. The matrix element $a_{ij}=1$, if the vertex v_i is adjacent to the vertex v_j. It is required to find a closed path (Hamiltonian cycle - if it exists) that passes through each vertex of the graph exactly once. We propose the following formalization of this problem in terms of implicit Boolean equations and a qualitative study of the properties of such models using the method of Boolean constraints.

Let us denote by $A_i^{'}$ the set of indices of unit elements in the *i*-th row of the matrix *A*: $A_i^{'}=\{j:a_{ij}=1\}$, i.e., $A_i^{'}$ is the set of graph vertices adjacent to vertex v_i. We introduce a Boolean state vector $x=(x_1,x_2,...,x_n)$, where $x_i=1$ means that the vertex v_i is included in the desired path and $x_i=0$ otherwise for all $i=1,2,...,n$.

First, we write down the equations for the Hamiltonian path existence in a graph. Let us represent the algorithm for constructing such paths as a dynamic process of changing the state *x* at successive moments of local time $\tau=1,2,...,n$ according to the following implicit Boolean equation (1) for $t=0$:

$$F(x^1,x^2,...,x^n)=0, \qquad (9)$$

where x^i means the state of the process at the moment of local time $\tau=i$. Each step of such a process (action) consists in including only one vertex in the desired path. Under these assumptions, the local trajectory $x^1,x^2,...,x^n$ must satisfy the following Boolean constraints:

- At each step in the path, there should be only one vertex of the graph (each state should contain one and only one unit):

$$F_1=\bigvee_{\tau=1}^{n}\Big(\bigvee_{i=1}^{n-1}\bigvee_{j=i+1}^{n}x_i^{\tau}\wedge x_j^{\tau}\vee \vee\bigwedge_{p=1}^{n}\bar{x}_p^{\tau}\Big)=0; \qquad (10)$$

- Each vertex v_j can be placed in the path once and only once, or each local trajectory for variable x_j contains one and only one unit:

$$F_2=\bigvee_{j=1}^{n}\Big(\bigvee_{\tau=1}^{n-1}\bigvee_{\tau_1=\tau+1}^{n}x_j^{\tau}\wedge x_j^{\tau_1}\vee \vee\bigwedge_{p=1}^{n}\bar{x}_j^{p}\Big)=0; \qquad (11)$$

TABLE 1. LOCAL SOLUTIONS (LS) FOR EQUATION (8)

No	LS	No	LS
1	0000 1010	11	1010 0000
2	0001 1011	12	1010 0010
3	0001 1101	13	1011 0001
4	0010 1010	14	1011 0010
5	0010 1011	15	1101 0001
6	0010 1110	16	1101 0100
7	0100 1101	17	1101 0101
8	0100 1110	18	1110 0010
9	0101 1101	19	1110 0100
10	0101 1111	20	1111 0101

- Conditions for choosing a valid vertex for each moment of local time τ (adjacent vertices in the path must be adjacent in the graph):

$$F_3=\bigvee_{\tau=1}^{n-1}\bigvee_{j=1}^{n}\bigvee_{i\notin A_j}x_j^{\tau}\wedge x_i^{\tau+1}=0. \qquad (12)$$

Thus, equation (9) takes the following form:

$$F(x^1,x^2,...,x^n)=F_1\vee F_2\vee F_3=0. \qquad (13)$$

The set of local solutions to (13) (if it is not empty) determines the Hamiltonian paths set in the graph. Evidently, the set of local solutions, extended by one step, determines the Hamiltonian cycles in this graph. Such cycles satisfy the one-step transition equation:

$$F(x^1,x^2,...,x^n)\vee F(x^2,x^3,...,x^n,x^{n+1})=0, \qquad (14)$$

which ensures the fulfillment of the condition $x^{n+1}=x^1$.

VI. PARALLEL ALGORITHM

Deeply structured Boolean models that reflect both the dynamics of the implicit Boolean network and the specification of the extensibility property of its local solutions represent the considered problems. To test the satisfiability of high-dimensional Boolean models (in particular, the Knight's tour problem), whose structure includes constraints of length 2 (mainly) and constraints of length *p* (*p* much greater than two, *p*>>2), a specialized

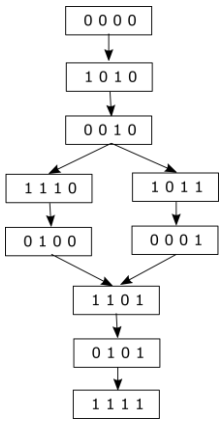

Figure 2. Decision graph for equation (8)

algorithm has been developed. This algorithm is based on data parallelism by splitting the Boolean model, which means choosing a variable from a Boolean model and replacing the original model with submodels for the two truth values of the selected variable. The algorithm uses two-level splitting. Queues of the top and bottom levels are formed.

The top-level queue Q^l includes sets of variables chosen for splitting - ordered *l*-tuples with consistent truth-values $S_i^l =< x_{i_1}^1, x_{i_2}^2, ..., x_{i_l}^l >, i = \overline{1, N}$, where N is the number of tuples in the queue, and $i_1 < i_2 < ... < i_l \le n \cdot l$ (n is the dimension of the state vector of system (9)). At the top level, all admissible *l*-tuples are found using the specialized author's AllSAT solver satall2p for an auxiliary SAT problem. This SAT problem is generated using the matrix *A* and constraints reflecting the system dynamics for a given splitting depth *l*. In particular, for the Knight's tour problem, the auxiliary SAT problem is generated according to constraints (10), (12) - (13) with the value of the quantity *k*, which depends on the size of the problem.

The resulting *l*-tuples at the bottom level are extended in parallel from *l* to *l* + *d* variables for a given depth *d* of the search tree.

Bottom-level queues Q_i^{l+d} are formed, including (*l* + *d*)-tuples $S_{ij}^{l+d} =< x_{ij_1}^1, x_{ij_2}^2, ..., x_{ij_l}^l, x_{ij_{l+1}}^{l+1}, ..., x_{ij_{l+d}}^{l+d} >$, $j = \overline{1, M_i}$, where M_i is the number of tuples, which can be different for each *i*-th tuple of queue Q^l, and the condition $n \cdot l < j_1 < j_2 < ... < j_{l+d} \le n \cdot (l + d)$ is satisfied. SAT subproblems are generated according to the matrix *A*, the constraints of the Boolean model (in particular, the (10) - (14) for the Knight's tour problem), and the unary constraints corresponding to the resulting tuples S_{ij}^{l+d}. Satisfiability testing of subtasks is performed in parallel and independently by sequential SAT solvers.

The algorithm implementation is performed using the previously developed software platform HPCSOMAS-MSC [9]. The TLA (Top Level Agent) agent performs the actions of the algorithm at the top level, and the group of BLA agents (Bottom Level Agent) performs the actions of the algorithm at the bottom level (Fig. 3).

The proposed scheme significantly reduces the enumeration in the search space by discarding inconsistent tuples of variable values at the stage of queuing. Two levels of parallelization reduce the overhead in the formation and distribution of subtasks.

VII. Computational Experiment

Computational experiments were carried out to determine closed routes for the Euler problem of the Knight's tour. This problem is a particular case of finding a Hamiltonian cycle in a graph problem. This problem has many solutions, but at the same time, it is hard for SAT solvers due to the high dimension and the dependence of variables [10]. The problem model is represented using an implicit Boolean network.

TABLE 2. COMPUTATIONAL RESULTS

	6×6	**8×8**
Variables	1332	4160
Constraints	108379	625825
BLA maximum count	4	16
Threads of one BLA	4	4
l	1	1
d	1	1
Sequential Runtime (s)	33,47	3617,43
Parallel Runtime (s)	0,07	79,10

The purpose of the experiment was to verify the presence of the trajectory periodicity property of the implicit Boolean network. For this verification, the extensibility property of the local solution to (13) was used, i.e., finding a solution to (14).

For searching closed paths, problems were solved for the chessboard of dimensions 6×6 and 8×8. In the proposed parallel algorithm for a more difficult 8×8 problem, when choosing tuples from the queue at the top level, the heuristic was used. This heuristic considers the degree of the graph vertex equal to six. At the bottom level, the lexicographic order of selecting tuples from the queue was used.

The experiments were carried out on the cluster "Akademik V.M. Matrosov" [11]. BLA agents were installed on its nodes. The Minisat solver [12] was used to solve subproblems on threads. Table 2 shows the results of the experiments. The runtime speedup and efficiency characterizing the proposed parallel algorithm are shown in Fig. 4. The speedup is close to linear, and the efficiency remains above 0.7.

Table 3 shows the solution to (14) for the 6×6 problem (n=36). For each state x^i, after the equal sign in parentheses, the row and column of the corresponding cell of the chessboard are indicated, separated by commas. The local solution $x^1, x^2, ..., x^{36}$ (Hamiltonian path) is extendable by one-step so that condition $x^{37} = x^1$ is satisfied, i.e., the Hamiltonian path is a Hamiltonian cycle.

TABLE 3. SOLUTION TO (14) FOR PROBLEM 6×6

x^1=(1,4)	x^{10}=(5,3)	x^{19}=(1,6)	x^{28}=(1,1)
x^2=(2,6)	x^{11}=(6,5)	x^{20}=(2,4)	x^{29}=(2,3)
x^3=(3,4)	x^{12}=(4,6)	x^{21}=(1,2)	x^{30}=(1,5)
x^4=(1,3)	x^{13}=(5,4)	x^{22}=(3,1)	x^{31}=(3,6)
x^5=(2,5)	x^{14}=(6,6)	x^{23}=(5,2)	x^{32}=(5,5)
x^6=(3,3)	x^{15}=(4,5)	x^{24}=(4,4)	x^{33}=(4,3)
x^7=(2,1)	x^{16}=(6,4)	x^{25}=(6,3)	x^{34}=(6,2)
x^8=(4,2)	x^{17}=(5,6)	x^{26}=(5,1)	x^{35}=(4,1)
x^9=(6,1)	x^{18}=(3,5)	x^{27}=(3,2)	x^{36}=(2,2)
x^{37}=(1,4)			

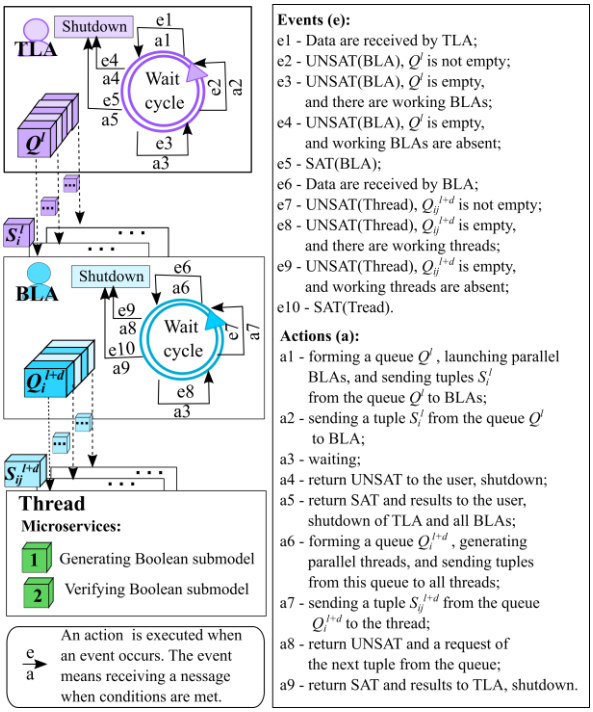

Figure 3. Scheme of parallel algorithm execution by TLA and BLA agents

VIII. Conclusion

In the article, Boolean networks, the behavior of which in time is described by an implicit Boolean equation of the *p*-th order, are considered. A definition of a local solution is given. Conditions of its extensibility to a finite number of steps are considered. From the point of view of one-step transitions, a classification of local solutions (deterministic, branching, dead-end) is given. A way for graphical representation of the dynamics of one-step transitions in the form of an extensibility graph in the space of local solutions is proposed. Boolean models are developed in the form of implicit Boolean equations for the problems of planning a river crossing and searching for a Hamiltonian cycle in a graph. The solving problem of planning actions is reduced to verifying dynamical properties based on the Boolean constraint method for qualitative analysis of BDS. Namely, the reachability property or the trajectory cyclicity property. The specialized parallel algorithm was developed for solving the Boolean satisfiability problem for a deeply structured formula that considered both the equation of dynamics of an implicit Boolean network and the dynamic property to be checked. This algorithm provides good scalability with increasing problem dimensions. Implicit Boolean networks of the *p*-th order and qualitative analysis of their dynamic properties using the Boolean constraint method have great potential for both a research perspective and practical applications.

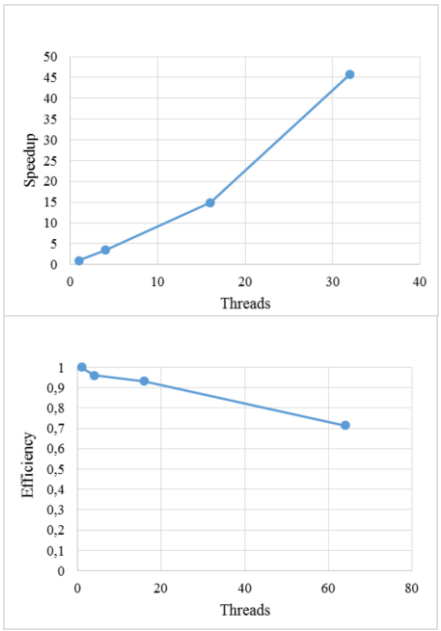

Figure 4. Runtime speedup and efficiency for 8×8

Acknowledgment

The study was supported by the Ministry of Science and Higher Education of the Russian Federation, project no. 121032400051-9. The authors would like to thank Irkutsk Supercomputer Center of SB RAS for providing access to HPC-cluster "Akademik V.M. Matrosov" [11].

References

[1] H. Kautz, B. Selman, Planning as satisfiability, "in Proc. of the 10th European Conference on Artificial Intelligence (ECAI 92), John Wiley and Sons, 1992.

[2] Y. Yu, J. Feng, M. Meng, and B. Wang, "Topological structure of implicit boolean networks", IET Control & Application, vol. 11, no. 13, pp. 2058-2064, 2017.

[3] G.A. Oparin, V.G. Bogdanova, and A.A. Pashinin, "Implicit Boolean Networks and Their Application to Combinatorial Problems," MESA, 2022, vol. 13, no. 1, pp. 25-35.

[4] G. Oparin, V. Bogdanova, and A. Pashinin, "Qualitative analysis of autonomous synchronous binary dynamic systems," MESA, vol. 10, no. 3, pp. 407-419, 2019.

[5] D. Bochmann and C. Posthoff, "Binary dynamic systems," Berlin: Academic Verlag, 320 p. , 1981.

[6] G. Oparin, V. Bogdanova, and A. Pashinin, "The Boolean Constraint Method Application for Qualitative Analysis of the Dynamical Properties of Singular Boolean Networks", in Proc. of the 2nd International Workshop on Advanced Information and Computation Technologies and Systems (AICTS 2021), Irkutsk, Russia, December 7-11, 2021, in press.

[7] Yu. Karpov, "Verification on parallel and distributed software systems", BKhV – Sankt-Peterburg, 560 p., 2010. (In Russian)

[8] Michael R. Garey, David S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", Freeman, 1979.

[9] G.A. Oparin, V.G. Bogdanova, A.A. Pashinin, and S.A. Gorsky, "Microservice-oriented approach to automation of distributed scientific computations," in Proc. of the 42nd Int. Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE, 2019, pp. 253-258.

[10] S. Prestwich, "SAT problems with chains of dependent variables," Discrete Applied Mathematics, 2003, vol. 130, no. 2, pp. 329-350.

[11] "Irkutsk Supercomputer Centre of SB RAS," http://hpc.icc.ru [online, accessed: January 31, 2022].

[12] N. Eén, N. Sörensson, "An Extensible SAT-solver," in Proc. of the 6th Int. Conference on Theory and Applications of Satisfiability Testing, 2003.