RESEARCH ARTICLE

Spiking Neural P Systems with Neuron Division and Dissolution

Yuzhen Zhao, Xiyu Liu*, Wenping Wang

School of Management Science and Engineering, Shandong Normal University, Jinan, China

* sdxyliu@163.com

Abstract

Spiking neural P systems are a new candidate in spiking neural network models. By using neuron division and budding, such systems can generate/produce exponential working space in linear computational steps, thus provide a way to solve computational hard problems in feasible (linear or polynomial) time with a “time-space trade-off” strategy. In this work, a new mechanism called neuron dissolution is introduced, by which redundant neurons produced during the computation can be removed. As applications, uniform solutions to two NP-hard problems: SAT problem and Subset Sum problem are constructed in linear time, working in a deterministic way. The neuron dissolution strategy is used to eliminate invalid solutions, and all answers to these two problems are encoded as indices of output neurons. Our results improve the one obtained in *Science China Information Sciences*, 2011, 1596-1607 by Pan et al.

Citation: Zhao Y, Liu X, Wang W (2016) Spiking Neural P Systems with Neuron Division and Dissolution. PLoS ONE 11(9): e0162882. doi:10.1371/journal.pone.0162882

Editor: Andrew Adamatzky, University of the West of England, UNITED KINGDOM

Received: May 5, 2016

Accepted: August 30, 2016

Published: September 14, 2016

Data Availability Statement: All relevant data are within the paper and its Supporting Information files.

Funding: This research was supported by National Natural Science Foundation of China (http://www.nsfc.gov.cn/) 61472231 to XL, 61170038 to XL, 61402187 (not applicable) and 61502283 (not applicable). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The authors have declared that no competing interests exist.

Introduction

Spiking neural P systems (in short, SN P systems) are a class of bio-inspired parallel computing models, initiated by Ionescu, Păun and Yokomori in 2006 [1], which are inspired from information processing strategy and communication strategy between neurons. A SN P system is constructed by a group of neurons (a class of cells with only one membrane) communicating by sending signals (spikes, represented by object *a*) to neighboring neurons through synapses. Each neuron has a certain number of spikes and rules. Spikes can evolve through application of rules. Since SN P systems were proposed, they become a rapid developing area of membrane computing [2–15].

Researchers pay close attention to computational efficiency of SN P systems, especially the judgement whether NP-complete problems have solutions or not in feasible time [16–26]. If a NP-complete problem has a solution, the output neuron outputs a spike; otherwise, the output neuron outputs nothing. However, we need to find out the solutions in many situations. For instance, the register allocation problem is an application of SAT problem. This problem aims to build a mapping relationship between the virtual registers and the physical registers, and realize the rational utilization of physical register resources. In this case, we need to judge whether a good solution exists, while searching the solution by distributing the physical register resources according to the solution. In applications, many problems can be transformed into graph coloring problems, which is equivalent to SAT problems. To solve these problems, exact solutions are also essential.

For this purpose, neuron dissolution, which is a basic biological phenomenon aiming to remove unnecessary neurons, is introduced into SN P systems [27, 28], and a new class of SN P systems, SN P systems with neuron division and dissolution (DDSN P systems, for short) is proposed in this work. In DDSN P systems, division rules can generate exponent work space (in terms of neurons) which can be used to enumerate all possible results (one result is contained in one neuron), and dissolution rules can dissolve redundant neurons which can be used to remove wrong results. Neurons which represent all possible results are set as output neurons, and these output neurons with invalid results are dissolved in computational process. When the computation halts, the remaining output neurons show all right results. Uniform solutions to SAT problem and Subset Sum problem, which work in a deterministic way, are constructed as examples in this work.

The contributions of this work focus on the following three aspects. 1. The computational space efficiency is improved. If these redundant neurons are reserved, they will occupy huge computational resources such as storage. The dissolution rule can reduce the computational space needed and improve the computational space efficiency. 2. The system structure is clearer. If the redundant neurons are reserved, the SN P system will become complicated, and the useful neurons are not highlighted enough. By introducing the neuron dissolution mechanism, redundant neurons are dissolved immediately, and each of the remaining neuron has its function. 3. Exact solutions to NP-complete problems can be obtained in linear time. Invalid solutions are eliminated during the computational process by neuron dissolution, and all solutions are encoded as indices of specific output neurons at halting, which can provide more valuable information for applications. Uniform solutions to SAT and Subset Sum problems are solved as examples.

The paper is organized as follows. Section 1 defines the SN P systems with neuron division and dissolution. Uniform solutions to SAT and Subset Sum problems in linear time using the proposed SN P systems with neuron division and dissolution are presented in section 2 and section 3. Conclusions are given in section 4.

1 SN P Systems with Neuron Division and Dissolution

1.1 Background

Biological systems, such as cells, tissues, and human brains, have deep computational intelligence. Biologically inspired computing, or bio-inspired computing in short, focuses on regenerating computing architecture from biological systems to construct computing models and algorithms. Membrane computing is a novel research branch of bio-inspired computing, initiated by Gh. Păun in 2002, which seeks to discover new computational models from the study of biological cells, particularly of the biological membranes [29, 30]. The obtained models are distributed and parallel bio-inspired computing devices, usually called P systems. There are three mainly investigated P systems, cell-like P systems, tissue P systems, and neural-like P systems (also known as spiking neural P systems). P systems, known as powerful computing models, are able to do what Turing machine can do, even solving computational hard problems [31–37].

SN P systems, as a new branch of membrane computing, are a shift from the cell-like architecture to the neural-like architecture. The topological structure of SN P systems is a directed graph: neurons are placed in the vertices of the graph, and synapses act as edges. Each neuron can have a certain number of objects *a* (spikes) and a certain number of *firing* rules and *forgetting* rules. Through a firing rule, a neuron can send information to other neurons by emitting spikes to these neurons. Through a forgetting rule, a certain amount of spikes can be removed from a neuron. Both the firing rules and the forgetting rules have conditions of applied. If the number of spikes in a neuron is contained within in the number set of spikes determined by a regular expression, a rule can have the possibility to be applied. At each time step, one rule is non-deterministically chosen to be applied in each neuron. That is to say, rules are applied in a sequential manner from the view of the neuron, and in parallel from the view of the whole system.

Pan et al. introduced a novel idea to solve SAT problem in polynomial time by using neuron division and budding [25], which uses the neuron division and budding rules to generate more neurons according to the need in the computational process. Wang et al. proved that SN P systems with neuron division, not using neuron budding, can also solve SAT problem in polynomial time [26]. The biological motivation of neuron division and budding comes from the neural stem cells division. The neural stem cells have the ability to proliferate and differentiate into neurons, astrocytes and oligodendrocytes, therefore, they can supply massive tissue cells. In these SN P systems, neuron division rules and neuron budding rules are used to regenerate the above biological phenomena.

In neurons, there is another biological phenomenon called neuron apoptosis, which has a close relationship with neuron division and budding. Neuron apoptosis is a programmed neuron death controlled by a series of activities controlled by genes, such as the activation, expression and regulation of genes. It is not a self damage phenomenon under the pathological condition, but a actively death process. When unnecessary neurons or abnormal neurons occur in the process of neuron development or under the influence of some factors, neuron apoptosis can remove these neurons in multicellular organism to maintain a stable internal environment and to adapt to the environment better. It plays an important role in the evolution of the organism, the stability of internal environment and the development of multiple systems.

For the above biological phenomena, the neuron apoptosis mechanism is introduced into SN P systems, and neuron dissolution rule is designed. In this way, redundant neurons can be eliminated immediately.

1.2 System description

A SN P system with neuron division and dissolution of degree m is a construct of the form

$$\Pi = (O, H, syn, n_1, n_2, \ldots, n_m, R, in, out),$$

where:

1. $O = \{a\}$ represents the *singleton alphabet* where a is the spike;
2. H represents the set of *labels* for neurons;
3. $syn \subseteq H \times H$ represents a *synapse dictionary* (for each $1 \leq i \leq m$, $(i, i) \notin syn$);
4. $n_i \geq 0$ represents *the spike numbers* in neuron σ_i in the initial state ($1 \leq i \leq m$);
5. R represents the set of all *developmental rules* of the following four forms
   - *firing rule* $[E/a^c \rightarrow a^p; d]_i$, where, $i \in H$, E is a regular expression over a, $c \geq 1$, $p \geq 1$, $c \geq p$, $d \geq 0$. If $E = a^c$, the firing rule is simply written as $[a^c \rightarrow a^p; d]_i$. If $d = 0$, the firing rule is simply written as $[E/a^c \rightarrow a^p]_i$. If $E = a^c$ and $d = 0$, the firing rule is simply written as $[a^c \rightarrow a^p]_i$;
   - *forgetting rule* $[E/a^s \rightarrow \lambda]_i$, where, $i \in H$, E is a regular expression over a, $s \geq 1$. If $E = a^s$, the forgetting rule is simply written as $[a^s \rightarrow \lambda]_i$;

- *neuron division rule* $[E]_i \to [\ ]_j \parallel [\ ]_k$, where, $i, j, k \in H$, E is a regular expression over a;
- *neuron dissolution rule* $[E]_i \to \delta$, where, $i \in H$, E is a regular expression over a, object δ represents that neuron σ_i is dissolved;

6. *in*, *out* $\subseteq H$ represent *the input and output neurons* of Π, respectively.

The synapse dictionary *syn* shows the initial structure of the system and guides how to establish new synapses when new neurons are established.

If neuron σ_i has h spikes, and $a^h \in L(E)$, $h \geq c$, the firing rule $[E/a^c \to a^p; d]_i$ can be applied. c spikes are consumed ($h - c$ spikes remain in neuron σ_i.), and p spikes are emitted after d time units (steps). If $d = 0$, p spikes are emitted immediately; if $d = 1$, p spikes are emitted at the next step; if this firing rule is applied at step t and $d \geq 1$, p spikes are emitted at step $t + d$. Neuron σ_i is closed at steps $t, t + 1, t + 2, \ldots, t + d - 1$, which means no rule will be applied and no spike will be received in this period. At step $t + d$, neuron σ_i becomes open again, and can receive new spikes. Once these p spikes are emitted from neuron σ_i, they reach each neuron σ_j which has a synapse going from neuron σ_i to neuron σ_j and is open. The spikes sent to a closed neuron are *lost*.

If neuron σ_i has h spikes, and $a^h \in L(E)$, $h \geq s$, the forgetting rule $[E/a^s \to \lambda]_i$ can be applied. s spikes are consumed immediately.

If (1). neuron σ_i has h spikes, and $a^h \in L(E)$, and (2). no synapse (i, j), (j, i), (i, k), (k, i) exists in the system, the neuron division rule $[E]_i \to [\ ]_j \parallel [\ ]_k$ can be applied. All h spikes in neuron σ_i are consumed and neuron σ_i is divided into two neurons σ_j and σ_k. No spike is in neurons σ_j and σ_k at this moment. The labels of the two generated neurons can be different or the same, and the labels of the two generated neurons can be different from or the same with the label of their father neuron σ_i, too. The new generated neurons inherit the synapses of their father neuron σ_i. That is to say, if there is a synapse (i, g) going from neuron σ_i to neuron σ_g, two synapses (j, g) and (k, g) are established after the division rule is applied; if there is a synapse (g, i) going from neuron σ_g to neuron σ_i, two synapses (g, j) and (g, k) are established after the division rule is applied. In addition to inheritance of synapses, new generated neurons also have synapses provided by the synapse dictionary *syn*. Synapses not existing in the synapse dictionary *syn* may appear because of inheritance of synapses. The condition (2) avoids the situation that the start and the end of a synapse are the same neuron. For example, if synapse (i, j) exists in the system, synapses (j, j), (k, j) will appear which is not permitted.

A simple example shown in Fig 1 is used to show how division rules are applied. One spike and two division rules are in neuron σ_3. Considering the two conditions mentioned in the above paragraph, 1). Neuron σ_3 has one spike a and the regular expressions of both two division rules are exactly $\{a\}$, where $a \in \{a\}$. Therefore, both of these two division rules meet the condition (1). 2). For rule $[a]_3 \to [\ ]_2 \parallel [\ ]_3$, the label 3 of the father neuron σ_3 corresponds to i in the normalization rule, and the label 2 and 3 of the two new neurons σ_2 and σ_3 corresponds to j and k in the normalization rule. Synapses (i, j), (j, i), (i, k), (k, i) cannot exist in the system. That is to say, (3, 2), (2, 3), (3, 3), (3, 3) cannot exist in this system. However, a synapse (2, 3) is in this system, therefore rule $[a]_3 \to [\ ]_2 \parallel [\ ]_3$ cannot be applied. Only rule $[a]_3 \to [\ ]_3 \parallel [\ ]_4$ meet the two conditions. The spike a in neuron σ_3 is consumed, neuron σ_3 is divided into two neurons σ_3 and σ_4, and two synapses (2, 3), (2, 4) going from neuron σ_2 to these two new neurons are established because there is a synapse going from neuron σ_2 to the father neuron σ_3 of the two new neurons (the inheritance of synapses). Because rules in this system are related to the labels of neurons, the new neuron σ_3 contains these two rules. The system is changed to Fig 2 after applying rule $[a]_3 \to [\ ]_3 \parallel [\ ]_4$.

If neuron σ_i has h spikes, and $a^h \in L(E)$, the neuron dissolution rule $[E]_i \to \delta$ can be applied. All h spikes in neuron σ_i are consumed and neuron σ_i is dissolved. All synapses going from/to neuron σ_i are dissolved, too.

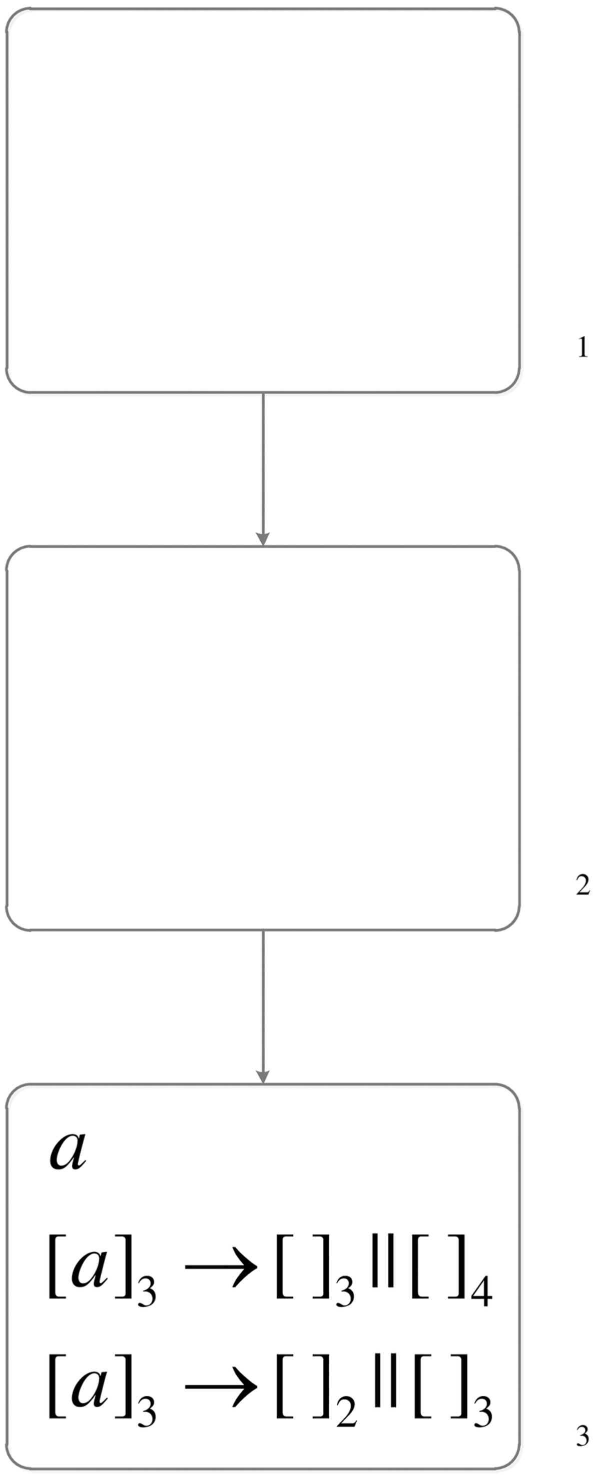
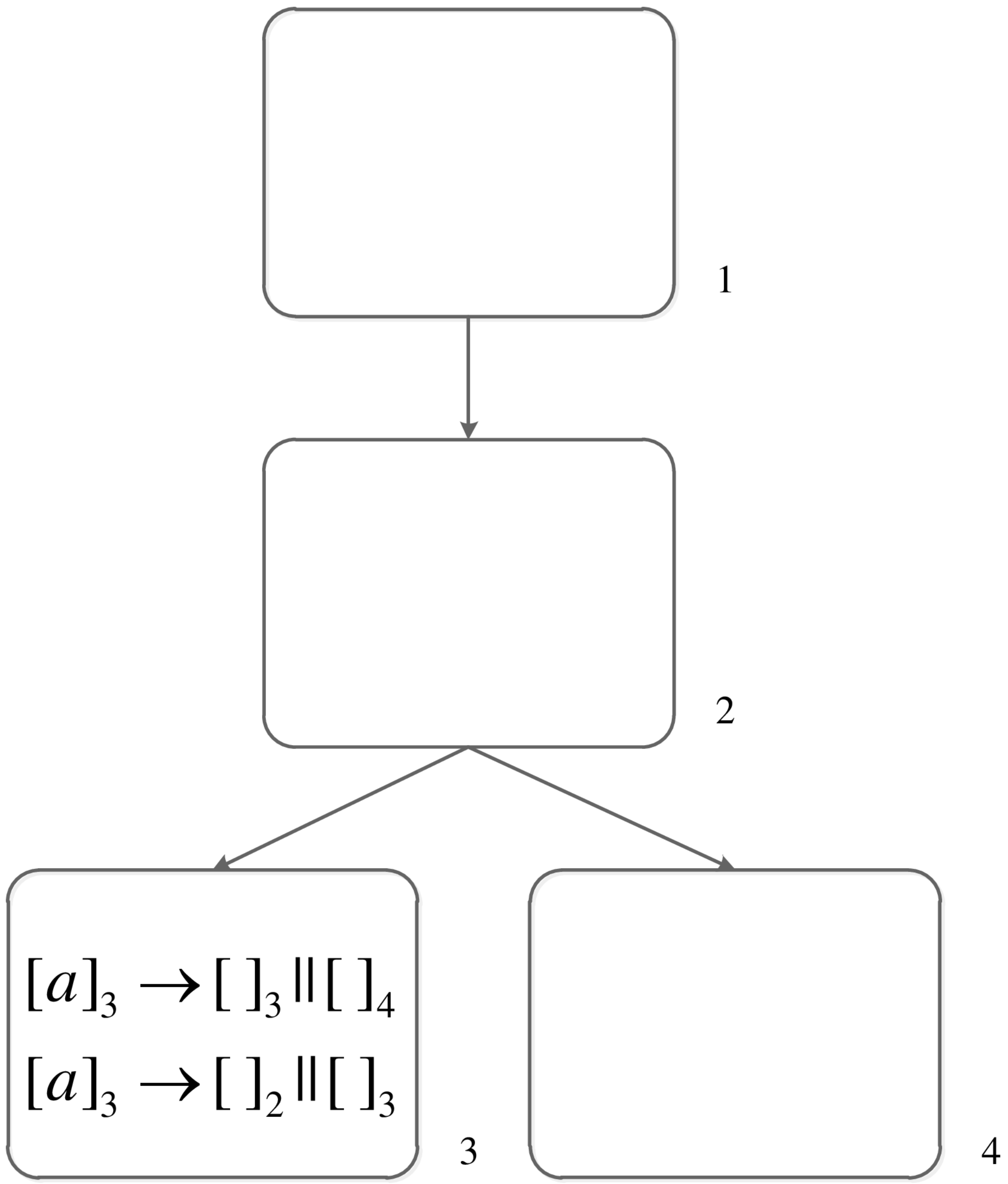

1
2
3
a
$[a]_3 \rightarrow [\ ]_3 \| [\ ]_4$
$[a]_3 \rightarrow [\ ]_2 \| [\ ]_3$

Fig 1. The SN P system before executing division rule.

doi:10.1371/journal.pone.0162882.g001



$[a]_3 \rightarrow [\ ]_3 \| [\ ]_4$
$[a]_3 \rightarrow [\ ]_2 \| [\ ]_3$
1
2
3
4

Fig 2. The SN P system after executing division rule.

doi:10.1371/journal.pone.0162882.g002

A simple example shown in Fig 3 is used to show how dissolution rules are applied. Neuron σ_1 has one spike a and the regular expression of the dissolution rule is exactly $\{a\}$, where $a \in \{a\}$. Then rule $[a]_1 \rightarrow \delta$ is applied and the system is changed to Fig 4 (Neuron σ_1 is dissolved, and synapse (1, 2) connected with neuron σ_1 is also dissolved.).

At each step, if only one rule in neuron σ_i can be applied, this rule must be applied; if two or more rules in neuron σ_i can be applied, one of these rules is applied non-deterministically. Rules are applied in a sequential manner in each neuron and in parallel between neurons.

The *configuration* of the system is described by the synapses connections, the spikes number in each neuron, and the state of each neuron (open or closed). By applying rules, the

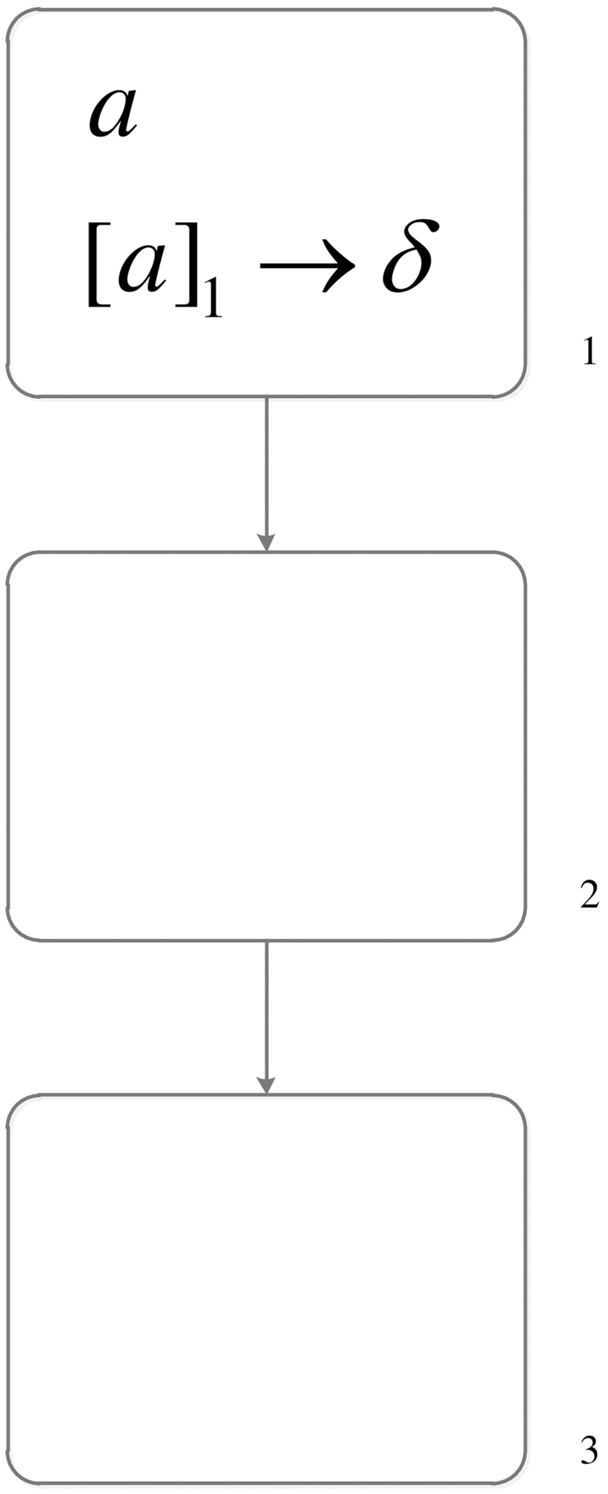


Fig 3. The SN P system before executing dissolution rule.

doi:10.1371/journal.pone.0162882.g003

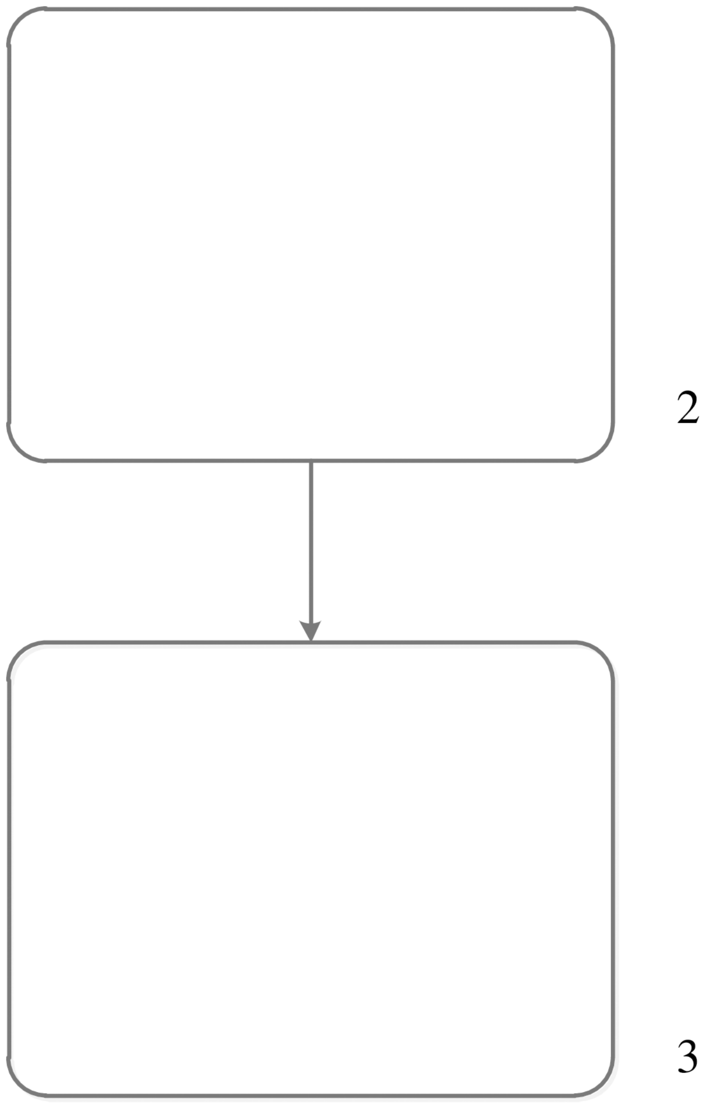


Fig 4. The SN P system after executing dissolution rule.

doi:10.1371/journal.pone.0162882.g004

configuration is *transformed* from one to the next one. The *transition* sequence starting from the initial configuration is called a *computation*, and a computation *halts* if it reaches a configuration where all neurons are open and no rule can be applied.

SN P systems can used to solve the decision problem I_X, Θ_X both in a semi-uniform way and in a uniform way, where I_X is a language over a finite alphabet and Θ_X is a total boolean function over I_X (The elements in I_X are instances.). In the semi-uniform way, a specified SN P system is constructed for each instance of a decision problem, in which the instance parameters are embedded in the SN P system. In the uniform way, a SN P system is constructed for all instances of a decision problem, in which the different instances parameters enter the SN P system as input spikes. The uniform solutions are preferred because they only relate to the structure of a problem.

The input of a SN P system is a spike train $a^{i_1} \cdot a^{i_2} \cdot \ldots \cdot a^{i_r}$, where $r \geq 1$, $i_j \geq 0$ for each $1 \leq j \leq r$, which means i_j spikes enter the system through input neuron σ_{in} at step j. Specially, $i_j = 0$ means no spike enters the system at step j.

2 A Uniform Solution to SAT Problem

SAT (the satisfiability of conjunctive normal form expression) problem is one of the most typical NP-complete problems. For a Boolean variable set $X = \{x_1, x_2, \ldots, x_n\}$, a literal l_i is x_i or $\neg x_i$ for $1 \leq i \leq n$. A clause C_i is a disjunction of literals $C_i = l_{n_1} \vee l_{n_2} \vee \ldots \vee l_{n_r}$, $1 \leq r \leq n$. A conjunctive normal form (CNF, for short) is a conjunction of clauses $C_1 \wedge C_2 \wedge \ldots \wedge C_m$. An assignment is a mapping $X \rightarrow \{0, 1\}$ from each variable x_i to its value (Value 1 represents true and value 0 represents false.). For example, $X = \{x_1, x_2, x_3\}$, the conjunctive normal form is $(x_1 \vee \neg x_2) \wedge (x_1 \vee x_3)$. The $x_1 \vee \neg x_2$ and $x_1 \vee x_3$ are the two clauses. The first clause contains two literals x_1 and $\neg x_2$, and the second clause contains two literals x_1 and x_3. If an assignment of $x_1, x_2, \ldots, x_n$ can be found, which makes at least one literal true in each clause and then makes all m clauses true, this SAT problem is satisfiable. Otherwise, this SAT problem is unsatisfiable [38]. In the above example, let $x_1 = x_2 = x_3 = 1$, the value of the conjunctive normal form is $(1 \vee 0) \wedge (1 \vee 0) = 1 \wedge 1 = 1$. Therefore, the SAT problem is satisfiable.

The formal definition of SAT problem is as follows.

Problem 1. NAME: SAT.

- INSTANCE: a set of clauses $C = \{C_1, C_2, \ldots, C_m\}$, which is built on a Boolean variable set $X = \{x_1, x_2, \ldots, x_n\}$.
- QUESTION: is there an assignment of Boolean variables $x_1, x_2, \ldots, x_n$ that can make the value of all clauses true?

$SAT(n, m)$ denotes the set of all instances of the SAT problem having n variables and m clauses. In this section, a uniform solution working in a deterministic way is constructed by DDSN P system, which can solve all $SAT(n, m)$ problems in linear time.

The instance parameters need to enter a SN P system, therefore the clauses need to be encoded as spikes form. Each clause contains either x_j, or $\neg x_j$, or none of these two. Different numbers of spikes are introduced into the system to distinguish these three situations.

$$\alpha_{i,j} = \begin{cases} a, & \text{if } x_j \text{ occurs in } C_i; \\ a^2, & \text{if } \neg x_j \text{ occurs in } C_i; \\ a^0, & \text{otherwise.} \end{cases}$$

A clause is represented by $\alpha_{i,1} \cdot \alpha_{i,2} \cdot \ldots \cdot \alpha_{i,n}$ in this way. For instance, a clause $\neg x_1 \bigvee x_2 \bigvee x_3$ is represented by $a^2 \cdot a \cdot a$.

In order to generate the necessary workspace before computing, a spike train $(a^0 \cdot)^{2n}$ is introduced into the front of each spike train.

The formal definition of DDSN P systems for $SAT(n, m)$ problems (shown in Fig 5) is as follows.

$$\Pi_{n,m} = (O, H, syn, n_1, n_2, \ldots, n_{3n+5}, R, in, out),$$

where:

1. $O = \{a\}$;
2. $H = \{0, 1, 2, 3, d, in_{x_i}, Cx_i1, Cx_i0, o_{t_1, t2, \ldots, tn}\}(i, t_1, t_2, \ldots, t_n = 1,2)$;
3. $syn = \{(3, 2), (2, 1), (1, 2), (1, 0), (3, d)\}$
$\bigcup\{(d, in_{x_i}) | i = 1, 2, \ldots, n\} \bigcup\{(in_{x_i}, Cx_i1), (in_{x_i}, Cx_i0) | i = 1, 2, \ldots, n\}$
$\bigcup\{(Cx_11, o_1), (Cx_i1, o_{t_1, t(i-1)1}) | i = 2, 3, \ldots, n\}$
$\bigcup\{(Cx_10, o_0), (Cx_i0, o_{t_1, t(i-1)0}) | i = 2, 3, \ldots, n\}$;

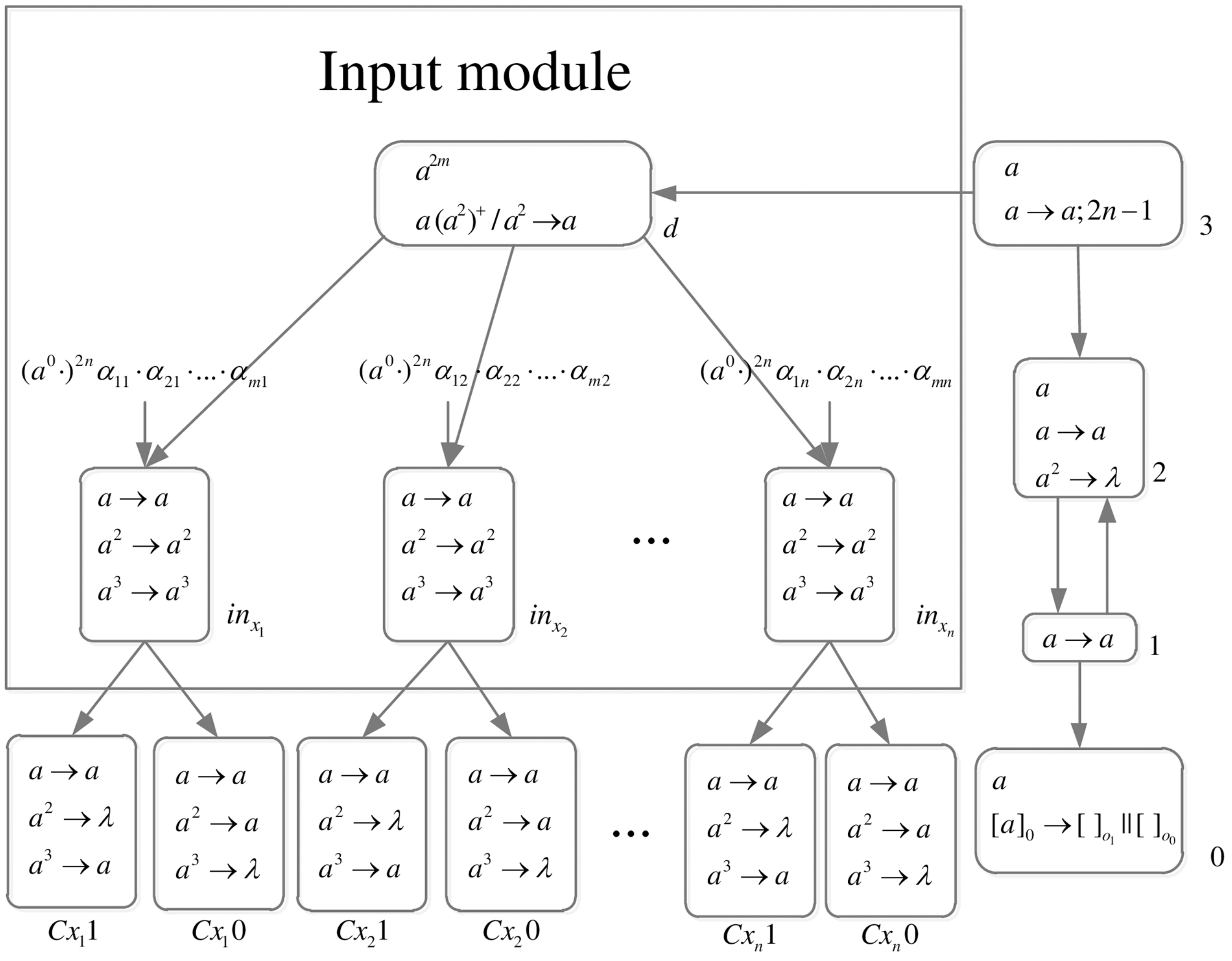


Fig 5. The initial system $\Pi_{n,m}$.

doi:10.1371/journal.pone.0162882.g005

4. $n_0 = 1$, $n_2 = 1$, $n_3 = 1$, $n_d = 2m$, the number of spikes in other neurons is zero;
5. $in = \sigma_{in_{xi}}$, $out = \sigma_{o_{t1,\; t2,\; \ldots,\; tn}}(i, t_1, t_2, \ldots, t_n = 0, 1)$;
6. **firing rule:**
   $[a \to a]_i$, $i = 1, 2$
   $[a \to a; 2n - 1]_3$
   $[a(a^2)^+/a^2 \to a]_d$
   $[a \to a]_{in_{xi}}$, $i = 1, 2, \ldots, n$
   $[a^2 \to a^2]_{in_{xi}}$, $i = 1, 2, \ldots, n$
   $[a^3 \to a^3]_{in_{xi}}$, $i = 1, 2, \ldots, n$
   $[a \to a]_{Cx_i1}$, $i = 1, 2, \ldots, n$
   $[a^3 \to a]_{Cx_i1}$, $i = 1, 2, \ldots, n$
   $[a \to a]_{Cx_i0}$, $i = 1, 2, \ldots, n$
   $[a^2 \to a]_{Cx_i0}$, $i = 1, 2, \ldots, n$
   forgetting rule:
   $[a^2 \to \lambda]_2$

$[a^2 \rightarrow \lambda]_{Cx_i1}$, $i = 1, 2, \ldots, n$
$[a^3 \rightarrow \lambda]_{Cx_i0}$, $i = 1, 2, \ldots, n$
$[a \rightarrow \lambda]_{o_{t1\ t2\ldots tn}}$, $t_1, t_2, \ldots, t_n = 0, 1$
$[a^2 \rightarrow \lambda]_{o_{t1\ t2\ldots tn}}$, $t_1, t_2, \ldots, t_n = 0, 1$
...
$[a^{n-1} \rightarrow \lambda]_{o_{t1\ t2\ldots tn}}$, $t_1, t_2, \ldots, t_n = 0, 1$
neuron division rule:
$[a]_0 \rightarrow [\ ]_{o_1} \,||\, [\ ]_{o_0}$
$[a]_{o_{t1}} \rightarrow [\ ]_{o_{t11}} \,||\, [\ ]_{o_{t10}}$, $t_1 = 0, 1$
$[a]_{o_{t1\ t2}} \rightarrow [\ ]_{o_{t1\ t2_1}} \,||\, [\ ]_{o_{t1\ t2_0}}$, $t_1, t_2 = 0, 1$
...
$[a]_{o_{t1\ t2\ldots tn-1}} \rightarrow [\ ]_{o_{t1\ t2\ldots tn-1_1}} \,||\, [\ ]_{o_{t1\ t2\ldots tn-1_0}}$, $t_1, t_2, \ldots, t_{n-1} = 0, 1$
neuron dissolution rule:
$[a^n]_{o_{t1\ t2\ldots tn}} \rightarrow \delta$, $t_1, t_2, \ldots, t_n = 0, 1$.

Computation starts when spike trains enter the system through input neurons $\sigma_{in_{x1}}$, $\sigma_{in_{x2}}$, ..., $\sigma_{in_{xn}}$, respectively. Neuron σ_0 and its children neurons need $2n$ steps to generate 2^n neurons (workspace) to enumerate all assignments of variables by applying neuron division rules (One neuron represents one assignment of variables.), therefore $(a^0 \cdot)^{2n}$ are added to the front of each spike train.

Generation Stage: At step one, neuron σ_0 has one spike, the division rule $[a]_0 \rightarrow [\ ]_{o_1} \,||\, [\ ]_{o_0}$ is applied to generate neurons σ_{o_1} and σ_{o_0}, which means an assignment in regard to x_1 has two choices: 1 or 0. Synapses $(1, o_1)$ and $(1, o_0)$ are established through the inheritance of synapse (1, 0), and synapses (Cx_11, o_1) and (Cx_10, o_0) are established through synapse dictionary *syn*. Synapse (Cx_11, o_1) establishes a channel between the input and the assignment including $x_1 = 1$; synapse (Cx_10, o_0) establishes a channel between the input and the assignment including $x_1 = 0$. At the same time, auxiliary neuron σ_2 has one spike, rule $a \rightarrow a$ is applied and one spike is emitted to neuron σ_1; auxiliary neuron σ_3 has one spike, rule $a \rightarrow a; 2n - 1$ is applied and one spike will be emitted to neurons σ_2 and σ_d at step $2n$. The system after step one is shown in Fig 6.

At step two, neuron σ_1 has one spike, the firing rule $a \rightarrow a$ is applied, and one spike is emitted to neurons σ_2, σ_{o_1} and σ_{o_0}.

At step three, each of neurons σ_{o_1} and σ_{o_0} has one spike, the division rule $[a]_{o_{t1}} \rightarrow [\ ]_{o_{t11}} \,||\, [\ ]_{o_{t10}}(t_1 = 1, 0)$ is applied to generate neurons $\sigma_{o_{11}}$, $\sigma_{o_{10}}$, $\sigma_{o_{01}}$ and $\sigma_{o_{00}}$, which means an assignment in regard to x_1 and x_2 has four choices: 11, 10, 01, 00. Synapses $(1, o_{11})$, $(1, o_{10})$, $(1, o_{01})$ and $(1, o_{00})$ are established through the inheritance of synapses $(1, o_1)$, $(1, o_0)$; synapses (Cx_11, o_{11}), (Cx_11, o_{10}), (Cx_10, o_{01}) and (Cx_11, o_{00}) are established through the inheritance of synapses (Cx_11, o_1), (Cx_10, o_0); synapses (Cx_21, o_{11}), (Cx_21, o_{01}), $S(Cx_20, o_{10})$ and (Cx_20, o_{00}) are established through synapse dictionary *syn*. Synapses (Cx_11, o_{11}) and (Cx_11, o_{10}) establish channels between the input and the assignments including $x_1 = 1$; synapses (Cx_10, o_{01}) and (Cx_10, o_{00}) establish channels between the input and the assignments including $x_1 = 0$; synapses (Cx_21, o_{11}) and (Cx_21, o_{01}) establish channels between the input and the assignments including $x_2 = 1$; synapses (Cx_20, o_{10}) and (Cx_20, o_{00}) establish channels between the input and the assignments including $x_2 = 0$. At the same time, auxiliary neuron σ_2 has one spike, rule $a \rightarrow a$ is applied and one spike is emitted to neuron σ_1. The system after step three is shown in Fig 7.

Similar process repeats. At step $2n - 1$, 2^n neurons labeled $o_{t_1\ t2\ldots tn}(t_1, t_2, \ldots, t_n = 0, 1)$ are generated. The system after step $2n - 1$ is shown in Fig 8.

At step $2n$, each neuron $\sigma_{o_{t1\ t2\ldots tn}}$ receives one spike emitted from neuron σ_1 which will be deleted at the next step by the forgetting rule $[a \rightarrow \lambda]_{o_{t1\ t2\ldots tn}}$. Neuron σ_2 receives two spikes

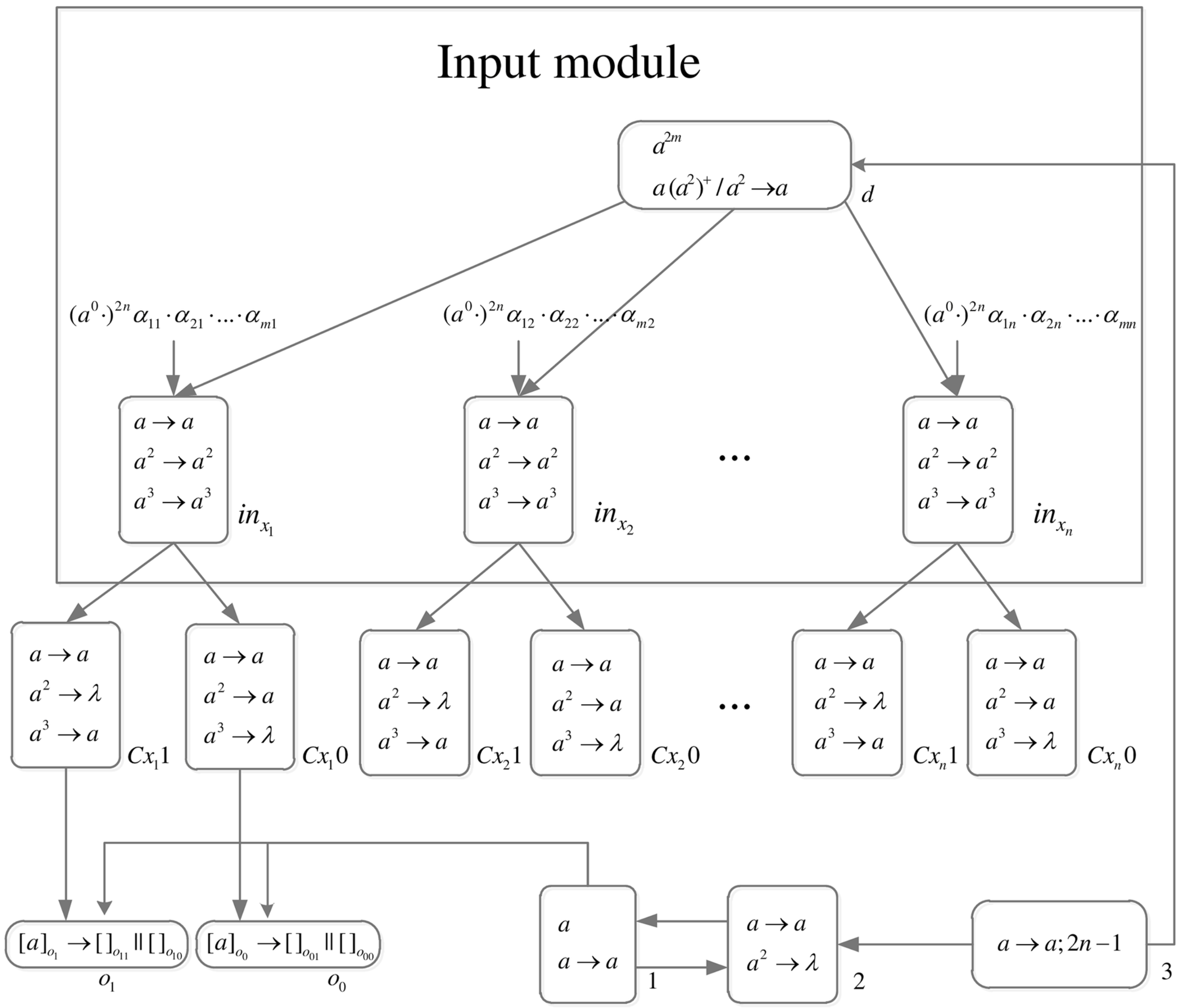

Input module
a^{2m}
$a(a^2)^+/a^2 \to a$
d
$(a^0\cdot)^{2n}\alpha_{11}\cdot\alpha_{21}\cdot\ldots\cdot\alpha_{m1}$
$(a^0\cdot)^{2n}\alpha_{12}\cdot\alpha_{22}\cdot\ldots\cdot\alpha_{m2}$
$(a^0\cdot)^{2n}\alpha_{1n}\cdot\alpha_{2n}\cdot\ldots\cdot\alpha_{mn}$
$a\to a$, $a^2\to a^2$, $a^3\to a^3$
in_{x_1}
in_{x_2}
in_{x_n}
$a\to a$, $a^2\to\lambda$, $a^3\to a$
$a\to a$, $a^2\to a$, $a^3\to\lambda$
$Cx_1 1$
$Cx_1 0$
$Cx_2 1$
$Cx_2 0$
$Cx_n 1$
$Cx_n 0$
$[a]_{o_1}\to[\,]_{o_{11}}\|[\,]_{o_{10}}$
$[a]_{o_0}\to[\,]_{o_{01}}\|[\,]_{o_{00}}$
o_1
o_0
a, $a\to a$
1
$a\to a$, $a^2\to\lambda$
2
$a\to a;2n-1$
3

Fig 6. The system $\Pi_{n,m}$ after step one.

doi:10.1371/journal.pone.0162882.g006

(One is emitted from neuron σ_1, and another one is emitted from σ_3), the forgetting rule $a^2 \to \lambda$ is applied at step $2n + 1$, and no spike will be emitted to neuron σ_1 later. At the same time, neuron σ_d receives one spike emitted from σ_3. The system after step $2n$ is shown in Fig 9.

Input Stage: At step $2n + 1$, the first clause of the conjunctive normal form expression enters the system through input neurons $\sigma_{in_{xi}}$, $i = 1, 2, \ldots, n$. The literal in regard to x_1 enters neuron $\sigma_{in_{x1}}$; the literal in regard to x_2 enters neuron $\sigma_{in_{x2}}$; . . . the literal in regard to x_n enters neuron $\sigma_{in_{xn}}$. At the same time, one spike is emitted to neuron $\sigma_{in_{xi}}$ from neuron σ_d.

At step $2n + 2$, the spikes in neuron $\sigma_{in_{xi}}$ are replicated, and are emitted to neurons σ_{Cx_i1} and σ_{Cx_i0}.

At step $2n + 3$, different rules are applied according to the number of spikes in neurons σ_{Cx_i1} and σ_{Cx_i0}.

For neuron σ_{Cx_i1}:

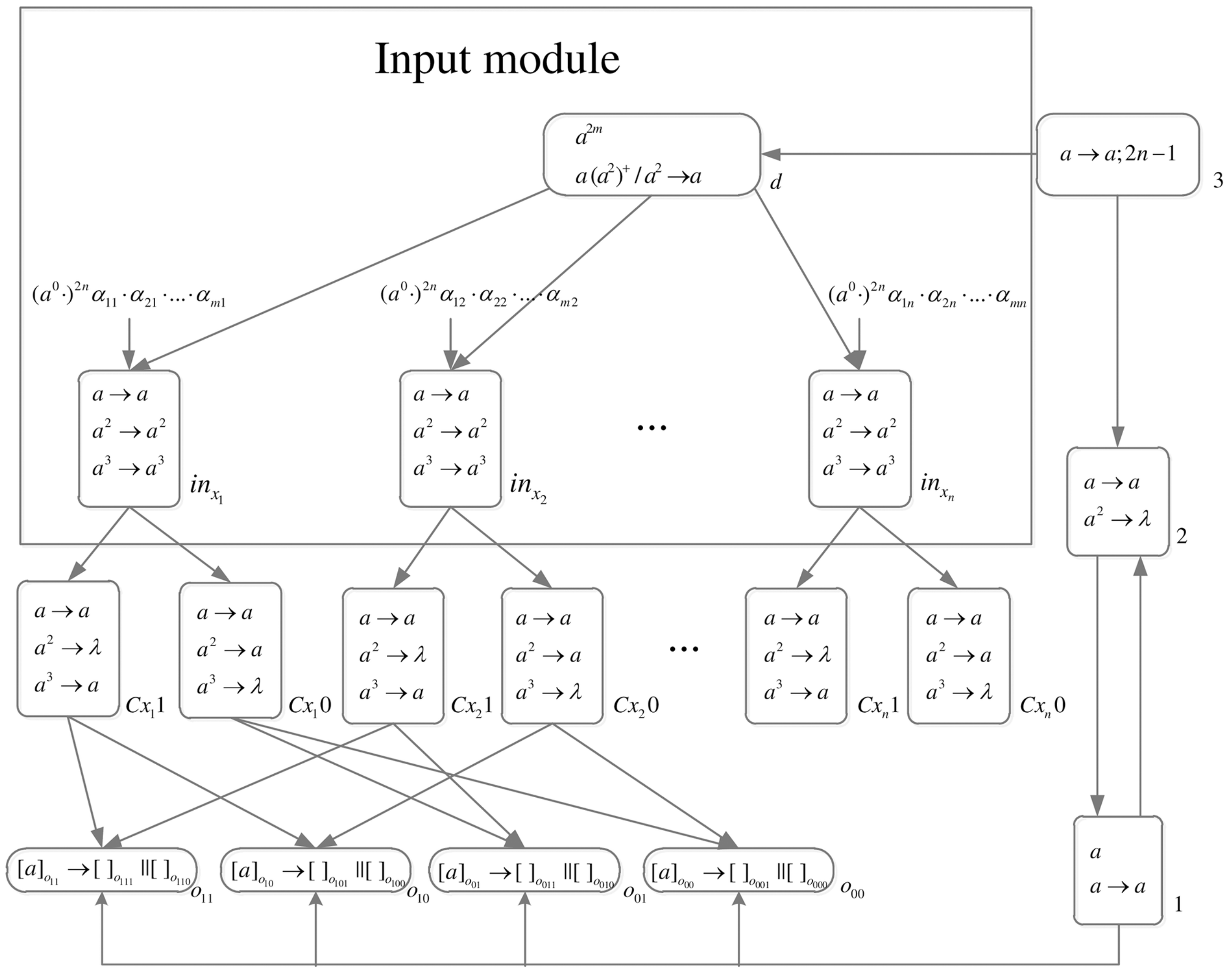

Input module
a^{2m}
$a(a^2)^+/a^2 \to a$
d
$a \to a; 2n-1$
3
$(a^0\cdot)^{2n}\alpha_{11}\cdot\alpha_{21}\cdot\ldots\cdot\alpha_{m1}$
$(a^0\cdot)^{2n}\alpha_{12}\cdot\alpha_{22}\cdot\ldots\cdot\alpha_{m2}$
$(a^0\cdot)^{2n}\alpha_{1n}\cdot\alpha_{2n}\cdot\ldots\cdot\alpha_{mn}$
$a \to a$ $a^2 \to a^2$ $a^3 \to a^3$
in_{x_1}
in_{x_2}
in_{x_n}
$a \to a$ $a^2 \to \lambda$
2
$a \to a$ $a^2 \to \lambda$ $a^3 \to a$
$a \to a$ $a^2 \to a$ $a^3 \to \lambda$
$Cx_1 1$
$Cx_1 0$
$Cx_2 1$
$Cx_2 0$
$Cx_n 1$
$Cx_n 0$
$[a]_{o_{11}} \to [\ ]_{o_{111}} \| [\ ]_{o_{110}}$
o_{11}
$[a]_{o_{10}} \to [\ ]_{o_{101}} \| [\ ]_{o_{100}}$
o_{10}
$[a]_{o_{01}} \to [\ ]_{o_{011}} \| [\ ]_{o_{010}}$
o_{01}
$[a]_{o_{00}} \to [\ ]_{o_{001}} \| [\ ]_{o_{000}}$
o_{00}
a $a \to a$
1

Fig 7. The system $\Pi_{n,m}$ after step three.

doi:10.1371/journal.pone.0162882.g007

- If one spike is in neuron σ_{Cx_i1}, which means neither x_i nor $\neg x_i$ is in the clause, rule $a \to a$ is applied. One spike is emitted to neurons having synapses going from neuron σ_{Cx_i1} to them. It aims to show that $x_i = 1$ makes no contribution to let the clause true.
- If two spikes are in neuron σ_{Cx_i1}, which means x_i is in the clause, rule $a^2 \to \lambda$ is applied. These two spikes are deleted. It aims to show that $x_i = 1$ makes contribution to let the clause true.
- If three spikes are in neuron σ_{Cx_i1}, which means $\neg x_i$ is in the clause, rule $a^3 \to a$ is applied. One spike is emitted to neurons having synapses going from neuron σ_{Cx_i1} to them. It aims to show that $x_i = 1$ makes no contribution to let the clause true.

For neuron σ_{Cx_i0}:

- If one spike is in neuron σ_{Cx_i0}, which means neither x_i nor $\neg x_i$ is in the clause, rule $a \to a$ is applied. One spike is emitted to neurons having synapses going from neuron σ_{Cx_i0} to them. It aims to show that $x_i = 0$ makes no contribution to let the clause true.

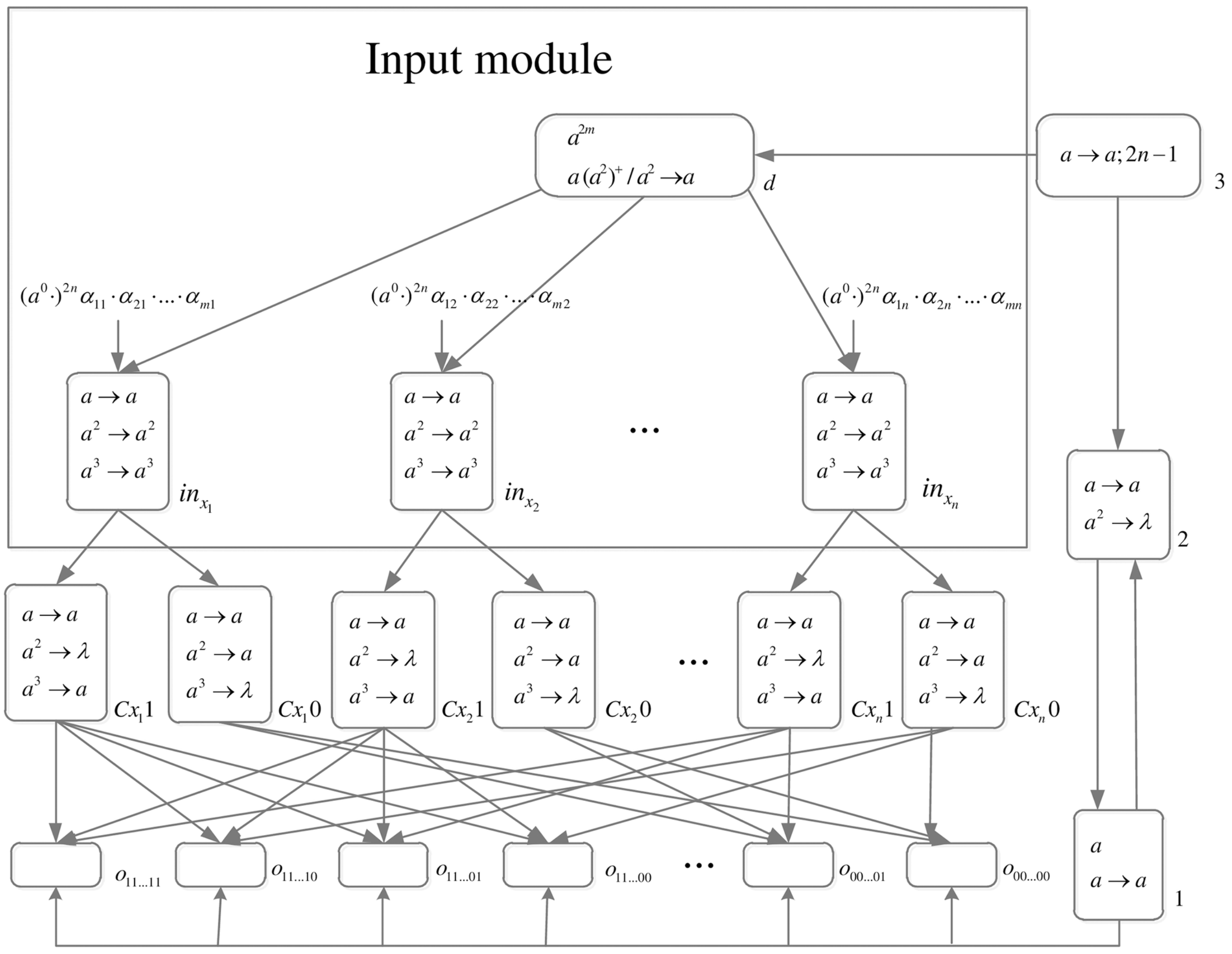

Input module
a^{2m}
$a(a^2)^+/a^2 \to a$
d
$a \to a; 2n-1$
3
$(a^0\cdot)^{2n}\alpha_{11}\cdot\alpha_{21}\cdot\ldots\cdot\alpha_{m1}$
$(a^0\cdot)^{2n}\alpha_{12}\cdot\alpha_{22}\cdot\ldots\cdot\alpha_{m2}$
$(a^0\cdot)^{2n}\alpha_{1n}\cdot\alpha_{2n}\cdot\ldots\cdot\alpha_{mn}$
$a \to a$, $a^2 \to a^2$, $a^3 \to a^3$
in_{x_1}
in_{x_2}
in_{x_n}
$a \to a$, $a^2 \to \lambda$
2
$a \to a$, $a^2 \to \lambda$, $a^3 \to a$
$Cx_1 1$
$a \to a$, $a^2 \to a$, $a^3 \to \lambda$
$Cx_1 0$
$Cx_2 1$
$Cx_2 0$
$Cx_n 1$
$Cx_n 0$
$o_{11\ldots11}$
$o_{11\ldots10}$
$o_{11\ldots01}$
$o_{11\ldots00}$
$o_{00\ldots01}$
$o_{00\ldots00}$
a
$a \to a$
1

Fig 8. The system $\Pi_{n,m}$ after step $2n-1$.

doi:10.1371/journal.pone.0162882.g008

- If two spikes are in neuron σ_{Cx_i0}, which means x_i is in the clause, rule $a^2 \to a$ is applied. One spike is emitted to neurons having synapses going from neuron σ_{Cx_i0} to them. It aims to show that $x_i = 0$ makes no contribution to let the clause true.

- If three spikes are in neuron σ_{Cx_i0}, which means $\neg x_i$ is in the clause, rule $a^3 \to \lambda$ is applied. These three spikes are deleted. It aims to show that $x_i = 0$ makes contribution to let the clause true.

Satisfiability Stage: Each neuron $\sigma_{o_{t_1 t_2 \ldots t_n}}(t_1, t_2, \ldots, t_n = 0, 1)$ receives zero or more spikes at step $2n + 3$. If one neuron $\sigma_{o_{t_1 t_2 \ldots t_n}}$ receives n spikes which means the clause contains n literals that make no contribution to let the clause true, the dissolution rule $[a^n]_{o_{t_1 t_2 \ldots t_n}} \to \delta$ is applied at step $2n + 4$ to dissolve this neuron (The value of the first clause is false, therefore this assignment is not the answer to this SAT problem.). Otherwise, at least one literal is true in this assignment and this assignment is reserved to check the next clause.

Due to m clauses are in a SAT problem, the satisfiability checking stage lasts for $m + 3$ steps. If some neurons $\sigma_{o_{t_1 t_2 \ldots t_n}}$ are still in the system at step $2n + m + 3$, the labels of these neurons $\sigma_{o_{t_1 t_2 \ldots t_n}}$ are all solutions to this SAT problem, i.e., this SAT problem is satisfiable. Otherwise, this SAT problem is unsatisfiable.

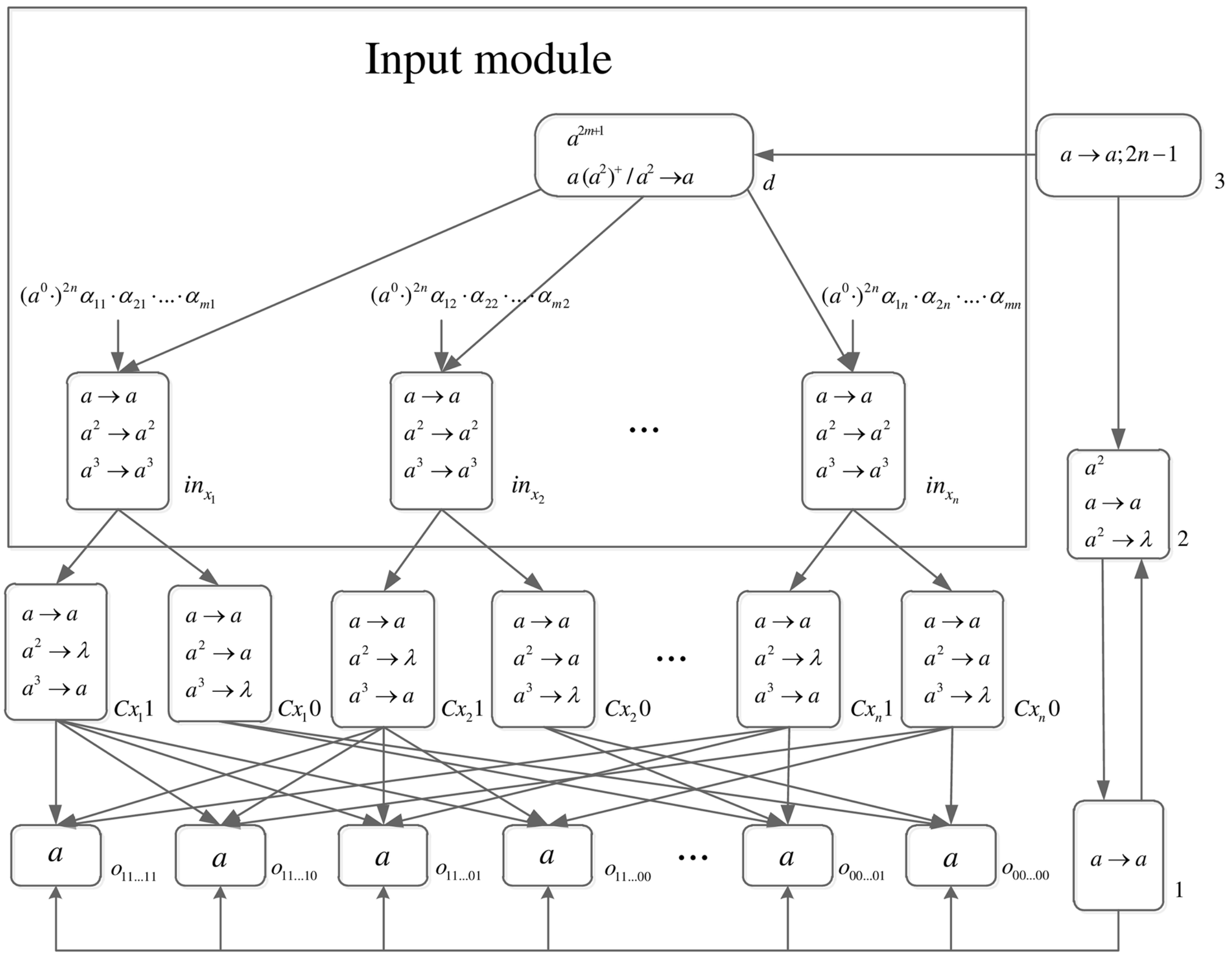


Fig 9. The system $\Pi_{n,m}$ after step 2n.

doi:10.1371/journal.pone.0162882.g009

It can be seen that any $SAT(n, m)$ problem can be solved in linear time, and all solutions can be obtained through this system.

Some steps comparison results between our solution and other solutions, which use the neuron division to solve the NP-complete problems, are shown in Table 1.

Considering a SAT problem $SAT\ (3, 3)$: $(x_1 \bigvee x_2) \bigwedge (\neg x_2 \bigvee x_3) \bigwedge (\neg x_1 \bigvee x_2 \bigvee x_3)$, the DDSN P system $\Pi_{3,3}$ is used to solve it. After 12 computational steps, neurons $\sigma_{o_{111}}$, $\sigma_{o_{101}}$ and $\sigma_{o_{011}}$ are remaining which shows that $\{x_1 = true, x_2 = true, x_3 = true\}$, $\{x_1 = true, x_2 = false, x_3 = true\}$ and $\{x_1 = false, x_2 = true, x_3 = true\}$ are all solutions to this SAT problem.

The SN P system with neuron division and budding and the SN P system with neuron division need 21 steps and 26 steps to judge this problem has solutions, respectively, while our DDSN P system need only 12 steps.

SAT problems with different sizes ($1 \leq n, m \leq 50$) are solved using the three systems in Table 1 and the computational steps of each system are shown in Figs 10, 11 and 12 by MATLAB R2014a. As can be seen from these figures, the computational steps of DDSN P system are stable and much fewer, especially when the problem size is larger.

Table 1. Steps comparison results of some uniform solutions to SAT problem.

solution	step complexity (steps)	determinism or nondeterminism
SN P systems with neuron division and budding [25]	$2n + mn + 6$	determinism
SN P systems with neuron division [26]	$4n + mn + 5$	determinism
DDSN P systems	$2n + m + 3$	determinism

doi:10.1371/journal.pone.0162882.t001

3 A Uniform Solution to Subset Sum Problem

Subset Sum problem is one of the most typical NP-complete problems. The formal definition of it is as follows [38].

Problem 2. NAME: SUBSET SUM.

- INSTANCE: a set of positive integers $X = \{x_1, x_2, \ldots, x_n\}$ and a positive integer S.
- QUESTION: is there a subset $B \subseteq X$ that $\sum_{b \in B} b = S$?

Subset Sum problem (n) denotes the set of all instances of the Subset Sum problem having n integers. In this section, a uniform solution working in a deterministic way is constructed by DDSN P system, which can solve all *Subset Sum problem* (n) problems in linear time.

An integer is represented by corresponding number of spikes. In order to generate necessary workspace before computing, a spike train $(a^0 \cdot)^{2n}$ is introduced into the front of each spike train.

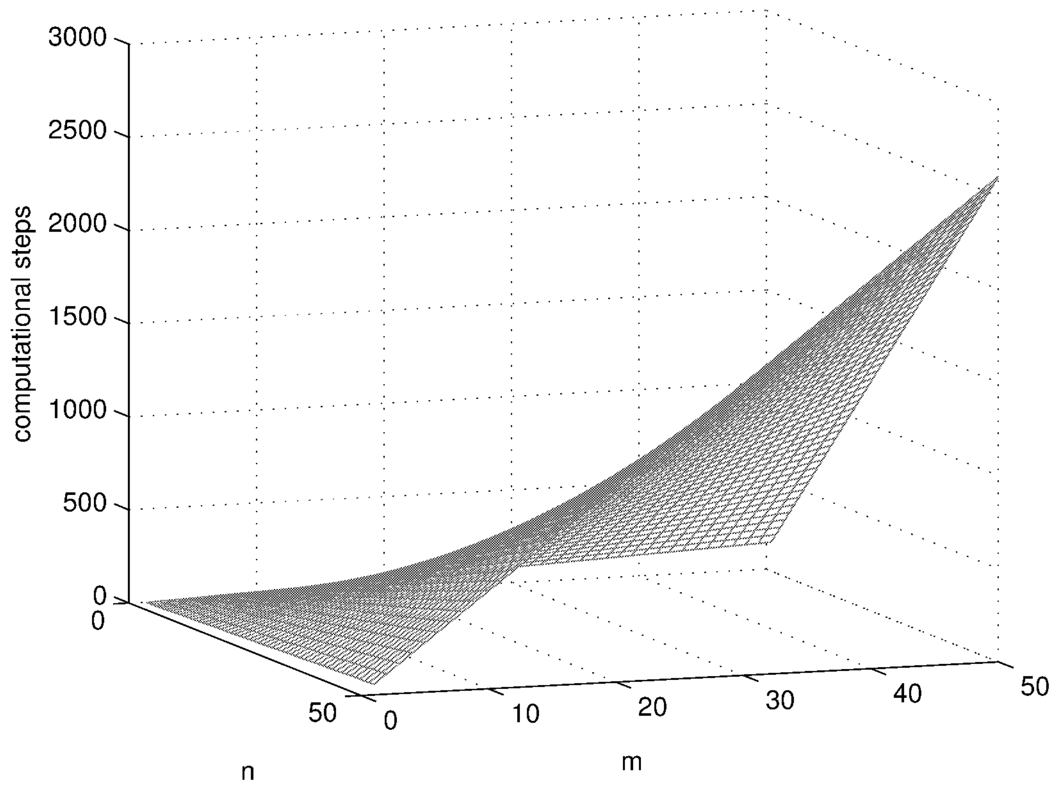


Fig 10. The computational steps of SN P system with neuron division and budding solving SAT problem.

doi:10.1371/journal.pone.0162882.g010

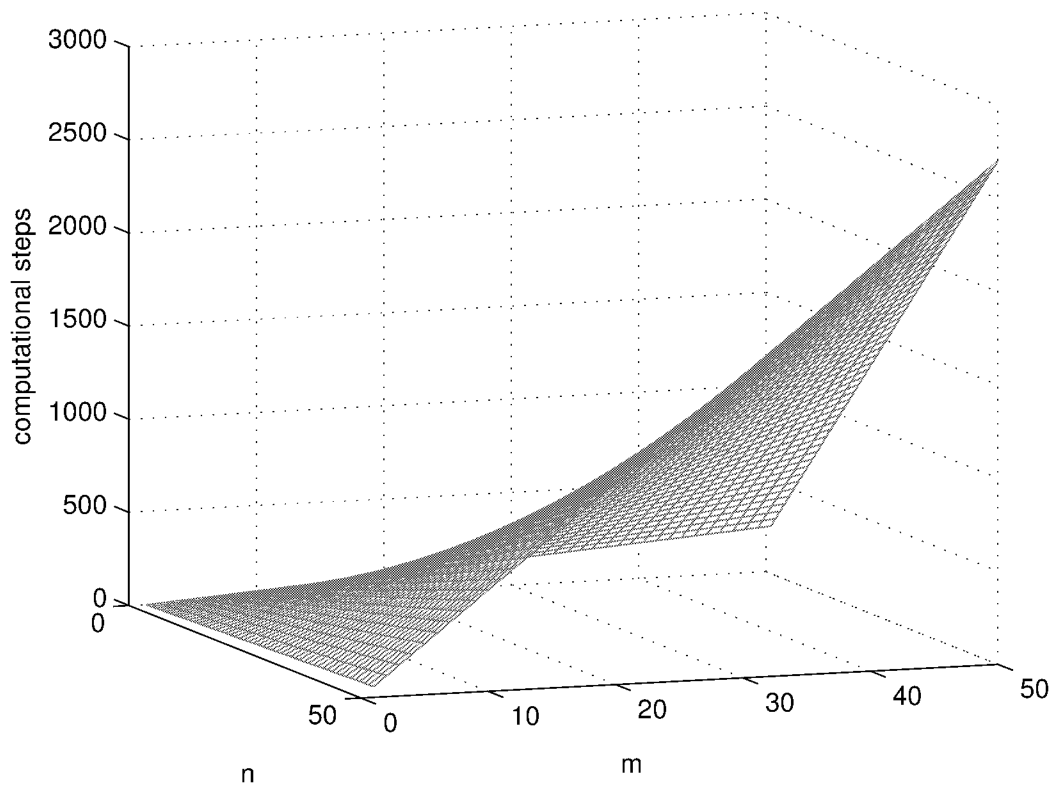


Fig 11. The computational steps of SN P system with neuron division solving SAT problem.

doi:10.1371/journal.pone.0162882.g011

The formal definition of DDSN P Systems for *Subset Sum problem* (n) (shown in Fig 13) is as follows.

$$\Pi_n = (O, H, syn, n_1, n_2, \ldots, n_{3n+6}, R, in, out),$$

where:

1. $O = \{a\}$;
2. $H = \{0, 1, 2, 3, 4, s, in_i, d_{i1}, d_{i2}\}(i = 1, 2, \ldots, n)$;
3. $syn = \{(3, 2), (2, 1), (1, 2), (1, 0), (4, s), (s, 0)\}$
   $\bigcup\{(in_i, d_{i1}), (in_i, d_{i2}), (d_{i1}, 4) | i = 1, 2, \ldots, n\}$
   $\bigcup\{(d_{1,2}, o_1)\}\bigcup\{(d_{i2}, o_{t_1, \ldots, t(i-1)1}) | i = 2, 3, \ldots, n\}$;
4. $n_0 = 1, n_2 = 1, n_3 = 1$, the number of spikes in other neurons is zero;
5. $in = \sigma_{in_i}, \sigma_s, out = \sigma_{o_{t1, t^2, \ldots, t^n}}(i, t_1, t_2, \ldots, t_n = 0, 1)$;
6. **firing rule:**
   $[a \to a]_i, i = 1, 2$
   $[a \to a; 2n - 1]_3$
   $[a^3(a^3)^+/a^3 \to a^3]_{in_i}, i = 1, 2, \ldots, n$
   $[a^3 \to a]_{in_i}, i = 1, 2, \ldots, n$
   $[a \to a]_{d_{i1}}, i = 1, 2, \ldots, n$
   $[a^3 \to a^3]_{d_{i2}}, i = 1, 2, \ldots, n$
   $[a^n \to a]_4$

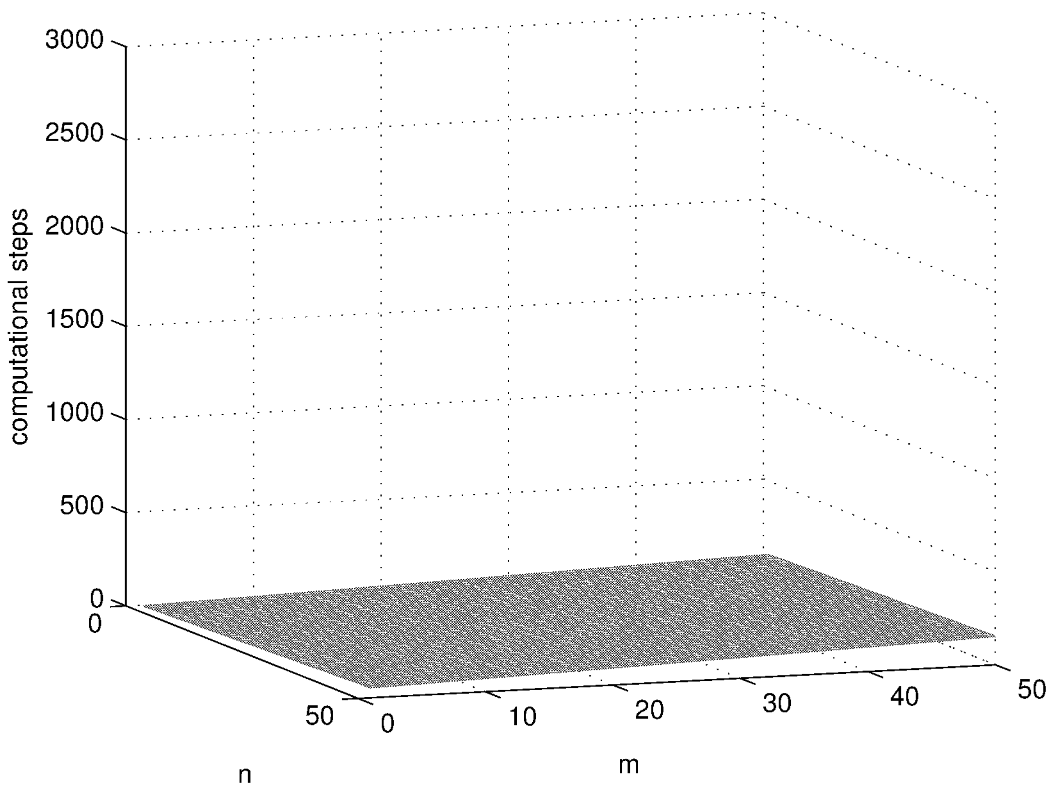


Fig 12. The computational steps of DDSN P system solving SAT problem.

doi:10.1371/journal.pone.0162882.g012

$[a(a^2)^+/a^2 \rightarrow a^2]_s$
$[a \rightarrow a]_s$
forgetting rule:
$[a^2 \rightarrow \lambda]_2$
$[a^3 \rightarrow \lambda]_{d_{i1}}, i = 1, 2, \ldots, n$
$[a \rightarrow \lambda]_{d_{i2}}, i = 1, 2, \ldots, n$
$[a^2(a^3)^+/a^5 \rightarrow \lambda]_{o_{t_1t_2\ldots t_n}}, t_1, t_2, \ldots, t_n = 0, 1$
$[a \rightarrow \lambda]_{o_{t_1t_2\ldots t_n}}, t_1, t_2, \ldots, t_n = 0, 1$
neuron division rule:
$[a]_0 \rightarrow [\ ]_{o_1} \,||\, [\ ]_{o_0}$
$[a]_{o_{t_1}} \rightarrow [\ ]_{o_{t_11}} \,||\, [\ ]_{o_{t_10}}, t_1 = 0, 1$
$[a]_{o_{t_1t_2}} \rightarrow [\ ]_{o_{t_1t_21}} \,||\, [\ ]_{o_{t_1t_20}}, t_1, t_2 = 0, 1$
$\ldots$
$[a]_{o_{t_1t_2\ldots t_{n-1}}} \rightarrow [\ ]_{o_{t_1t_2\ldots t_{n-1}1}} \,||\, [\ ]_{o_{t_1t_2\ldots t_{n-1}0}}, t_1, t_2, \ldots, t_{n-1} = 0, 1$
neuron dissolution rule:
$[a(a^3)^+]_{o_{t_1t_2\ldots t_n}} \rightarrow \delta, t_1, t_2, \ldots, t_n = 0, 1$
$[a^2]_{o_{t_1t_2\ldots t_n}} \rightarrow \delta, t_1, t_2, \ldots, t_n = 0, 1.$

Computation starts when spike trains enter the system thrgouth input neurons $\sigma_{in_1}, \sigma_{in_2}, \ldots, \sigma_{in_n}$ and σ_s, respectively. Neuron σ_0 and its children neurons need $2n$ steps to generate 2^n neurons (workspace) to enumerate all subsets of $x_1, x_2, \ldots, x_n$ by applying neuron division rules (One neuron represents one subset.), therefore $(a^0 \cdot)^{2n}$ are added to the front of each spike train.

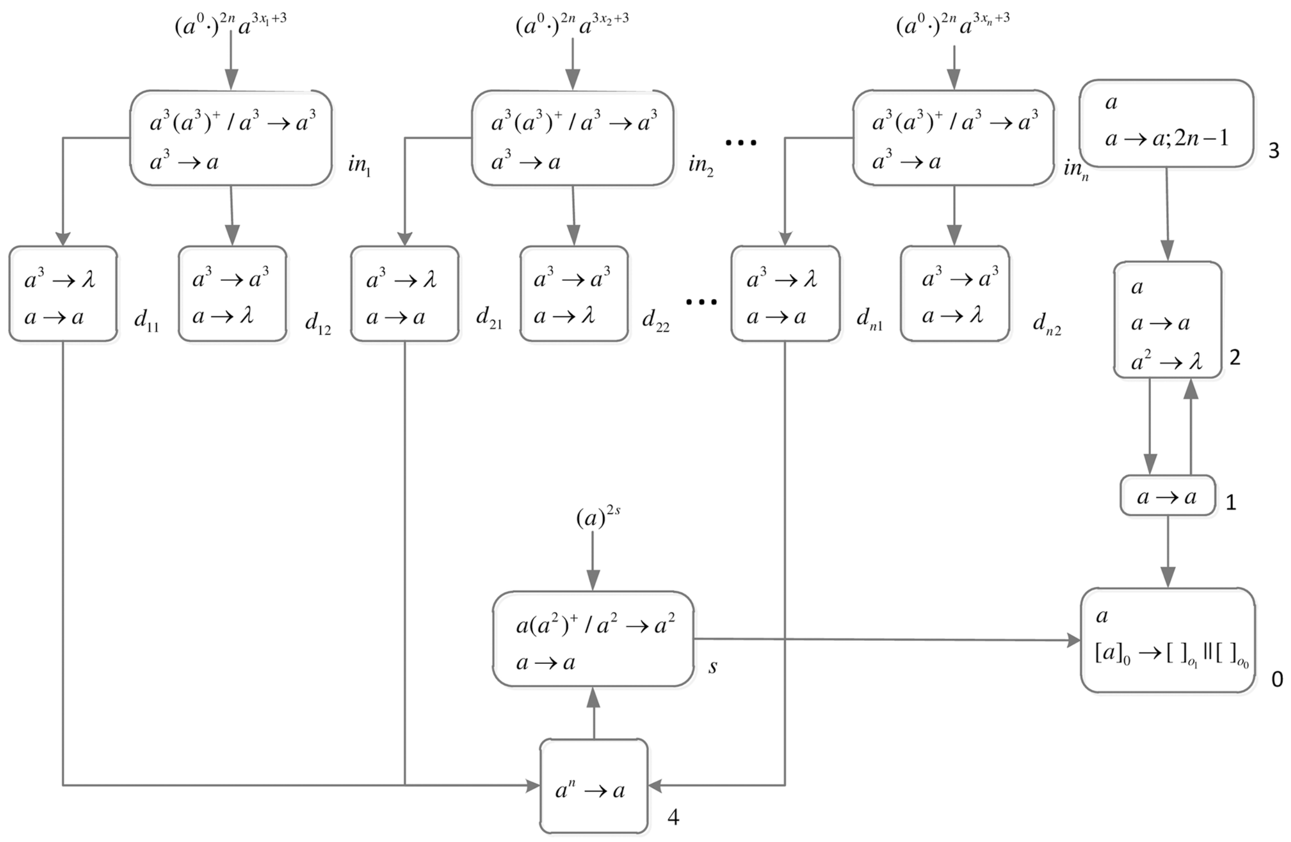


Fig 13. The initial system Π_n.

doi:10.1371/journal.pone.0162882.g013

Generation Stage: At step one, neuron σ_0 has one spike, the division rule $[a]_0 \to [\,]_{o_1} \| [\,]_{o_0}$ is applied to generate neurons σ_{o_1} and σ_{o_0}, which means one subset in regard to x_1 has two choices: x_1 is included in this subset (represent by 1) and x_1 is not included in this subset (represent by 0). Synapses $(1, o_1)$ and $(1, o_0)$ are established through the inheritance of synapse $(1, 0)$; synapses (s, o_1) and (s, o_0) are established through the inheritance of synapse $(s, 0)$; the synapse (d_{12}, o_1) is established through synapse dictionary *syn*. The synapse between neurons d_{12} and o_1 establishes a channel between the input and the subset having x_1. At the same time, auxiliary neuron σ_2 has one spike, rule $a \to a$ is applied and one spike is emitted to neuron σ_1; auxiliary neuron σ_3 has one spike, rule$a \to a$; $2n - 1$ is applied and one spike will be emitted to neurons σ_2 at step $2n$. The system after step one is shown in Fig 14.

At step two, neuron σ_1 has one spike, the firing rule $a \to a$ is applied, and one spike is emitted to neurons σ_2, σ_{o_1} and σ_{o_0}.

At step three, each of neurons σ_{o_1} and σ_{o_0} has one spike, the division rule $[a]_{o_{t1}} \to [\,]_{o_{t11}} \| [\,]_{o_{t10}} (t_1 = 1, 0)$ is applied to generate neurons $\sigma_{o_{11}}$, $\sigma_{o_{10}}$, $\sigma_{o_{01}}$ and $\sigma_{o_{00}}$, which means one subset in regard to x_1 and x_2 has four choices: x_1x_2 are included in this subset (represent by 11), x_1 is included in this subset and x_2 is not included in this subset (represent by 10), x_1 is not included in this subset and x_2 is included in this subset (represent by 01), and x_1x_2 are not included in this subset (represent by 00). Synapses $(1, o_{11})$, $(1, o_{10})$, $(1, o_{01})$, $(1, o_{00})$, (s, o_{11}), (s, o_{10}), (s, o_{01}), (s, o_{00}), (d_{12}, o_{11}) and (d_{12}, o_{10}) are established through the inheritance of synapse $(1, o_1)$, $(1, o_0)$, (s, o_1), (s, o_0) and (d_{12}, o_1); synapses (d_{22}, o_{11}) and (d_{22}, o_{01}) are established through synapse dictionary *syn*. Synapses (d_{12}, o_{11}) and (d_{12}, o_{10}) establish channels between the input and the

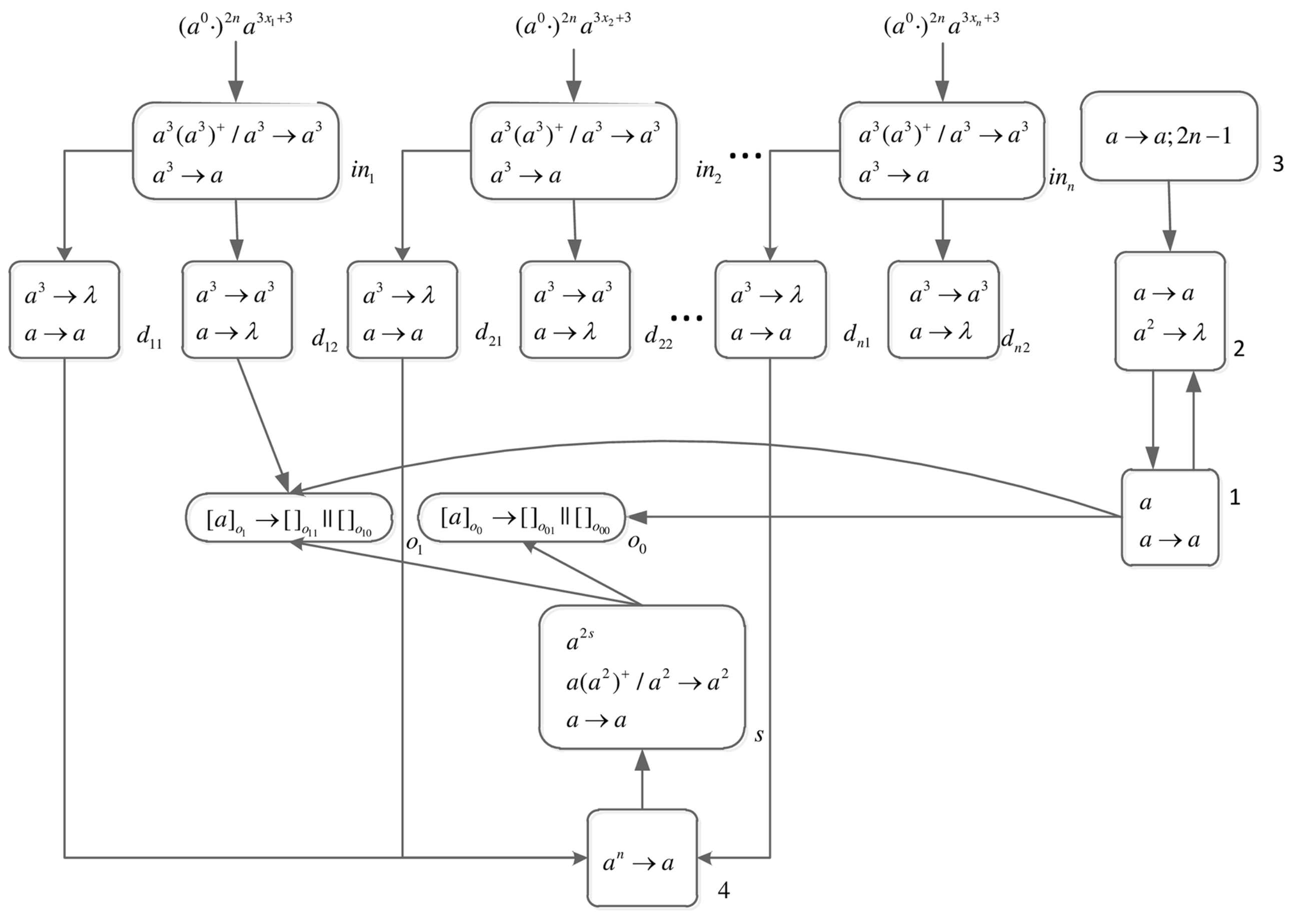


Fig 14. The system Π_n after step one.

doi:10.1371/journal.pone.0162882.g014

subsets having x_1; synapses (d_{22}, o_{11}) and (d_{22}, o_{01}) establish channels between the input and the subsets having x_2. At the same time, auxiliary neuron σ_2 has one spike, rule $a \to a$ is applied and one spike is emitted to neuron σ_1. The system after step three is shown in Fig 15.

Similar process repeats. At step $2n - 1$, 2^n neurons labeled $o_{t_1\ t2\ldots tn}(t_1, t_2, \ldots, t_n = 1, 2, \ldots, n)$ are generated. The system after step $2n - 1$ is shown in Fig 16.

At step $2n$, each neuron $\sigma_{o_{t1\ t}2\ldots{}_tn}$ receives one spike emitted from neuron σ_1 which will be deleted at the next step by the forgetting rule $[a \to \lambda]_{o_{t1\ t}2\ldots{}_tn}$. Neuron σ_2 receives two spikes (One is emitted from neuron σ_1, and another one is emitted from neuron σ_3.), the forgetting rule $a^2 \to \lambda$ is applied at step $2n + 1$, and no spike will be emitted to neuron σ_1 later. The system after step $2n$ is shown in Fig 17.

Input Stage: At step $2n + 1$, $x_1, x_2, \ldots, x_n$ enter the system through input neurons $\sigma_{in_i}(i = 1, 2, \ldots, n)$. $3x_1 + 3$ spikes (a^{3x_1+3}) enter neuron σ_{in_1}; $3x_2 + 3$ spikes (a^{3x_2+3}) enter neuron σ_{in_2};. . . $3x_n + 3$ spikes (a^{3x_n+3}) enter neuron σ_{in_n}.

At step $2n + 2$, the firing rule $a^3(a^3)^+/a^3 \to a^3$ is applied, and three spikes are replicated and are emitted to neurons $\sigma_{d_{i1}}$ and $\sigma_{d_{i2}}$. Spikes in neuron $\sigma_{d_{i1}}$ are forgotten, and spikes in neuron $\sigma_{d_{i2}}$ are emitted to these $\sigma_{o_{t1\ t}2\ldots{}_tn}$ having synapses going from neuron $\sigma_{d_{i2}}$ to them (These neurons represent the subsets having the integer x_i.) at step $2n + 3$. This process repeats until only 3 spikes are in neuron σ_{in_i}.

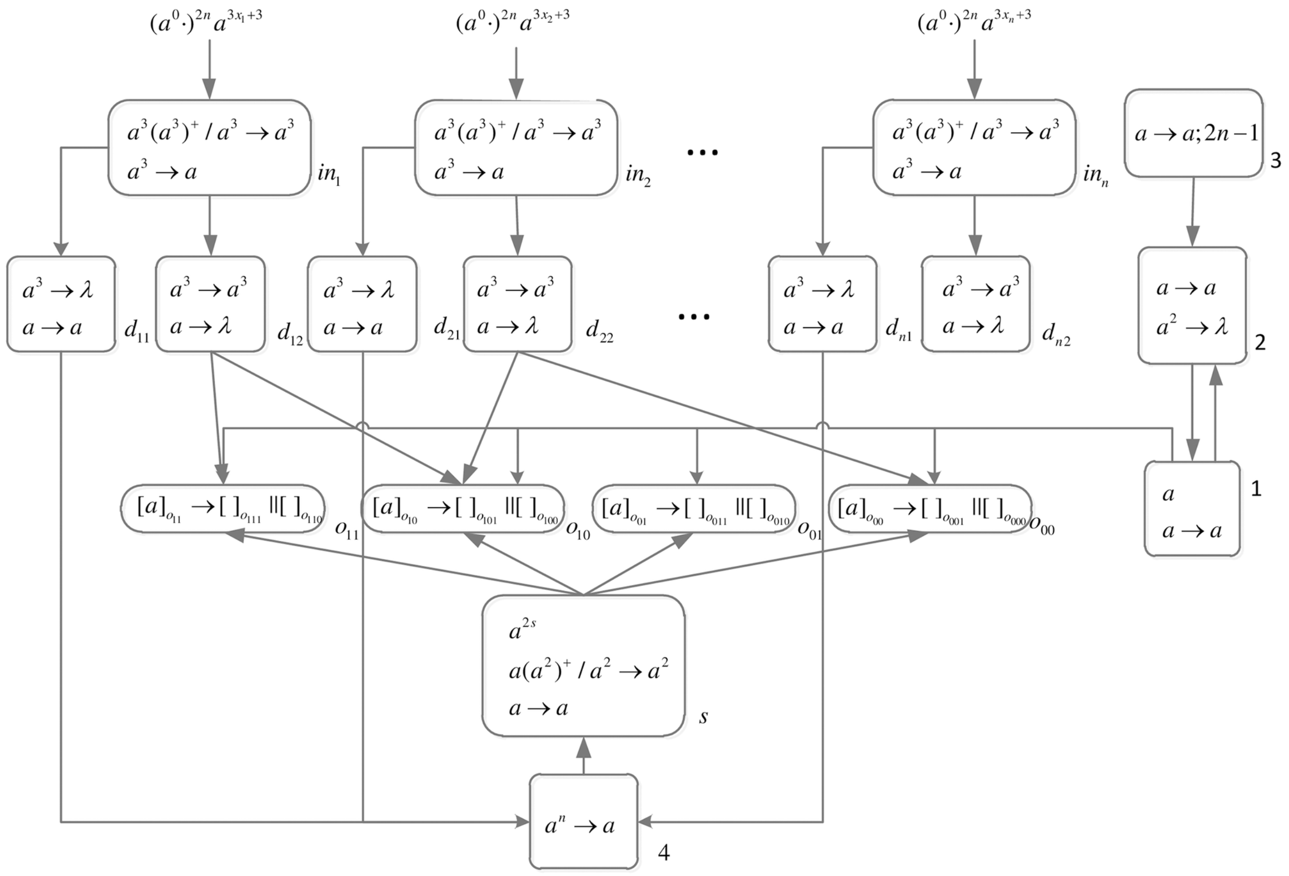


Fig 15. The system Π_n after step three.

doi:10.1371/journal.pone.0162882.g015

At step $2n + x_i + 2$, the firing rule $a^3 \to a$ is applied, and one spike is replicated and is emitted to neurons $\sigma_{d_{i1}}$ and $\sigma_{d_{i2}}$.

At step $2n + x_i + 3$, the spike in neuron $\sigma_{d_{i1}}$ is emitted to neuron σ_4 showing that all spikes in neuron σ_{in_i} have been passed to neurons $\sigma_{o_{t1\ t2\ldots tn}}$ having synapses going from neuron $\sigma_{d_{i2}}$ to them. The spike in neuron $\sigma_{d_{i2}}$ is forgotten. Up to this step, $3x_i$ spikes are emitted to neurons $\sigma_{o_{t1\ t2\ldots tn}}$ which represent the subsets having the integer x_i.

When all input spikes in neurons σ_{in_i} are passed to neurons $\sigma_{o_{t1\ t2\ldots tn}}$ at step $2n + x_{max} + 3$ (x_{max} represents the maximum integer of all n integers.), neuron σ_4 receives n spikes, and one spike is emitted to neuron σ_s at step $2n + x_{max} + 4$. Up to this step, the number of spikes in neurons $\sigma_{o_{t1\ t2\ldots tn}}$ is $3\sum_{b\in B} b$.

Checking Stage: At step $2n + x_{max} + 5$, $2s + 1$ spikes are in neuron σ_s, the firing rule $a(a^2)^+/a^2 \to a^2$ is applied, two spikes are emitted to neurons $\sigma_{o_{t1\ t2\ldots tn}}$. This process lasts for S circles.

At step $2n + x_{max} + s + 4$, only one spike is in neuron s, and this spike is emitted to neurons $\sigma_{o_{t1\ t2\ldots tn}}$.

There are three rule execution situations in neurons $\sigma_{o_{t1\ t2\ldots tn}}$.

1. $\sum_{b\in B} b = S$

   $3\sum_{b\in B} b$ spikes are in neuron $\sigma_{o_{t1\ t2\ldots tn}}$ initially. 2 spikes are emitted to this neuron from

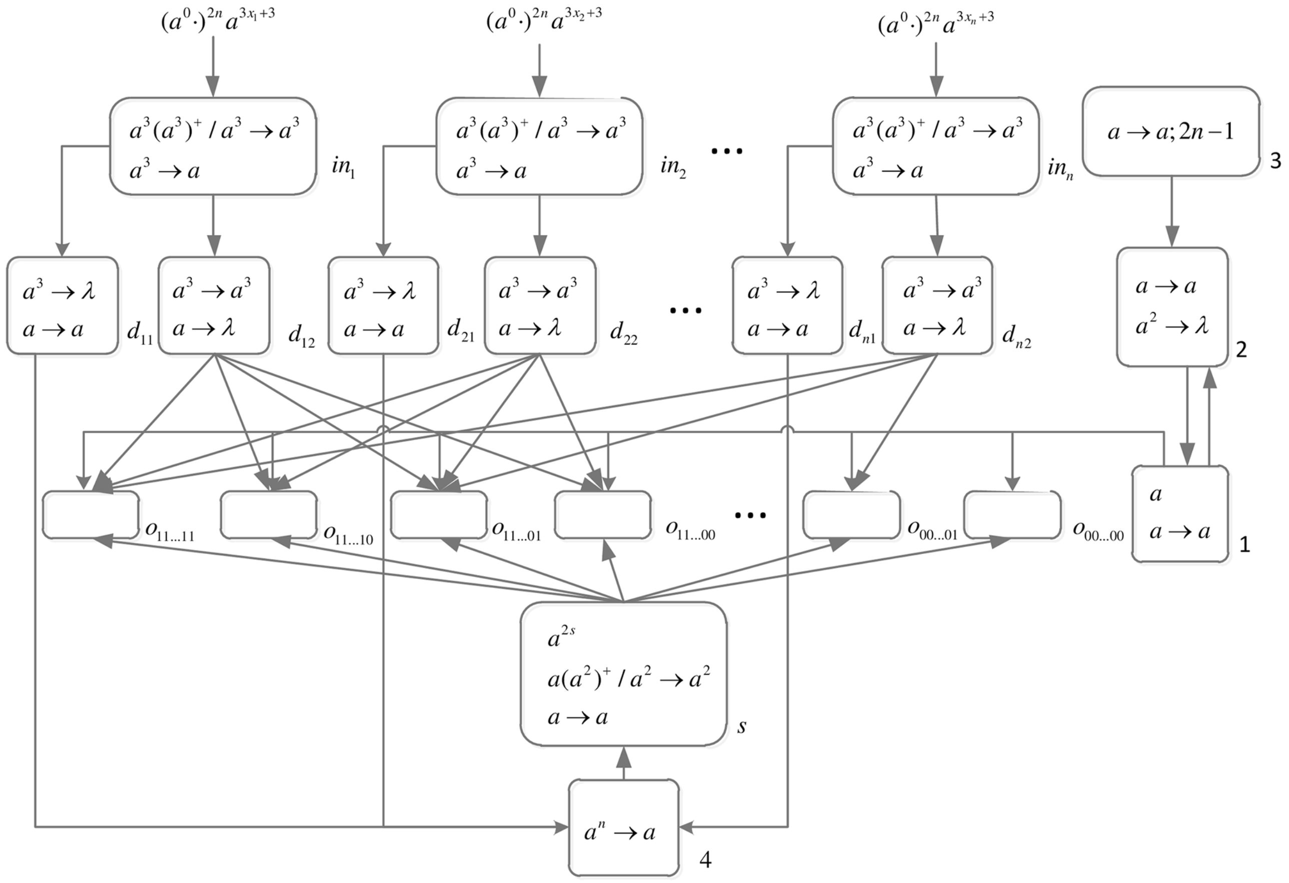


Fig 16. The system Π_n after step $2n - 1$.

doi:10.1371/journal.pone.0162882.g016

neuron σ_S, then forgetting rule $a^2(a^3)^+/a^5 \to \lambda$ can be applied with 5 spikes consumed. The number of spikes decreases to $3(\sum_{b\in B} b - 1)$. This process repeats S times, and all spikes in neuron $\sigma_{o_{t1\ t2\dots tn}}$ are consumed. At this step, the last one spike is emitted to this neuron from neuron σ_S, forgetting rule $a \to \lambda$ can be applied to consume this spike.

2. $\sum_{b\in B} b < S$

   $3\sum_{b\in B} b$ spikes are in neuron $\sigma_{o_{t1\ t2\dots tn}}$ initially. 2 spikes are emitted to this neuron from neuron σ_S, then forgetting rule $a^2(a^3)^+/a^5 \to \lambda$ can be applied with 5 spikes consumed. The number of spikes decreases to $3(\sum_{b\in B} b - 1)$. This process repeats $\sum_{b\in B} b$ times, and all spikes in neuron $\sigma_{o_{t1\ t2\dots tn}}$ are consumed. At this step, two spikes are emitted to this neuron from neuron σ_S, and neuron dissolution rule $[a^2]_{o_{t1\ t2\dots tn}} \to \delta(t_1\ t_2\dots t_n = 1, 2, \dots, n)$ is applied to dissolve this neuron.

3. $\sum_{b\in B} b > S$

   $3\sum_{b\in B} b$ spikes are in neuron $\sigma_{o_{t1\ t2\dots tn}}$ initially. 2 spikes are emitted to this neuron from neuron σ_S, then forgetting rule $a^2(a^3)^+/a^5 \to \lambda$ can be applied with 5 spikes consumed. The

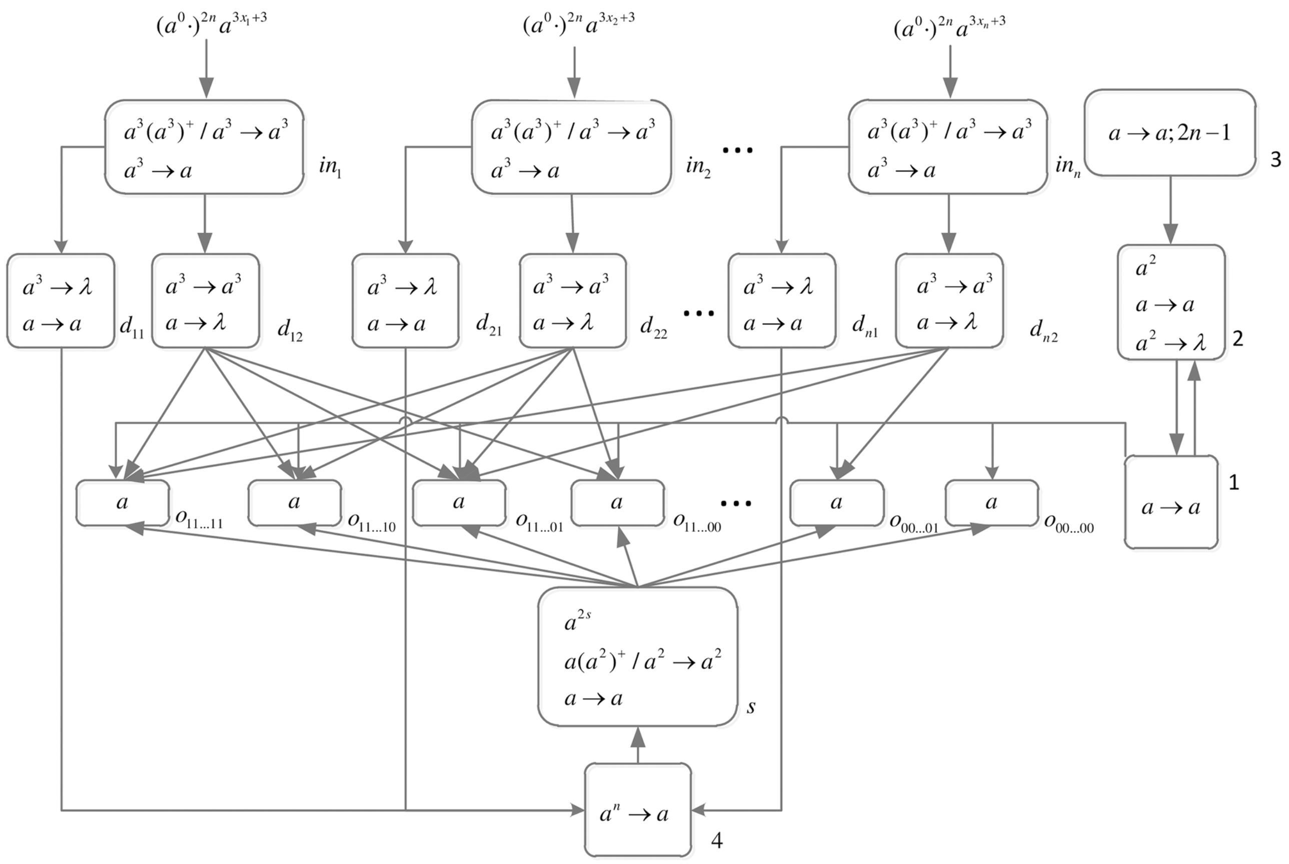


Fig 17. The system Π_n after step 2n.

doi:10.1371/journal.pone.0162882.g017

number of spikes decreases to $3(\sum_{b\in B} b - 1)$. This process repeats S times, and $3(\sum_{b\in B} b - S)$ spikes are remaining. At the next step, the last one spike in neuron σ_S is emitted to this neuron, and dissolution rule $[a(a^3)^+]_{o_{t1\ t2\ldots t n}} \to \delta(t_1\ t_2\ldots t_n = 1, 2, \ldots, n)$ is applied to dissolve this neuron.

If some neurons $\sigma_{o_{t1\ t2\ldots t n}}$ are still in the system after step $2n + x_{max} + s + 5$, the labels of these neurons $\sigma_{o_{t1\ t2\ldots t n}}$ are all solutions to this Subset Sum problem.

It can be seen that any *Subset Sum problem* (n) can be solved in linear time, and all solutions can be obtained through this system.

Some steps comparison results between our solution and other solutions, which use the non-deterministic method to solve the NP-complete problems, are shown in Table 2, where, k means that all $x_1, \ldots, x_n, S$ can be transformed into k-bit binary numbers.

Table 2. Steps comparison results of some uniform solutions to Subset Sum problem.

solution	step complexity (steps)	determinism or nondeterminism
SN P systems [19]	$3k + 2$	nondeterminism
SN P systems [20]	$3\Sigma_{i=1}^n x_i + 6$	nondeterminism
SNPSP systems [22]	$2\Sigma_{i=1}^n x_i + 6$	nondeterminism
time-free SN P systems [23]	$3\Sigma_{i=1}^n x_i + 2$	nondeterminism
DDSN P systems	$2n + x_{max} + s + 5$	determinism

doi:10.1371/journal.pone.0162882.t002

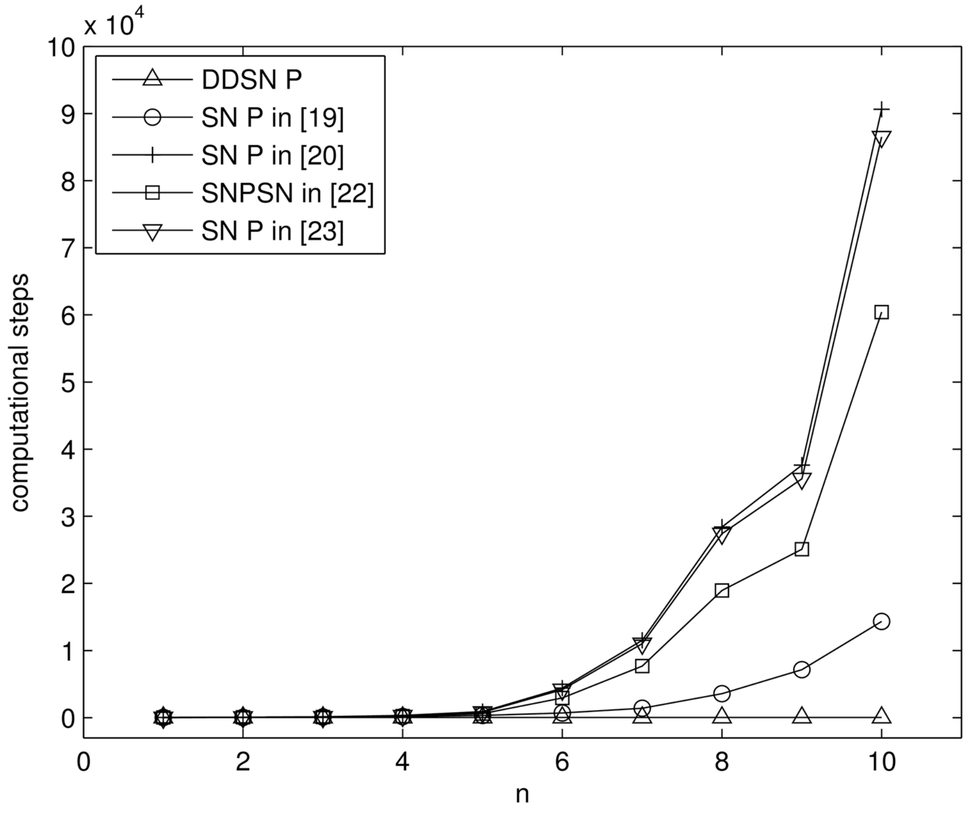


Fig 18. The computational steps of five types of SN P system solving Subset Sum problem.

doi:10.1371/journal.pone.0162882.g018

The conventional methods use the nondeterminism of SN P systems to solve Subset Sum problem, which means that whether a random combination of integers is one of the solutions or not can be checked by one computational process. These SN P systems can only judge whether a certain subset is the answer or not, but cannot search all solution space to judge whether a Subset Sum problem has solutions. Even if we let all combinations be traversed artificially to determine whether a Subset Sum problem has solutions or not, $(2^n - 1)$-times computations should be processed. Although the time complexity of each computation is a constant, the whole time complexity cannot be a polynomial of n. The proposed DDSN P system can solve the Subset Sum problem in a linear time, which improves the computational efficiency.

Considering a Subset Sum problem *Subset Sum problem (4)*: $X = \{1, 2, 3, 4\}$, $S = 5$, the DDSN P system Π_4 is used to solve it. After 22 computational steps, neurons $\sigma_{o_{0110}}$ and $\sigma_{o_{1001}}$ are remaining which shows that {2, 3} and {1, 4} are all solutions to this Subset Sum problem. Methods proposed in [19, 20, 23] need 165 steps, 330 steps and 270 steps, respectively to judge this problem.

A series of Subset Sum problems: $X = \{1, 2, \ldots, n\}$, $S = 5$ are solved using the five systems in Table 2 and the computational steps of these systems are shown in Fig 18 by MATLAB R2014a. The computational steps of DDSN P system are much fewer, especially when the problem size is larger.

4 Conclusions

The new mechanism called neuron dissolution is introduced into the framework of SN P systems in this work. By this mechanism, redundant neurons can be dissolved immediately. The computational resources can be saved, which means more work can be done using the same resources, or the same work can be done using less resources. We also proved that this new variant of SN P system can obtain all solutions to NP-complete problems (Invalid solutions are eliminated by neuron dissolution.), such as SAT problem and the Subset Sum problem, in linear time, which enhances the application fields of SN P systems such as the register allocation problem.

This work provides a new thought of storing information in SN P systems, which can be used to store other information. The dissolution rule can be used to many situations to decrease the space complexity of a SN P system. This variant of SN P system can be used to solve other NP-complete problems and application problems. It is also an attractive direction to introduce other biological phenomena into SN P systems to reduce computational resources and enhance computational space efficiency.

Author Contributions

Conceptualization: YZ XL.

Formal analysis: YZ WW.

Funding acquisition: XL.

Methodology: YZ XL.

Project administration: XL.

Resources: WW.

Software: YZ WW.

Supervision: XL.

Validation: YZ WW.

Visualization: XL.

Writing – original draft: YZ WW.

Writing – review & editing: XL.

References

1. Ionescu M, P un G, Yokomori T. Spiking neural P systems. Fundamenta Informaticae. 2006 Aug; 71 (2):279–308.
2. Cabarle F G C, Adorna H N, Pérez-Jiménez M J, Song T. Spiking neural P systems with structural plasticity. Neural Computing and Applications. 2015 Feb; 26(8):1905–1917. doi: 10.1007/s00521-015-1857-4
3. Cabarle F G C, Buño K C, Adorna H N. On the delays in spiking neural P systems. Symposium on Mathematical Aspects of Computer Science. 2012 Dec; 12(12):25–29.
4. Cabarle F G C, Buño K C, Adorna H N. Time after time: notes on delays in spiking neural P systems. Theory and Practice of Computation. 2012 Sep; 7:82–92. doi: 10.1007/978-4-431-54436-4_6
5. Cabarle F G C, Adorna H N, Pérez-Jiménez M J. Sequential spiking neural P systems with structural plasticity based on max/min spike number. Neural Computing and Applications. 2016 Jul; 27(5):1337–1347. doi: 10.1007/s00521-015-1937-5
6. Song T, Pan L. Spiking neural P systems with rules on synapses working in maximum spikes consumption strategy. IEEE Transactions on Nanobioscience. 2015 Jan; 14(1):38–44. doi: 10.1109/TNB.2014.2367506 PMID: 25389243
7. Song T, Pan L. Spiking neural P systems with rules on synapses working in maximum spiking strategy. IEEE Transactions on Nanobioscience. 2015 Jun; 14(4):465–477. doi: 10.1109/TNB.2015.2402311

8. Song T, Zou Q, Zeng X, Liu X. Asynchronous spiking neural P systems with rules on synapses. Neurocomputing. 2015 Mar; 151(1):1439–1445. doi: 10.1016/j.neucom.2014.10.044
9. Song T, Xu J, Pan L. On the universality and non-nniversality of spiking neural P systems with rules on synapses. IEEE Transactions on Nanobioscience. 2015 Dec; 14(8):960–966. doi: 10.1109/TNB.2015.2503603 PMID: 26625420
10. Pan L, Zeng X, Zhang X, Jiang Y. Spiking neural P systems with weighted synapses. Neural Processing Letters. 2012 Feb; 35(1):13–27. doi: 10.1007/s11063-011-9201-1
11. Tu M, Wang J, Peng H, Shi P. Application of adaptive fuzzy spiking neural P systems in fault diagnosis model of power systems. Chinese Journal of Electronics. 2014 Jan; 23(1):87–92.
12. Zhang G, Rong H, Neri F, Pérez-Jiménez M J. An optimization spiking neural P system for approximately solving combinatorial optimization problems. International Journal of Neural Systems. 2014 Aug; 24(05):1–16. doi: 10.1142/S0129065714400061
13. Zhang X, Pan L, P un A. On universality of axon P systems. IEEE Transactions on Neural Networks and Learning Systems. 2015 Nov; 26(11):2816–2829. doi: 10.1109/TNNLS.2015.2396940 PMID: 25680218
14. Zeng X, Zhang X, Zhang J, Liu J. Simulating spiking neural P systems with circuits. Journal of Computational and Theoretical Nanoscience. 2015 Sep; 12(9):2023–2026. doi: 10.1166/jctn.2015.3981
15. Song T, Zheng P, Wong M L D, Wang X. Design of logic gates using spiking neural P systems with homogeneous neurons and astrocytes-like control. Information Sciences. 2016 Dec; 372: 380–391. doi: 10.1016/j.ins.2016.08.055
16. Chen H, Ionescu M, Ishdorj T O. On the efficiency of spiking neural P systems. Fourth Brainstormming Week on Membrane Computing. 2006 Jan; 1:195–206.
17. Ishdorj T O, Leporati A. Uniform solutions to SAT and 3-SAT by spiking neural P systems with pre-computed resources. Natural Computing. 2008 Dec; 7(4):519–534. doi: 10.1007/s11047-008-9081-0
18. Leporati A, Gutiérrez-Naranjo M A. Solving Subset Sum by spiking neural P systems with pre-computed resources. Fundamenta Informaticae. 2008 Nov; 87(1):61–77.
19. Leporati A, Zandron C, Ferretti C, Mauri G. Solving numerical NP-complete problems with spiking neural P systems. Membrane Computing. 2007 Jun; 4860:336–352.
20. Leporati A, Mauri G, Zandron C, P un Gh, Pérez-Jiménez M J. Uniform solutions to SAT and Subset Sum by spiking neural P systems. Natural Computing. 2009 Dec; 8(4):681–702. doi: 10.1007/s11047-008-9091-y
21. Leporati A, Zandron C, Ferretti C, Mauri G. On the computational power of spiking neural P systems. International Journal of Unconventional Computing. 2009 Jan; 5(5):459–473.
22. Cabarle F G C, Hernandez N H S, Martínez-del Amor M A. Spiking Neural P Systems with Structural Plasticity: Attacking the Subset Sum Problem. Membrane Computing. 2015 Aug; 9504:106–116. doi: 10.1007/978-3-319-28475-0_8
23. Song T, Luo L, He J, Chen Z, Zhang K. Solving subset sum problems by time-free spiking neural P systems. Applied Mathematics and Information Sciences. 2014 Jan; 8(1):327–332. doi: 10.12785/amis/080140
24. Song T, Zheng H, He J. Solving vertex cover problem by tissue P systems with cell division. Applied Mathematics and Information Science. 2014 Jan; 8(1):333–337. doi: 10.12785/amis/080141
25. Pan L, P un Gh, Pérez-Jiménez M J. Spiking neural P systems with neuron division and budding. Science China Information Sciences. 2011 Aug; 54(8):1596–1607. doi: 10.1007/s11432-011-4303-y
26. Wang J, Hoogeboom H J, Pan L. Spiking neural P systems with neuron division. Membrane Computing. 2011 Aug;361–376.
27. Galli R, Gritti A, Bonfanti L, Vescovi A L. Neural stem cells an overview. Circulation Research. 2003 May; 92(6):598–608. doi: 10.1161/01.RES.0000065580.02404.F4 PMID: 12676811
28. Brown D A, Yang N, Ray S D. Encyclopedia of Toxicology (Third Edition). Amsterdam: Elsevier.; 2014.
29. P un Gh. Computing with membranes. Journal of Computer and System Sciences. 2000 Aug; 61(1):108–143. doi: 10.1006/jcss.1999.1693
30. P un Gh., Rozenberg G., Salomaa A.. The Oxford handbook of membrane computing. Oxford: Oxford University Press.; 2010.
31. Song T, Pan L. Spiking neural P systems with request rules. Neurocomputing. 2016 Jun; 193(12):193–200. doi: 10.1016/j.neucom.2016.02.023
32. Wang X, Song T, Gong F, Zheng P. On the computational power of spiking neural P systems with self-organization. Scientific Reports. 2016 Jun; 6:27624. doi: 10.1038/srep27624 PMID: 27283843

33. Song T, Wang X. Homogenous spiking neural P systems with inhibitory synapses. Neural Processing Letters. 2015 Aug; 42(1):199–214. doi: 10.1007/s11063-014-9352-y

34. Song T, Liu X, Zeng X. Asynchronous spiking neural P systems with anti-spikes. Neural Processing Letters. 2015 Dec; 42(3):633–647. doi: 10.1007/s11063-014-9378-1

35. Díaz-Pernil D, Berciano A, Pena-Cantillana F, Gutiérrez-Naranjo M A. Segmenting images with gradient-based edge detection using membrane computing. Pattern Recognition Letters. 2013 Jun; 34 (8):846–855. doi: 10.1016/j.patrec.2012.10.014

36. P un Gh, P un R. Membrane computing and economics: numerical P systems. Fundamenta Informaticae. 2006 Sep; 73(1–2):213–227.

37. Pan L, P un Gh, Song B. Flat maximal parallelism in P systems with promoters. Theoretical Computer Science. 2016 Apr; 623(11):83–91. doi: 10.1016/j.tcs.2015.10.027

38. Michael R G, David S J. Computers and intractability: a guide to the theory of NP-completeness. San Francisco: WH Freeman and Co.; 1979.