Research of Improved Tissue-like P System for Solving Satisfiability Problem

Huang Yourui, Ge Pingping, Xu Jiachang, Tang Chaoli
School of Electrical and Information Engineering, Anhui University of Science and Technology
Huainan, Anhui, China, 232000
hyr628@163.com

***Abstract*—Satisfiability problem is the first NP-complete problem that is not only the basic problem of logic but also the main content of computer science and artificial intelligence research nowadays. In this paper, the maximum parallelism of P system and the basic tissue-like P system are utilized to imitate life activities of charged organisms in nature; a recognizable evolutionary tissue-like P system based on cell division is proposed to solve the satisfiability problem. The system can obtain exponentially growing computing cells by cell division in a short time and each cell processes one truth value assignment situation; carries out parallel processing of the Boolean expression in polynomial time by using communication rules, splitting rule, evolutionary rule and output rules. To prove the correctness and efficiency of the algorithm, this paper completes the simulation of the algorithm by using pLingua Plugin that is dedicated to P system simulation, which proves that the algorithm is correct and feasible, and improves the calculation efficiency by changing the space for time.**

Keywords*—*Satisfiability problem, Membrane computing, Tissue-like P system

I. Introduction

As the first NP-complete problem that has been proved, the SAT problem (Satisfiability problem) of Boolean logic expression has laid a solid foundation for the theory of computer complexity and been widely used in many fields such as hardware circuit feasibility testing and computer science. In polynomial time, the satisfiability problem can sum up other NPC problems in its scope. Thus, it is not only the basic problem of logic but also the main content of computer science and artificial intelligence research nowadays [1]. The SAT problem is to judge a propositional logic formula given in the form of a Boolean logic expression to find out whether there is a truth value assignment that makes the result of the propositional logic formula true. The SAT problem exists on the condition of Boolean logic. All variables are Boolean variables of which values can only be 0 (false) or 1 (true). If a set of truth value assignments can make the result of the expression 1, then the Satisfiability Problem (SAT) is satisfied. Otherwise, UNSAT. The SAT problem based on conjunctive normal form is shown in Equation (1).

$$\begin{cases} F(X) = C_1 \wedge C_2 \wedge \cdots \wedge C_m \\ X = (x_1, x_2, \cdots, x_n) \\ C_i = x_{i_1} \vee x_{i_2} \vee \cdots \vee x_{i_M} \vee \overline{x_{j_1}} \vee \overline{x_{j_2}} \vee \cdots \vee \overline{x_{j_N}} \\ i \in [1, m] \\ i_1, i_2, \cdots, i_M \in [1, n] \\ j_1, j_2, \cdots, j_N \in [i, n] \end{cases} \tag{1}$$

Where X is a set of Boolean variables; text can be variable ($Q = x_i$) or variable inversion ($Q = \overline{x_i}$), and each clause C is composed of "or" of some texts and represented by symbol $\vee$; the “and” of some clauses form a conjunctive normal form F that is represented by symbol $\wedge$. If the sum of i_M and of j_N of each clause is k, $F(X)$ will be the $k-SAT$ problem [2].

Many scholars have proposed many classical and effective solutions to SAT problems, including complete algorithm and incomplete algorithm. Complete algorithms usually search the whole search space and can always determine whether a propositional formula can be satisfied or not. However, computational efficiency of such algorithm is low as or the time complexity is usually exponential in large-scale SAT problems. The incomplete algorithm does not perform complete search but only searches part of the search space in a limited time, which usually fails to correctly judge the satisfiability of SAT problems.

Membrane computing, also known as P system, is a kind of natural computing that derived from the network structure, function and relationship of biological cells, tissues and organs and the most basic model that be able to reflect the essence of computing. Randomness and maximum parallelism are its two main characteristics [3]. As membrane computing can realize maximum parallelism computation, it is far more powerful than ordinary computers [4]. Thus, membrane computing is used to solve the satisfiability problem in this paper due to its capability of high-strength computation [5].

II. Recognizable evolutionary tissue-like P System

Since tissue-like P system is composed of basic cell units and cannot be separated from the properties of basic cells, a new membrane computing model is established by fusing the basic cell-like P system and the recognizable tissue-like P system [6] based on cell division. That is, the recognizable revolutionary tissue-like P system based on cell division can be used to solve the Satisfiability problem.

Degree $m \geq 1$ recognizable revolutionary tissue-like P system based on cell division can be expressed as Equation (2).

$$\Pi = (O, E, w_1, \dots w_m, R, i_0) \quad (2)$$

Where,

(1) $m \geq 1$, degree is the number of initial membranes;

(2) O is an object alphabet set;

(3) E is the set on O that represent any object set in the environment;

(4) $w_1, \dots, w_m$ is the character string on O that represents the initial object multiset in the region that labeled m;

(5) R is a series of rule set that mainly includes:

(a) Communication rule:

$$(\alpha i, a/b, \beta j),\ 1 \leq i \leq m, 1 \leq j \leq m, i \neq j, \alpha, \beta \in \{0, +, -\}, a, b \in O \quad (3)$$

$1, 2, \dots, m$ represent different membranes (abbreviation of cell membranes, the same below) and 0 represents the environment. Due to its polarity, it is not only authentic but also extends the computing power of membrane system by introducing biological cell membranes into the membrane system. Here, α, β represent the polarity of the membranes to which they belong: 0 denotes being neutral, + denotes carrying a positive charge and - denotes carrying a negative charge. Generally, 0 can be omitted in default. The rule indicates that object α in the layer i with polarity a enters the layer j with polarity β while object b in the layer j with polarity β enters the layer i with polarity a and complete the exchange of objects a and b. Weight of communication rules of the current structure will be 3 at most. If there are too many objects to be transferred, intermediate objects can be utilized for transformation and transport. For example, rule $(1, d\ e\ f\ h/k, 2)$ can be replaced by rule $(1, d\ e\ f/g, 0), (1, g\ h/k, 2)$ in which g is an object in the environment, that is, the intermediate object here. One side can only carry up 3 objects to be transferred at a time.

(b) Splitting rule:

$$[a]_i \to [b]_i [c]_i, \quad 1 \leq i \leq m, a, b, c \in O \quad (4)$$

In the biosphere, cells before division are mother cells which new cells after division are called daughter cells [7]. In the splitting rule, mother cell i is divided into two daughter cells i. In the first daughter cell, the object a is removed while the other objects are the same to i, and the object a is replaced by the object b. In the second daughter cell, the object a is removed while the other objects are also the same to i, and the object a is replaced by the object c.

(c) Evolutionary rule:

$$\alpha[a \to b]_i, 1 \leq i \leq m, a, b \in O, \alpha \in \{0, +, -\} \quad (5)$$

a evolves into b in the membrane i with polarity α; a and b still represent objects.

(d) Output rule:

$$[v\ a]_i \to a[v]_i, 1 \leq i \leq m, a, v \in O \quad (6)$$

With the aid of the object v, the object a is transported to the outside of membrane i, where the object v is equivalent to the transportation enzyme.

(6) $i_0 = 1$ indicates that the surface layer 1 is the output membrane.

III. Improved Tissue-Like Membrane System For Solving DNF-SAT Problem

The conjunctive normal form (CNF) is the standard form of logical formulas. Construct a logical formula of conjunctive normal form $\gamma = K_1 \vee \dots \vee K_m$ that is also made up of m clauses that contain $K_j = y_{j,1} \wedge \dots \wedge y_{j,k_j}, 1 \leq j \leq m$ [8], among which $y_{j,i} \in \{x_l, \overline{x_l} | 1 \leq l \leq n\}, 1 \leq i \leq k_j$ (although n variables will be needed, there will not be necessarily n texts in each clause). Similarly, there will not be two or more x_i and $\overline{x_i}$ in each clause [9].

Specific rule is shown in Equation (7).

$$[a_i]_2 \to [T_i]_2 [F_i]_2, i = 1, 2, \dots, n \quad (7)$$

Membrane 2 is divided into two parts, and each time object a_i will be evolved into T_i and F_i and distributed to corresponding sub-membranes (corresponds to variables *true* and *false* respectively). After the rule being carried out for one step, one membrane 2 will be divided to two membrane 2; after two steps, the two newly-obtained membranes will be divided to another two membranes 2 respectively as per former splitting manner, that is, four membranes 2 will be obtained. In this way, after n steps, 2^n membranes with the label 2 will be obtained, and 2^n times of truth value distribution of n variables will be carried out. That is, after "0" and "1" distribution of each variable, there will be 2^n combinations of truth value distribution and each daughter cell 2 will contain one combination situation, that is, each contains one arbitrary possibility. g is copied and there will be copies of it in each sub-membrane.

$$\left[b_i \to b^2_{i+1}\right]_1, 1 \le i \le n+1 \tag{8}$$

$$\left[c_i \to c_{i+1}\right]_1, 1 \le i \le 2n+nm+2m+4 \tag{9}$$

During the parallel processing of splitting of membrane 2, the subscripts of objects b_i and c_i in membrane 1 are also increased. In each step, object b_i multiplies, thus, 2^n b_{n+1} will be obtained after n step. b_i mainly catalyzes membrane 2 to be positively charged, which provides conditions for subsequent carrying-out of the rule and checks for whether there is the existence truth value distribution of all groups that makes all clauses unsatisfactory in later stages. c_i is mainly used to generate object yes.

$$[\,]_2 b_{n+1} \to +[b_{n+1}]_2 \tag{10}$$

In step $n+1$, the counter b_{n+1} in the membrane 1 enters the membrane 2, exchanges with the object g in the membrane 2 and becomes positive as the counter b_{n+1} enters the membrane 2. At this point, there will be 2^n membranes 2 and 2^n different types of object. According to the maximum parallelism of the rule, there will be 2^n objects g in membrane 1. Meantime, each membrane that labeled 2 will get a b_{n+1} through exchange. Please note that since objects a_i are exhausted, membrane 2 will no longer split.

$$+\left[T_i \to T_{i,1}\right]_2, i = 1, 2, \ldots, n \tag{11}$$

$$+\left[F_i \to F_{i,1}\right]_2, i = 1, 2, \ldots, n \tag{12}$$

$$+\left[T_{i,j} \to t_i T_{i,j+1}\right]_2, i = 1, 2, \ldots, n \text{ and } j = 1, 2, \ldots, m \tag{13}$$

$$+\left[F_{i,j} \to f_i F_{i,j+1}\right]_2, i = 1, 2, \ldots, n \text{ and } j = 1, 2, \ldots, m \tag{14}$$

The initial configuration is shown in Equation (15), (16), (17) and (18).

$$w_1 = yes\ no\ b_1\ c_1\ g \tag{15}$$

$$w_2 = a_1\ a_2 \cdots a_n \tag{16}$$

$$\left(+2, t_i s_{i,j} / v_j, 0\right), 1 \le i \le n \text{ and } 1 \le j \le m \tag{17}$$

$$\left(+2, f_i \overline{s_{i,j}} / v_j, 0\right), 1 \le i \le n \text{ and } 1 \le j \le m \tag{18}$$

t_i and f_i are used to check the text values in membrane 2. nm steps are needed here and the main ideas adopted are: not "0" is taken as 1 and "1" takes itself as 1. In each clause, as long as one character is 1, the clause in which it is located is 1. Subscript of c in membrane 1 increases continuously. Since $2n+m+1$ steps have been already implemented before the implementation of this rule, that is, subscript of c has reached $2n+m+1$, the subscript will become $2n+nm+m+1$ after finishing the text check.

$$+\left[b_{n+i} v_i \to b_{n+i+1}\right]_2, i = 1, 2, \ldots, m \tag{19}$$

After checking the text in each clause, each clause will be checked one by one. In all membranes 2, check whether all clauses that have gone through truth value distribution are satisfactory or not. For clauses satisfying truth value distribution, the subscript value of b will be increased. Thus, the subscript of b will reach the maximum value $n+m+1$ only if all clauses are satisfactory.

$$\left(+2, b_{n+m+1} / g, 1\right) \tag{20}$$

If all clauses in at least one membrane among 2^n membranes 2 conform to the truth value distribution, there will be b_{n+m+1}. At that time, it will exchange with object g to enter into membrane 1 and g will become an object in membrane 2.

$$\left[b_{n+m+1}\ \mathrm{yes}\right]_1 \to yes\left[b_{n+m+1}\ \right]_1 \tag{21}$$

The main process for solving the satisfiability problem with conjunctive normal form is shown in Figure 1.

(1) Initialization: Initial configuration also includes initial object configuration, initial structure configuration and adding input multiple sets;

(2) All membranes 1 and 2 perform a series of rules indefinitely and in maximum parallel: transportation rules, splitting rules, evolutionary rules and output rules;

(3) Check whether there are objects b_{n+m+1} in membrane 1. If there are, it shows that at least one truth value distribution makes the Boolean logic expression satisfactory; then, output the object yes and complete the calculation.

(4) If no object b_{n+m+1} is generated, it shows that none of the truth value distribution in inner membranes that labeled 2 satisfies Boolean logic expression and the object

g remains in the inner membrane 1; at this time, subscript of counter c in membrane 1 reaches $2n+nm+2m+4$ and output no to the environment under their combined action, indicating that the Boolean logic expression is unsatisfactory and the calculation ends.

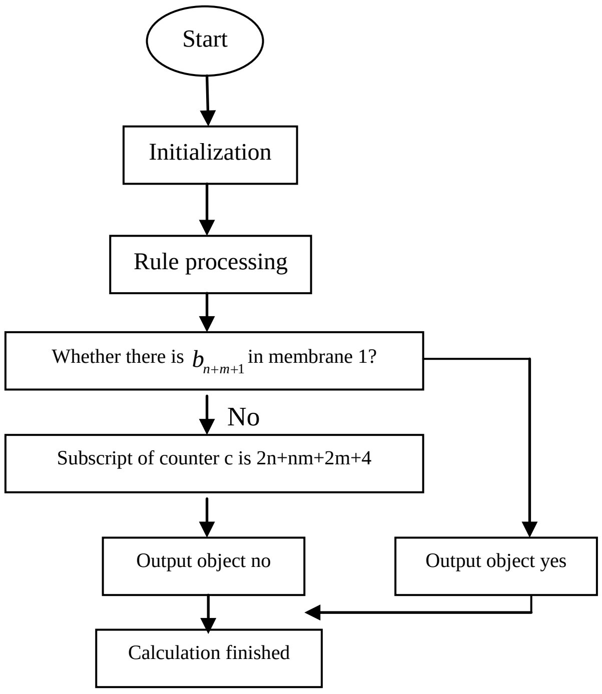

Fig. 1 The main flow chart

IV. Simulation Results

Aiming at the satisfiability problem of the most typical logic formula in NP-complete problem, a recognizable evolutionary tissue-like P system algorithm based on cell division is proposed and pLinguaPlugin is used. Also, simulation tests are carried out by introducing pLingua Core into Eclipse.

Structure of the initial membrane consists of only two nested membranes and both are not charged. Initial object g is in membrane 1not membrane 2.The input multiple sets in membrane 2 are {sp1, 1, sp2, 1, s5, 1, s1, 2, s3, 2, sp4, 2, sp2, 3, s5, 3, sp6, 3, s4, sp5, 4, s6, 4}.

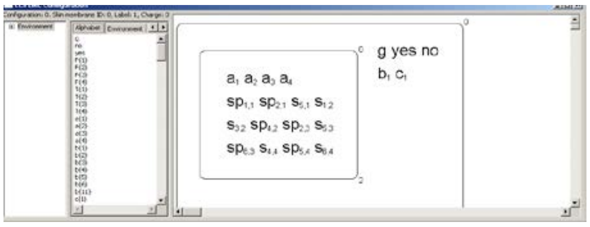
Alphabet
Environment
g yes no
b_1 c_1
a_1 a_2 a_3 a_4
$sp_{1,1}$ $sp_{2,1}$ $s_{5,1}$ $s_{1,2}$
$s_{3,2}$ $sp_{4,2}$ $sp_{2,3}$ $s_{5,3}$
$sp_{6,3}$ $s_{4,4}$ $sp_{5,4}$ $s_{6,4}$

Fig. 2 Diagram of the initial state

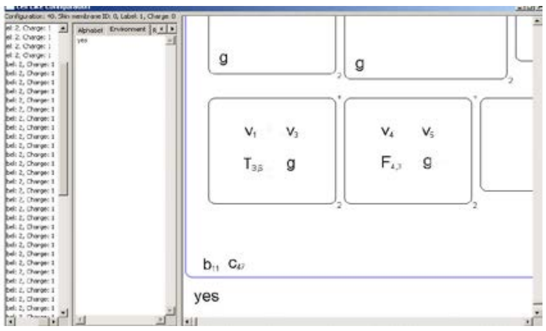

Fig. 3 Diagram of the final state

The final result yes is output to the environment and enters the final state. In the first column of the result bar on the left side, we can see the structure of the whole system. There is a surface layer inside the environment, and the surface layer contains parallel, split and charged membrane 2. The second column contains title bars of object, environment, rule, etc. and contents are shown in lower part. Here “yes” is displayed, meaning that the Boolean expression above can be satisfied. Similarly, on the right side of the window are the cases of each membrane and their internal object sets. There are objects b_{11} and c_{47} in the membrane 1 and too many objects in membrane 2 that are not explained here.

V. Conclusions

As an important branch of natural computing, membrane computing can provide a method to solve NP problems in polynomial time (usually linear time) by changing space for time. In this paper, computing cells with exponential growth are obtained through cell division so that each computing cell can deal with one truth value distribution situation. The Boolean expression is processed in polynomial time by using communication rule, splitting rule, evolutionary rule and output rule. Together with the pLinguaPlugin that dedicated to P system simulation, simulation of membrane computing is accomplished, proving the correctness and efficiency of the algorithm.

Acknowledgment

This work is funded by the National Natural Science Foundation of China (61772033).

References

[1] Gh.Pǎun, A quick introduction to membrane computing[J], The Journal of Logic and Algebraic Programming. 2010,79(6): 291-294.

[2] Pan L, Alhazov A. Solving HPP and SAT by P systems with active membranes and separation rules[J].Acta Informatica, 2006, 43(2):131-145.

[3] Gutiérrez-Naranjo M A, Pérez-Jiménez M J, Romero-Campero F J. A uniform solution to SAT using membrane creation[J]. Theoretical Computer Science, 2007, 371(1): 54-61.

[4] X.X. Zeng, T. Song, X.Y. Zhang, L.Q. Pan. Performing Four Basic Arithmetic Operationswith Spiking Neural P Systems[J]. IEEE Transactions on Nanobioscience,2012,11 (4) :366-374.

[5] T.O. Ishdorj, A. Leporati. Uniform solutions to SAT and 3-SAT by spiking neural P systemswith pre-computed resources[J]. Natural Computing, 2008, 7(4):519-534.

[6] Ishii K, Fujiwara A, Tagawa H. Asynchronous P systems for SAT and Hamiltonian cycle problem[C]. Nature and Biologically Inspired Computing (NaBIC), 2010 Second World Congress on. IEEE, 2010: 513-519.

[7] Gazdag Z, Kolonits G. A new approach for solving SAT by p systems

with activemembranes[M]. Membrane Computing: Springer Berlin Heidelberg, 2013: 195-207.

[8] P. Guo, J.F. Ji and H.Z. Chen, Solving All-SAT problems by P systems[J]. Chinese Journal ofElectronics, 2015, 24(4):744–749.

[9] W. Song, P. Guo, H. Z. Chen. Asolution for all-SAT problem based on P systems[J]. Journal ofComputational and Theoretical Nanoscience, 2016, 13(7):4293-4301.