Neurocomputing

Spiking neural P systems with neuron permeability

--Manuscript Draft--

Manuscript Number:	NEUCOM-D-23-05734
Article Type:	Regular article
Section/Category:	Other research areas not listed above
Keywords:	Spiking neural P systems, Membrane computing, Neuron permeability, Turing universality
Abstract:	Spiking neural P systems (SNP systems) are a class of distributed parallel and interpretable computing models developed in recent years, which are abstracted from the mechanism of spiking neurons and the nervous system. At present, the development of SNP variants has become a hot spot. To enhance the plasticity of SNP systems, inspired by the biological neural mechanism of the variable permeability of neurons, spiking neural P systems with neuron permeability (NP-SNP systems) are discovered and proposed as a novel variant of SNP systems. In NP-SNP systems, neurons have variable permeability directly related to membrane thickness. Membrane permeability changes with the change of membrane thickness. The proposed permeability spike rules are used to quantify changes in permeability. A specific NP-SNP system for generating arbitrary natural numbers is constructed. It is proved that the computing power of NP-SNP systems possesses Turing universality from number-generation and number-acceptance. Devoted to the NP-complete problem, the NP-SNP system deterministically solves the Subset Sum problem in linear time. Compared with five variants, NP-SNP systems show advantages in less time steps and deterministic solutions.

Abstract

Spiking neural P systems (SNP systems) are a class of distributed parallel and interpretable computing models developed in recent years, which are abstracted from the mechanism of spiking neurons and the nervous system. At present, the development of SNP variants has become a hot spot. To enhance the plasticity of SNP systems, inspired by the biological neural mechanism of the variable permeability of neurons, spiking neural P systems with neuron permeability (NP-SNP systems) are discovered and proposed as a novel variant of SNP systems. In NP-SNP systems, neurons have variable permeability directly related to membrane thickness. Membrane permeability changes with the change of membrane thickness. The proposed permeability spike rules are used to quantify changes in permeability. A specific NP-SNP system for generating arbitrary natural numbers is constructed. It is proved that the computing power of NP-SNP systems possesses Turing universality from number-generation and number-acceptance. Devoted to the NP-complete problem, the NP-SNP system deterministically solves the Subset Sum problem in linear time. Compared with five variants, NP-SNP systems show advantages in less time steps and deterministic solutions.

Spiking neural P systems with neuron permeability

Liping Wang[a], Xiyu Liu[a,*], Yuzhen Zhao[a,*]

[a] *College of Business, Shandong Normal University, Jinan, China*

Abstract

Spiking neural P systems (SNP systems) are a class of distributed parallel and interpretable computing models developed in recent years, which are abstracted from the mechanism of spiking neurons and the nervous system. At present, the development of SNP variants has become a hot spot. To enhance the plasticity of SNP systems, inspired by the biological neural mechanism of the variable permeability of neurons, spiking neural P systems with neuron permeability (NP-SNP systems) are discovered and proposed as a novel variant of SNP systems. In NP-SNP systems, neurons have variable permeability directly related to membrane thickness. Membrane permeability changes with the change of membrane thickness. The proposed permeability spike rules are used to quantify changes in permeability. A specific NP-SNP system for generating arbitrary natural numbers is constructed. It is proved that the computing power of NP-SNP systems possesses Turing universality from number-generation and number-acceptance. Devoted to the NP-complete problem, the NP-SNP system deterministically solves the Subset Sum problem in linear time. Compared with five variants, NP-SNP systems show advantages in less time steps and deterministic solutions.

Keywords: Spiking neural P systems, Membrane computing, Neuron permeability, Turing universality

*Corresponding author
Email address: `xyliu@sdnu.edu.cn`; `zhaoyuzhen@sdnu.edu.cn` (Yuzhen Zhao)

1. Introduction

Neurons are the basic unit of brain information processing. Neurons and nerve tissues constitute a huge information processing system to maintain life activities. In nature-inspired computing, inspired by the nervous system, neural computing developed at the cellular level is under intense research, which is devoted to the modeling and information processing of artificial neural systems or networks. One of the promising directions is the exploration of new computing models and computing capabilities [1, 2]. Initially, for the recurrent neural network, this fully connected model with a reasonable distribution of synaptic weights was proved to be as powerful as the Turing machine [3]. Subsequently, the spiking neural network (SNN), a network that integrates time into the firing behavior of neurons, has been studied, and it is regarded as a representative of the third generation of neural networks [2, 4]. Wolfgang Maass analyzed in [2] that SNN composed of single spiking neurons has at least the same computing power as similar-scale network models such as multi-layer perceptions and sigmoidal neural nets.

Spiking neural P (SNP) systems are one of the most advanced and active computing models in the field of membrane computing of nature-inspired computing [5, 6, 7]. SNP systems have also become a research hotspot in natural computing and computer science. They have a directed graph structure of neurons and synaptic connections. The basic elements of SNP systems include spikes, rules, neurons and synapses. In SNP systems, the state of neurons is usually intuitively represented by the number of spikes, and the use of rules causes the state update. The state transition of SNP systems is regarded as the calculation of systems. SNP systems have the characteristics of distributed parallelism, uncertainty and interpretability. The computing results are given by the time interval between the first two spikes emitted to the environment [8], or the number of spikes sent to the environment [5], or the number of spikes in the specified neuron [7].

Since SNP systems were first formally proposed [5], research on them has continued to advance. In theoretical research, more research focuses on the variant model of SNP systems, proposes new SNP systems from the aspects of system structure [9, 10, 11, 12], objects and rules [13, 14, 15, 16], and verifies their computing power [17, 18]. Some studies further explore the differences in the calculation process by changing the operating mode of systems [19, 20, 21]. The computing power of the SNP system and its variants is reflected by verifying the Turing completeness [22, 23]. The variant of SNP systems can also complete these works of generating language [24] and solving NP-complete problems [25, 26]. In addition, researchers have applied SNP systems in image processing [27, 28], fault diagnosis [29, 30], combinatorial optimization [31, 32], natural language processing [33] and other fields [34, 35, 36]. Although the theoretical research on SNP systems has progressed rapidly, the SNP variant that considers the characteristic of neuron permeability has not been developed. This work focuses on the construction of an originative SNP system, the proof of computing power and the proof of computing efficiency by solving the NP-complete problem.

Studies have shown that membrane permeability is directly related to membrane thickness and negatively correlated [37]. The larger the membrane thickness, the worse the permeability and the lower the communication quality. At normal membrane thickness, cells usually maintain normal permeability and communication. And the cell survival time is shorter in the thinnest membrane thickness. The balance of cell permeability is interrupted, and cell membrane rupture leads to cell dissolution.

However, in traditional spiking neural P systems, neurons only play the role of separating the computational space. Through synaptic connections between neurons, postsynaptic neurons can receive the spike signals sent by presynaptic neurons. Inspired by the above biological facts, this work proposes a novel SNP system based on the variable permeability of neurons, called spiking neural P systems with neuron permeability (shortly, NP-SNP systems), in which permeability is characterized by membrane thickness.

In NP-SNP systems, the spike participates in the rule calculation as a communication object. Three different membrane thicknesses, 0, 1, and 2, are assigned to each neuron as the permeability coefficient k. The default permeability of neurons is 1, i.e., $k = 1$. Neurons with $k = 1$ play a role in dividing the space, and the internal reaction of neurons and the communication between neurons can be carried out normally. The membrane thickness may change dynamically with the execution of rules in neurons, and the permeability changes. Neurons with $k = 2$, can respond internally, but spikes cannot penetrate the neuron, and spikes that try to send out or try to send in will disappear. Neurons with $k = 0$, will be dissolved, and spikes and rules will also be dissolved. In addition, unlike the spiking rules in traditional SNP systems, the execution of rules in NP-SNP systems not only causes the state variation of neurons, but is also accompanied by dynamic changes in neuronal permeability. In view of this, the rules in NP-SNP systems are designated as permeability spike rules to show differences in form and use from traditional spike rules. Compared with the existing SNP systems, NP-SNP systems have stronger plasticity and a more novel computing mode. Therefore, the main contributions of this work are listed as follows.

(1) Driven by the biological fact of the dynamics of biofilm permeability, the computational model in the field of membrane computing, spiking neural P systems with neuron permeability (shortly, NP-SNP systems) are innovatively proposed, which not only has plasticity but also is easy to apply. In NP-SNP systems, the variable permeability of neurons is considered, and the permeability coefficient k is used to represent the permeability difference.

(2) The permeability spike rule is proposed, and the change of neuronal permeability is reflected by rule execution. After neurons receive the spike, rules are

activated to cause dynamics of membrane thickness, and the permeability of neurons changes accordingly.

(3) As number-generation and number-acceptance devices, the computing power of NP-SNP systems with Turing universality is verified, which means that NP-SNP systems are feasible as new membrane computing models. Moreover, the computing efficiency of NP-SNP systems is verified by solving the Subset Sum problem. Compared with with five models, the uniform solution of the Subset Sum problem is deterministically found by NP-SNP systems with fewer time steps.

In the follow-up arrangement, a detailed introduction and a specific example of NP-SNP systems are given in Section 2 and Section 3. Section 4 focuses on proving the computing power of NP-SNP systems. NP-SNP systems completely solve the Subset Sum problem in Section 5. The last section summarizes this work.

2. NP-SNP systems

Formally, a NP-SNP system is defined as follows:

$$\Pi = (O, \sigma_1, \sigma_2, ..., \sigma_m, syn, in, out),$$

(1) $O = \{a\}$ is the alphabet of system Π, and a represents a spike.

(2) $\sigma_i = (n_i, k, R_i)$, $1 \leq i \leq m$, denotes neuron i in system Π,

where,

a) $n_i \geq 0$ represents the number of spikes contained in neuron σ_i,

b)$k \in \{0, 1, 2\}$ is the permeability coefficient of neuron σ_i.

c) R_i denotes the set of permeability spike rules:

$E/a^u \rightarrow a^v; k(d)$, E is the regular expression over O, $u > 0$, $v \geq 0$, $k \in \{0, 1, 2\}$, $d \in \mathbb{N}^+$ is the duration associated with k.

(3) $syn = \{(i, j), 1 \leq i, j \leq m, i \neq j\}$ is the set of synapses in system Π.

(4) in, out indicate the labels of input and output neurons of system Π, respectively.

In this definition, for (1), the spike a, as the object of system Π, participates in the calculation through synaptic transmission.

For (2), σ_i, $1 \leq i \leq m$, denotes neuron i of m neurons. Each neuron has spikes, permeability, and rules. The permeability coefficient k is 0, 1, 2. In the initial state of neurons, $k = 1$ is default. If the initial $k \neq 1$, it will be noted. For the rule $E/a^u \rightarrow a^v; k(d)$, E is a regular expression. If exactly $E = a^u$, the rule can be written directly as $a^u \rightarrow a^v; k(d)$. $u > 0$ denotes the number of consumed spike a, $v \geq 0$ denotes the number of generated spike a. When $v = 0$, the rule can be written as $E/a^u \rightarrow \lambda; k(d)$. In permeability spike rules, d is a specific duration associated with k, indicating the duration of permeability maintenance. After d time step, the permeability changes. The use of rules will cause changes in membrane permeability. According to $k \in \{0, 1, 2\}$, three cases need specific analysis.

(i) When $k = 0$, the rule $E/a^u \rightarrow a^v; k(d)$ can be abbreviated as $E/a^u \rightarrow a^v; 0$. If $E/a^u \rightarrow a^v; 0$ is used, spikes a^u are consumed, and spikes a^v are generated and sent backwards. But the permeability coefficient k of the neuron changes from 1 to 0, and then the neuron will be dissolved, and spikes and rules in it will also be dissolved. No longer participate in subsequent calculations. Note that since the rule $E/a^u \rightarrow a^v; 0$ is used, the neuron is dissolved and the duration d no longer exists, so d is omitted in the rule $E/a^u \rightarrow a^v; 0$.

(ii) When $k = 1$, the permeability of the neuron remains unchanged. The rule $E/a^u \rightarrow a^v; k(d)$ can be abbreviated as $E/a^u \rightarrow a^v$. Its use will allow the internal response and communication of neurons to proceed normally.

(iii) When $k = 2$, the rule $E/a^u \rightarrow a^v; k(d)$ becomes $E/a^u \rightarrow a^v; 2(d)$. If $E/a^u \rightarrow a^v; 2(d)$ is used, u spikes are consumed, v spikes are generated and transmitted backwards. After this rule is used, the neuron permeability changes, the permeability coefficient k changes from 1 to 2, and the duration of $k = 2$ is d ($d \in \mathbb{N}^+$). This means that this neuron is only allowed to have internal responses during the subsequent d period, and the emitted or incoming spikes will disappear.

After the duration d ends, k recovers from 2 to 1, and the neuron continues to participate in normal communication.

(3) gives a set of synapses that represent connections between neurons. In (4), the labels of input and output neurons are given. The input neuron is used to input spikes, and the output neurons are used to output spikes to obtain the calculation result.

In NP-SNP systems, for neurons with permeability changes, their permeability changes occur after the completion of rule execution. For example, the rule $a^2 \to a; 2(3)$ lies in neuron m. After the execution of the rule, that is, the generated a is transmitted, the permeability of neuron m changes, and the permeability coefficient transits from the initial $k = 1$ to $k = 2$. Because of the duration $d = 3$, after 3 time steps, the rule can be used normally again. A concrete example can be seen in the Section 3.

The neurons in NP-SNP systems work in parallel, and only one rule is used in a time step in a neuron. If multiple rules can be activated in a time step, the rules in this neuron are selected non-deterministically and applied.

A global clock is introduced to count the time of the entire NP-SNP system. At step t, the set $\{(n_1(t), k_1(t)), ..., (n_m(t), k_m(t))\}$ of spike numbers and membrane permeability values in m neurons of a NP-SNP system is the configuration $\mathcal{C}_t$, and the set $\{(n_1(0), k_1(0)), ..., (n_m(0), k_m(0))\}$ in the initial state is the initial configuration $\mathcal{C}_0$. As the rules are used, the configuration changes. Converting from one configuration to another is a process of transformation, and a series of transformations can be defined as a calculation of the system Π. When no rules in the system can be used again, the computing process ends, and the calculation stops. In this work, the time interval between the first two nonempty spikes of the output neuron sent to the environment is used as the calculation result of NP-SNP systems.

For NP-SNP systems with at most m neurons, the natural number set family generated by them is represented by $N_{gen}NPSNP_m$; the natural number set family accepted by them is represented by $N_{acc}NPSNP_m$. In the case that the number of neurons is unlimited, m can be replaced by $*$

3. A specific example: generating arbitrary natural numbers

Fig. 1, the directed graph of the NP-SNP system Π_1, is designed. The preliminary contents of neurons σ_1, σ_2, σ_3, σ_4, σ_5 are as follows, where initially only σ_1 has a. The default initial permeability of five neurons is 1. The permeability of neurons σ_1, σ_2 and σ_4 are variable.

$\sigma_1 = (1, 1, \{a \to a; 0\})$,

$\sigma_2 = (0, 1, \{a \to a; 0\})$,

$\sigma_3 = (0, 1, \{a \to a\})$,

$\sigma_4 = (0, 1, \{a \to a, \quad a \to a; 2(1)\})$,

$\sigma_5 = (0, 1, \{a^3 \to a, \quad a^2 \to \lambda, \quad a \to a\})$.

The neuron σ_1 first fires to execute rule $a \to a; 0$, and then the permeability changes, $k = 0$, and σ_1 is dissolved. A spike is transmitted to neurons σ_2, σ_3, σ_4. At step 2, all three neurons are ignited. σ_2 and σ_3 use the only rule to send a to σ_5, while σ_4 activates a rule non-deterministically.

For the case where $a \to a; 2(1)$ is activated, neurons σ_2, σ_3 and σ_4 generate a spike that reaches σ_5 at the same time. σ_3 and σ_4 send spike a to each other. Then σ_2 dissolves due to the permeability transition to $k = 0$. At step 3, σ_5 sends out a through $a^3 \to a$, and σ_3 sends a spike to σ_4 and σ_5. σ_4 can only execute rule $a \to a; 2(1)$ internally because of $k = 2$, and spike a from σ_3 and spike a generated by itself disappear. In the next step, σ_5 sends out a using $a \to a$, and the calculation result 1 is generated by the interval $(4 - 3)$ of neuron σ_5 sending out a.

For the case where $a \to a$ is activated, neurons σ_2, σ_3 and σ_4 generate a spike that reaches σ_5 at the same time. σ_3 and σ_4 send spike a to each other. Then σ_2 dissolves due to the permeability transition to $k = 0$. At step 3, σ_5 sends a out by rule $a^3 \to a$. Neurons σ_3 and σ_4 still send a spike to each other, and both pass one to σ_5, which is removed by rule $a^2 \rightarrow \lambda$. If rule $a \rightarrow a$ in σ_4 is used continuously for n times, at step $n+1$, the rule $a \rightarrow a; 2(1)$ in σ_4 is activated, and the permeability coefficient k of σ_4 transits to 2, rule $a \rightarrow a; 2(1)$ is only executed internally at step $n+2$. Thus, at step $n+3$, σ_5 receives only one spike from σ_3 to enable rule $a \rightarrow a$ to send a out. Arbitrary $N \geq 1$ can be generated by $(n+3)-3$.

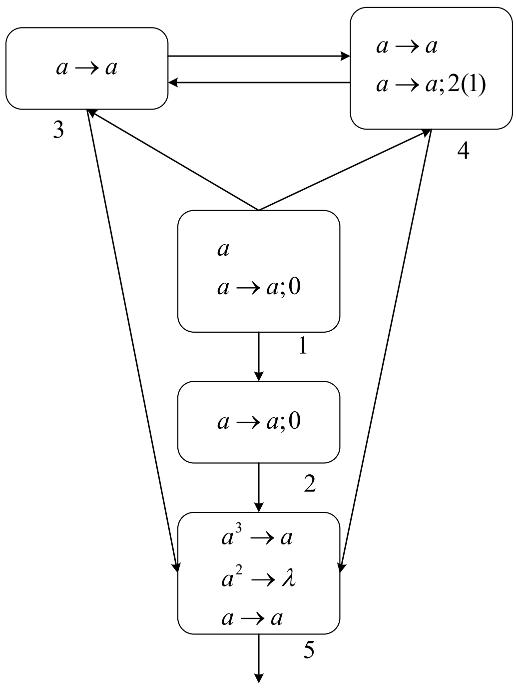

Figure 1: An example NP-SNP system Π_1

4. Computation power of NP-SNP systems

As number-generation and number-acceptance devices, the computational power of NP-SNP systems is demonstrated in this section. It simulates a general register machine and proves the Turing versatility of NP-SNP systems.

Table 1 gives the construction of the register machine represented by a tuple

Table 1: The construction of $M = (m, H, l_0, l_h, I)$.

Component	Content
m	Number of registers.
H	Instruction label set.
l_0	Start instruction.
l_h	Halt instruction.
I	Instruction set involving three forms:
	ADD instruction $l_i : (ADD(r), l_j, l_k)$: adds 1 to the number stored in register r, and then runs instruction l_j or l_k non-deterministically.
	SUB instruction $l_i : (SUB(r), l_j, l_k)$: subtracts 1 from the number stored in register r, and then runs instruction l_j if the stored number is greater than 0, or goes to instruction l_k, otherwise.
	HALT instruction $l_h : HALT$: terminates the calculation.

$M = (m, H, l_0, l_h, I)$ [38]. M can inscribe the family of length sets of all recursively enumerable languages (NRE).

The following Theorems prove that $N_{gen}NPSNP_*$ and $N_{acc}NPSNP_*$ are equivalent to NRE. The NP-SNP system consists of some modules composed of neurons (σ_r, σ_{l_i}, σ_{l_j}, σ_{l_k}) representing registers, instructions, and auxiliary neurons. In the generation mode, all σ_r are empty at first, the NP-SNP system simulating M executes the operation from l_0 to l_h until the calculation stops, and the generated number is stored in σ_r $(r = 1)$. Here, the ADD instruction is non-deterministic. In the acceptance mode, σ_r $(r = 1)$ stores the number identified, and the calculation process serves to accept the number. Here, the ADD instruction is determinate, namely $l_i : (ADD(r), l_j)$. Note that $2n$ in σ_r corresponds to n in register r.

Theorem 1. $N_{gen}NPSNP_{*} = NRE$

Proof. Through the Church-Turing thesis, it is obvious that $N_{gen}NPSNP_{*} \subseteq NRE$. The following proof shows that $N_{gen}NPSNP_{*} \supseteq NRE$. In this part, three modules (ADD, SUB and FIN) are included in the NP-SNP system simulating M. The initial permeability k of neurons is assumed to be 1 and omitted.

Module ADD (Fig. 2)

Suppose that at step t, three spikes are transmitted into neuron σ_{l_i}. Rules $a^3/a^2 \rightarrow a^2; 2(1)$ and $a^3 \rightarrow a^3$ satisfying the number of spikes are activated non-deterministically.

If $a^3/a^2 \rightarrow a^2; 2(1)$ is applied, two of the three spikes are replicated to neurons σ_{c_1}, σ_{c_2} and σ_{c_3}. After $a^3/a^2 \rightarrow a^2; 2(1)$ is used, the permeability of neuron σ_{l_i} changes, and the remaining spike a disappears after being replicated by rule $a \rightarrow a$. Two spikes in neuron σ_{c_1} are replicated by $a^2 \rightarrow a^2$ and sent to neuron σ_r. The number in σ_r increases by 1. Two spikes in neuron σ_{c_3} make the rule $a^2 \rightarrow a^2$ respond, generating a^2 and transmitting them to neurons σ_{c_2} and σ_{l_j}. Spikes a^2 in σ_{l_j} are inactivated by the rule $a^{2i} \rightarrow \lambda (i \geq 1)$. Neuron σ_{c_2} has a total of 4 spikes, rule $a^4 \rightarrow a^3$ is applied, and a^3 is passed into neuron σ_{l_k}. Neuron σ_{l_k} is activated for the following operation.

If $a^3 \rightarrow a^3$ is applied, three spikes are replicated to neurons σ_{c_1}, σ_{c_2} and σ_{c_3}. Spikes a^2 in neuron σ_{c_1} are generated by $a^3 \rightarrow a^2$ and sent to neuron σ_r. The number in σ_r increases by 1. a^3 received in σ_{c_2} is forgotten by rule $a^3 \rightarrow \lambda$. a^3 received in σ_{c_2} is copied to neurons σ_{c_2} and σ_{l_j} by rule $a^3 \rightarrow a^3$. a^3 in neuron σ_{c_2} is still inactivated. Neuron σ_{l_j} fires for the following operation.

Module SUB (Fig. 3)

The neuron σ_{l_i} receives three spikes at step t, and uses the rule $a^3 \rightarrow a^3$ to generate three spikes to neurons σ_{c_1}, σ_{c_2}, and σ_r in the next step. σ_{c_1} activates its rule $a^3 \rightarrow a$ and sends the spike a to postsynaptic σ_{l_k} and σ_{c_2}, but σ_{c_2} cannot receive it in the next step. σ_{c_2} activates its rule $a^3 \rightarrow a^2; 2(1)$ and sends a^2 to σ_{l_j},

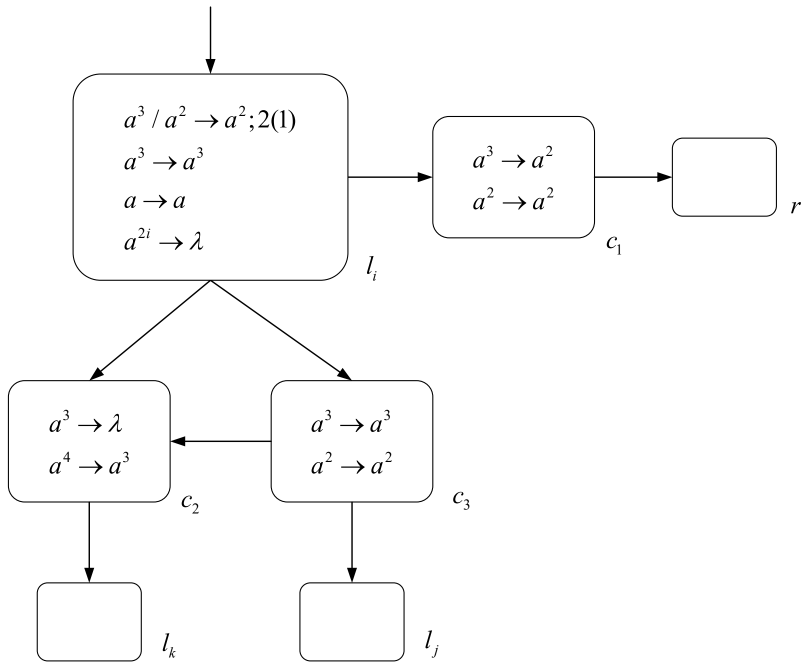

Figure 2: ADD

with the permeability of σ_{c_2} blocked for one step. Neuron σ_r activates different rules through a different number of spikes contained in itself.

If only a^3 exists in σ_r, the rule $a^3 \to a^2$ is applied at step $t+2$. σ_r sends a^2 to neurons σ_{c_1} and σ_{l_k}, but the spike entering σ_{c_1} is forgotten in the next step. Subsequently, σ_{l_j} can not be activated due to the rule $a^{2i} \to \lambda$. σ_{l_k} contains a total of three spikes from both σ_r and σ_{c_1}, which is activated.

If $2n + 3(n \geq 1)$ spikes are contained in neuron σ_r, σ_r activates the rule $a^3(a^2)^+/a^5 \to a^3$ at step $t+2$, consumes 5 spikes, and replicates 4 spikes to neurons σ_{c_1} and σ_{l_k}. At step $t+3$, σ_{l_k} has 5 spikes from neurons σ_r and σ_{c_1}, removes them using its rule $a^5 \to \lambda; 2(1)$, and causes a time-step permeability blockade. At this step σ_{l_j} also does not fire using its rule $a^{2i} \to \lambda$. σ_{c_1} emits its spikes to postsynaptic neurons σ_{l_k} and σ_{c_2} through the rule $a^4 \to a^4$, but the spikes cannot enter σ_{l_k}. One step later, σ_{c_2} uses its rule $a^4 \to a^3$ to replicate 3 spikes and emit to σ_{l_j} along the synapse (c_2, l_j). σ_{l_j} is activated and fires.

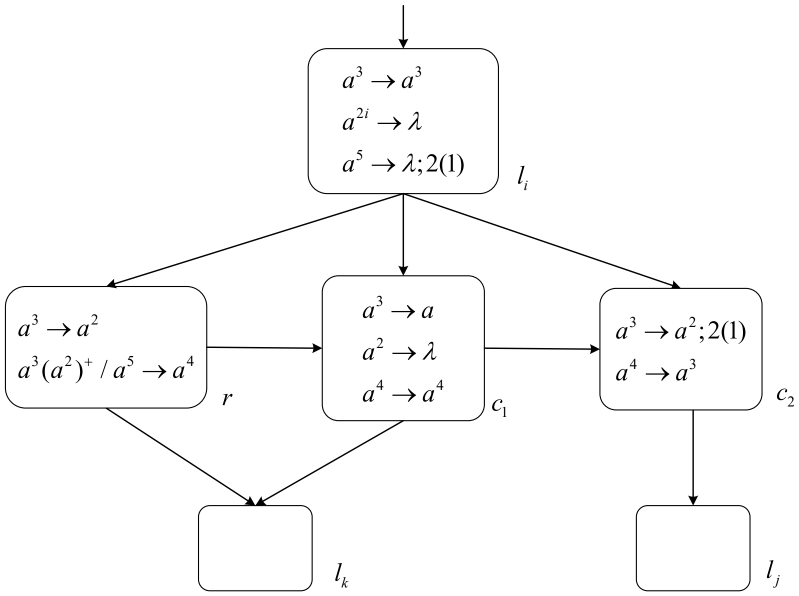

Figure 3: SUB

Module FIN (Fig. 4)

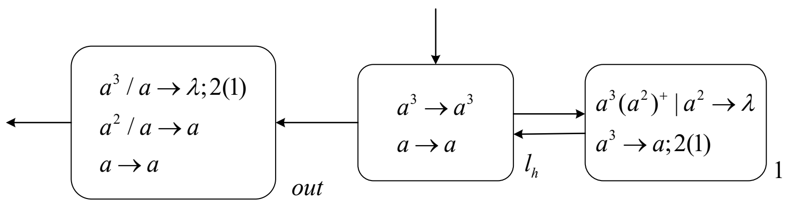

Figure 4: FIN

Assuming that 3 spikes reach neuron σ_{l_h}, then the halting instruction l_h begins to be simulated. σ_{l_h} executes the rule $a^3 \to a^3$ at step t and sends a^3 to σ_{out} and σ_1.

At step $t+1$, σ_{out} executes rule $a^3/a \to \lambda; 2(1)$, forgets a spike, and is accompanied by changes in neuronal permeability. In the next step, $a^2/a \to a$ can be executed inside σ_{out}, because $k=2$, the copy a fails to be transmitted and disappears. At step $t+3$, the neuron permeability recovers $k=1$, and the remaining a in σ_{out} can be sent out through the $a \to a$. For neuron σ_1, it corresponds to register $r(r=1)$, and $2n$ in σ_1 corresponds to n in $r(r=1)$. At step $t+1$, σ_1 has $2n+3$ spikes, and applies $a^3(a^2)^+/a^2 \to \lambda$ to consume 2 spikes repeatedly until step $t+n$. At step $t+n+1$, the final 3 spikes activate $a^3 \to a; 2(1)$, and a copy of a is transmitted to σ_{l_h}, causing a time-step impermeability of the neuron σ_1. After a time-step transmission of σ_{l_h}, at step $t+n+3$, σ_{out} uses $a \to a$ to send a out again. σ_{out} sends spike a with an interval of n (by $(t+n+3)-(t+3)$) steps. When the NP-SNP system ends the simulation of M, the interval is the same as the value in register r.

The above modules are calculated, and Theorem 1 is established.

□

Theorem 2. $N_{acc}NPSNP_* = NRE$

Proof. The capability of NP-SNP systems for number-acceptance is proved in this Theorem. Three modules (INPUT, ADD and SUB) are used, where ADD is the deterministic version.

Module INPUT (Fig. 5)

It is assumed that neuron σ_{in} receives 4 spikes and fires at step t. Applying rule $a^4/a^2 \to a^2$, 2 spikes are consumed, and 2 spikes are given to σ_{c_1} and σ_{c_2}, respectively. Subsequently, neurons σ_{in} , σ_{c_1} and σ_{c_2} are fired. σ_{in} and σ_{c_1} use the same rule $a^2 \to a^2$ to send 2 spikes to each other, and both send a^2 to σ_{c_2}. The $2i(i \geq 1)$ spikes in σ_{c_2} are all forgotten by $a^{2i} \to \lambda$. σ_{c_1} also continuously sends 2 spikes to neuron σ_1. After n steps, σ_{in} increases spike a, and rule $a^3 \to a; 2(1)$ is activated to send a to σ_{c_1} and σ_{c_2}. At this moment, σ_{c_2} receives a from σ_{in} and a^2

from σ_{c_1}, and rule $a^3 \to a^3$ is applied to send a^3 to neuron σ_{l_0}. The spike a in σ_{c_1} is removed.

Thus, σ_1 receives $2n$ spikes, and σ_{l_0} receives 3 spikes for the following simulation.

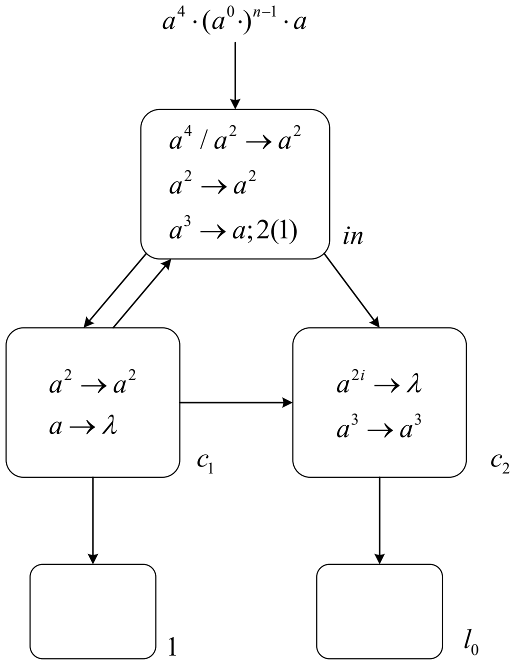

Figure 5: INPUT

Module ADD (Fig. 6)

After receiving 3 spikes, neuron σ_{l_i} first sends a^2 to σ_{c_1} and σ_{l_j} using $a^3/a^2 \to a^2$. These 2 spikes are sent to neuron σ_r through σ_{c_1}. In the next step, σ_{l_i} sends a to σ_{c_1} and σ_{l_j}, and the spike to σ_{c_1} will disappear. Neuron σ_{l_j} has a total of 3 spikes and fires for simulation.

Module SUB (Fig. 3)

The SUB module is described in Theorem 1 above, omitted here.

After all instructions from l_0 to l_h are simulated, the calculation stops. $N_{acc}NPSNP_* =$

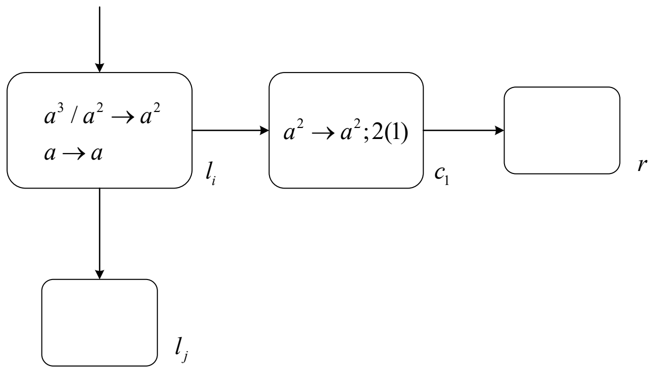

Figure 6: ADD

NRE is verified.

□

5. Uniform solution to Subset Sum problem

Using the variable permeability of neurons, the solution of Subset Sum problem in NP-complete problems [39] is proposed in this section. Specifically, a uniform solution is constructed by an NP-SNP system Π_s operating in a deterministic manner, which reduces the step complexity compared to advanced works [25, 40, 41, 42, 43].

Subset Sum problem: for an instance, S, and a (multi)set $\{x_1, x_2, ..., x_n\}$, with S, $x_i \in \mathbb{N}$ and $1 \leqslant i \leqslant n$. Is there a sub(multi)set $B \subseteq \{x_1, x_2, ..., x_n\}$ such that $\sum_{b \in B} b = S$?

Fig. 7 shows the working diagram of the system Π_s. Neurons $\sigma_1, \sigma_2, ..., \sigma_n$ and σ_s are input neurons, responsible for introducing $x_1, x_2, ..., x_n$ and S. Neurons $\sigma_{x_1,x_2,...,x_n}$, $x_i \in \{0, 1\}(1 \leq i \leq n)$ represents all combinations that can be formed when the variable x_i is 0 and 1, respectively, where 1 indicates that x_i is positive and 0 is negative. Taking $\sigma_{11...10}$ as an example, $\sigma_{11...10}$ indicates that the values of all variables except variable x_n are true. Neuron $\sigma_{x_1,x_2,...,x_n}$, $x_i \in \{0, 1\}$ and neuron σ_s participate in the satisfaction check. Other neurons $\sigma_{c_{i1}}$, $\sigma_{c_{i2}}$, σ_h involve calculations to complete the calculation process. The detailed calculation process is as follows.

Firstly, input information, and input $x_1, x_2, ..., x_n$, and S in the instance into the system Π_s in a readable spike coding. The spike trains enter the system Π_s through neurons $\sigma_1, \sigma_2, ..., \sigma_n$ and σ_s. In order to ensure that the spike train encoding S can be completely introduced into the neuron σ_s, $(a^0\cdot)^{s-4}$ is added before each spike train entering the neurons $\sigma_1, \sigma_2, ..., \sigma_n$.

At step $s-3$, $3x_1+2$ spikes a^{3x_1+2} enter σ_1, $3x_1+2$ spikes a^{3x_1+2} enter σ_2, and so on, $3x_n+2$ spikes a^{3x_n+2} enter neuron σ_n.

At step $s-2$, the rule $a^2(a^3)^{+/a^3 \to a^3}$ in $\sigma_i(1 \leq i \leq n)$ is activated, and spike a^3 is transmitted to neurons $\sigma_{c_{i1}}$, $\sigma_{c_{i2}}$. At step $s-1$, the three spikes in $\sigma_{c_{i2}}$ are discarded by $a^3 \to \lambda$, and the spikes in neuron $\sigma_{c_{i1}}$ are copied through rule $a^3 \to a^3$ and emitted into the connected neuron $\sigma_{x_1,x_2,...,x_n}$, $x_i \in \{0,1\}$. This process was repeated until only two spikes remained in neuron σ_i.

At step $s+x_i-2$, the rule $a^2 \to a$ in σ_i is used, spike a is generated and transmitted to neurons $\sigma_{c_{i1}}$ and $\sigma_{c_{i2}}$. At step $s+x_i-1$, the spike $\sigma_{c_{i1}}$ is forgotten. In $\sigma_{c_{i2}}$, the spike a is replicated by $a \to a$ and sent to σ_h. At this point, the $3x_i$ spikes in neuron σ_i have been transmitted to neuron $\sigma_{x_1,x_2,...,x_n}$, $x_i \in \{0,1\}$, which can reflect the explanation of $B \subseteq \{x_1, x_2, ..., x_n\}$ in the above instance, and the subset B contains x_i. The number of spikes in the neuron $\sigma_{x_1,x_2,...,x_n}$, $x_i \in \{0,1\}$ can be recorded as $3\sum_{b\in B} b$.

At step $s+x_{max}-1$ (x_{max} denotes the largest of $x_1, x_2, ..., x_n$), a total of n spikes are received by the neuron σ_h. In the next step, σ_h sends spike a to σ_s. So far, there are $2s+1$ spikes in neuron σ_s.

Secondly, the satisfiability check is performed to verify whether $\sum_{b\in B} b = S$, where $B \subseteq \{x_1, x_2, ..., x_n\}$.

At step $s+x_{max}+1$, rule $a(a^2)^+/a^2 \to a^2$ is activated in σ_s, and two spikes are replicated to neuron $\sigma_{x_1,x_2,...,x_n}$, $x_i \in \{0,1\}$. From step $s+x_{max}+1$ to step

$2s + x_{max}$, this process is repeated s times. At step $2s + x_{max} + 1$, only one spike is left in σ_s, which is sent to $\sigma_{x_1,x_2,\ldots,x_n}$, $x_i \in \{0,1\}$ through the rule $a \rightarrow a$.

In the process of $s + x_{max} + 1$ to $2s + x_{max} + 1$, for $\sum_{b\in B} b = S$ in $\sigma_{x_1,x_2,\ldots,x_n}$, $x_i \in \{0,1\}$: after receiving spike a^2 from σ_s, $\sigma_{x_1,x_2,\ldots,x_n}$, $x_i \in \{0,1\}$ fires due to the rule $a^2(a^3)^+/a^5 \rightarrow \lambda$, consuming five spikes and remaining $3\sum_{b\in B} b - 3$. Similarly, this process occurs s times, and then the last spike a from σ_s is inactivated.

For $\sum_{b\in B} b < S$ in $\sigma_{x_1,x_2,\ldots,x_n}$, $x_i \in \{0,1\}$: after receiving spike a^2 from σ_s, $\sigma_{x_1,x_2,\ldots,x_n}$, $x_i \in \{0,1\}$ fires due to the rule $a^2(a^3)^+/a^5 \rightarrow \lambda$, consuming five spikes and remaining $3\sum_{b\in B} b - 3$. This process occurs $\sum_{b\in B} b$ times, all spikes in $\sigma_{x_1,x_2,\ldots,x_n}$, $x_i \in \{0,1\}$ are consumed. Subsequent spike a^2 from σ_s activates rule $a^2 \rightarrow \lambda; 0$, the permeability of neuron $\sigma_{x_1,x_2,\ldots,x_n}$, $x_i \in \{0,1\}$ changes, and both neuron and rules dissolve.

For $\sum_{b\in B} b > S$ in $\sigma_{x_1,x_2,\ldots,x_n}$, $x_i \in \{0,1\}$: after receiving spike a^2 from σ_s, $\sigma_{x_1,x_2,\ldots,x_n}$, $x_i \in \{0,1\}$ fires due to the rule $a^2(a^3)^+/a^5 \rightarrow \lambda$, consuming five spikes and remaining $3\sum_{b\in B} b - 3$. This process repeats s times, and $3\sum_{b\in B} b - 3S$ spikes are left in $\sigma_{x_1,x_2,\ldots,x_n}$. Subsequently, the last spike a in σ_s reaches $\sigma_{x_1,x_2,\ldots,x_n}$, $x_i \in \{0,1\}$, and rule $a(a^3)^+ \rightarrow \lambda$ is applied. The permeability of neuron $\sigma_{x_1,x_2,\ldots,x_n}$, $x_i \in \{0,1\}$ changes, and both neuron and rules are dissolved.

Finally, determine the solution to the problem. After the above process is completed, if there are neurons left, then the answer to the Subset Sum problem is positive, and its solution can be known by the label of remaining neurons. Thus, the problem is solved in $2s + x_{max} + 1$ time steps.

In addition, this work also gives a comparison with advanced works, see Table 2. In this Table, v_i is the same as x_i, S is the same as s and both are much smaller than n. In 2016, the author used the proposed DDSNP system to find the solution of Subset Sum problem in $2n + x_{max} + s + 5$ steps. Later, NSNP-DW systems, SNPE systems, SNPCS systems and RSSNP systems verify the satisfiability of Subset Sum problem in a non-deterministic way, but they can not get a clear solution. NP-SNP systems proposed in this work use the variable permeability of neurons to delete non-solutions by rule execution, which helps to find the solution to the problem. Compared with DDSNP systems, NP-SNP systems take less time to solve the Subset Sum problem, which makes the solution process more optimized. Compared with NSNP-DW systems, SNPE systems, SNPCS systems and RSSNP systems, NP-SNP systems can not only determine whether the problem is satisfied in less time, but also give a definite solution. By improving existing works, a feasible solution with fewer steps is obtained.

In summary, NP-SNP systems constructed using the variable permeability of neurons in this work can effectively solve the Subset Sum problem and have more advantages.

Table 2: Comparison of several works in time steps and finding solutions

Models	time steps	find solutions
DDSNP[25]	$2n + x_{max} + s + 5$	yes
NSNP-DW[40]	$2\sum_{i=1}^{n} v_i + 5$	no
SNPE[41]	$2\sum_{i=1}^{n} v_i + 3$	no
SNPCS[42]	$2(\sum_{i=1}^{n} v_i + S) + S + 8$	no
RSSNP[43]	$2\sum_{i=1}^{n} v_i + 5$	no
NP-SNP	$2s + x_{max} + 1$	yes

6. Conclusions

This work proposes spiking neural P systems with neuron permeability (shortly, NP-SNP systems), which utilizes the variable permeability of neurons. In NP-SNP systems, the permeability is quantified by three different k values. The dynamic spike rule is proposed and applied for the first time, and the change of neuron permeability is reflected by rule execution. As a new membrane computing model, it is proved that the NP-SNP system is as powerful as the Turing machine from number-generation and number-acceptance. By solving the Subset Sum problem, the advantages of the NP-SNP system for solving NP-complete problems are demonstrated. The neuron permeability reflected by rule execution helps to find specific solutions. However, compared with existing SNP systems, NP-SNP systems have the computational characteristic of variable neuron permeability and stronger plasticity.

So far, it is still challenging to establish NP-SNP systems that use the least number of neurons. Later, it is worth exploring to control preventable behaviors with NP-SNP systems, such as constructing the deadlock prevention model. Of course, designing algorithms based on NP-SNP systems and finding a counterpart application scenario for NP-SNP systems will also be valuable tasks.

Acknowledgments

This work has been supported by the National Natural Science Foundation of China (61876101, 61806114, 61802234, 62172262), the Natural Science Fund Project of Shandong Province, China (ZR2023MF079, ZR2019QF007), the Postdoctoral Project, China (2017M612339, 2018M642695), the Postdoctoral Special Funding Project, China (2019T120607).

References

References

[1] G. Rozenberg, T. Bäck, J. N. Kok, Handbook of natural computing, Springer, 2012.

[2] W. Maass, Network of spiking neurons: the third generation of neural network models, Transactions of the Society for Computer Simulation International 14 (4) (1997) 1659–1671.

[3] H. T. Siegelmann, E. D. Sontag, Turing computability with neural nets, Applied Mathematics Letters 4 (6) (1991) 77–80.

[4] S. Ghosh-Dastidar, H. Adeli, Spiking neural networks, International journal of neural systems 19 (04) (2009) 295–308.

[5] M. Ionescu, G. Păun, T. Yokomori, Spiking neural P systems, Fundamenta informaticae 71 (2-3) (2006) 279–308.

[6] H. Peng, Z. Lv, B. Li, X. Luo, J. Wang, X. Song, T. Wang, M. J. Pérez-Jiménez, A. Riscos-Núñez, Nonlinear spiking neural P systems, International Journal of Neural Systems 30 (10) (2020) 2050008.

[7] T. Wu, L. Pan, Spiking neural P systems with communication on request and mute rules, IEEE Transactions on Parallel and Distributed Systems 34 (2) (2022) 734–745.

[8] M. Cavaliere, O. H. Ibarra, G. Păun, O. Egecioglu, M. Ionescu, S. Woodworth, Asynchronous spiking neural P systems, Theoretical Computer Science 410 (24-25) (2009) 2352–2364.

[9] R. T. A. de la Cruz, F. G. C. Cabarle, I. C. H. Macababayao, H. N. Adorna, X. Zeng, Homogeneous spiking neural P systems with structural plasticity, Journal of Membrane Computing 3 (2021) 10–21.

[10] F. G. C. Cabarle, H. N. Adorna, M. Jiang, X. Zeng, Spiking neural P systems with scheduled synapses, IEEE Transactions on Nanobioscience 16 (8) (2017) 792–801.

[11] L. Garcia, G. Sanchez, E. Vazquez, G. Avalos, E. Anides, M. Nakano, G. Sanchez, H. Perez, Small universal spiking neural P systems with dendritic/axonal delays and dendritic trunk/feedback, Neural Networks 138 (2021) 126–139.

[12] T. Wu, Z. Zhang, G. Păun, L. Pan, Cell-like spiking neural P systems, Theoretical Computer Science 623 (2016) 180–189.

[13] T. Wu, L. Pan, Q. Yu, K. C. Tan, Numerical spiking neural P systems, IEEE Transactions on Neural Networks and Learning Systems 32 (6) (2020) 2443–2457.

[14] L. Wang, X. Liu, M. Sun, Y. Zhao, Evolution-communication spiking neural P systems with energy request rules, Neural Networks 164 (2023) 476–488.

[15] H. Peng, B. Li, J. Wang, X. Song, T. Wang, L. Valencia-Cabrera, I. Pérez-Hurtado, A. Riscos-Núñez, M. J. Pérez-Jiménez, Spiking neural P systems with inhibitory rules, Knowledge-Based Systems 188 (2020) 105064.

[16] Y. Liu, Y. Zhao, Spiking neural P systems with lateral inhibition, Neural Networks 167 (2023) 36–49.

[17] Y. Zhao, Y. Shen, X. Liu, Y. Luo, W. Zang, X. Liu, Spiking neural P systems with long-term potentiation and depression, Information Sciences 640 (2023) 119082.

[18] X. Wang, T. Song, F. Gong, P. Zheng, On the computational power of spiking neural P systems with self-organization, Scientific reports 6 (1) (2016) 27624.

[19] S. Jiang, Y. Liu, B. Xu, J. Sun, Y. Wang, Asynchronous numerical spiking neural P systems, Information Sciences 605 (2022) 1–14.

[20] X. Zhang, X. Zeng, B. Luo, L. Pan, On some classes of sequential spiking neural P systems, Neural Computation 26 (5) (2014) 974–997.

[21] T. Wu, A. Păun, Z. Zhang, L. Pan, Spiking neural P systems with polarizations, IEEE transactions on neural networks and learning systems 29 (8) (2017) 3349–3360.

[22] L. Garcia, G. Sanchez, J.-G. Avalos, E. Vazquez, Spiking neural P systems with myelin and dendritic spines, Neurocomputing 552 (2023) 126522.

[23] T. Wu, F.-D. Bîlbîe, A. Păun, L. Pan, F. Neri, Simplified and yet turing universal spiking neural P systems with communication on request, International journal of neural systems 28 (08) (2018) 1850013.

[24] F. G. C. Cabarle, R. T. A. de la Cruz, X. Zhang, M. Jiang, X. Liu, X. Zeng, On string languages generated by spiking neural P systems with structural plasticity, IEEE transactions on nanobioscience 17 (4) (2018) 560–566.

[25] Y. Zhao, X. Liu, W. Wang, Spiking neural P systems with neuron division and dissolution, PLoS One 11 (9) (2016) e0162882.

[26] L. Zhang, F. Xu, Spiking neural P systems with cooperative synapses, Neurocomputing 501 (2022) 222–230.

[27] J. Xue, Q. Li, X. Liu, Y. Guo, J. Lu, B. Song, P. Huang, Q. An, G. Gong, D. Li, Hybrid neural-like P systems with evolutionary channels for multiple brain metastases segmentation, Pattern Recognition 142 (2023) 109651.

[28] T. Song, S. Pang, S. Hao, A. Rodríguez-Patón, P. Zheng, A parallel image skeletonizing method using spiking neural P systems with weights, Neural Processing Letters 50 (2019) 1485–1502.

[29] T. Wang, W. Liu, L. V. Cabrera, P. Wang, X. Wei, T. Zang, A novel fault diagnosis method of smart grids based on memory spiking neural P systems considering measurement tampering attacks, Information Sciences 596 (2022) 520–536.

[30] H. Peng, J. Wang, J. Ming, P. Shi, M. J. Pérez-Jiménez, W. Yu, C. Tao, Fault diagnosis of power systems using intuitionistic fuzzy spiking neural P systems, IEEE transactions on smart grid 9 (5) (2017) 4777–4784.

[31] G. Zhang, H. Rong, F. Neri, M. J. Pérez-Jiménez, An optimization spiking neural P system for approximately solving combinatorial optimization problems, International Journal of Neural Systems 24 (05) (2014) 1440006.

[32] J. Dong, G. Zhang, B. Luo, H. Rong, An optimization numerical spiking neural P system for solving constrained optimization problems, Information Sciences 626 (2023) 428–456.

[33] Y. Huang, Q. Liu, H. Peng, J. Wang, Q. Yang, D. Orellana-Martín, Sentiment classification using bidirectional LSTM-SNP model and attention mechanism, Expert Systems with Applications 221 (2023) 119730.

[34] Q. Liu, L. Long, H. Peng, J. Wang, Q. Yang, X. Song, A. Riscos-Núñez, M. J. Pérez-Jiménez, Gated spiking neural P systems for time series forecasting, IEEE Transactions on Neural Networks and Learning Systems.

[35] Y. Zhang, Q. Yang, Z. Liu, H. Peng, J. Wang, A prediction model based on gated nonlinear spiking neural systems, International Journal of Neural Systems 33 (06) (2023) 2350029.

[36] Q. Liu, H. Peng, L. Long, J. Wang, Q. Yang, M. J. Pérez-Jiménez, D. Orellana-Martín, Nonlinear spiking neural systems with autapses for predicting chaotic time series, IEEE Transactions on Cybernetics (2023) 1–13.

[37] Y. Ma, Y. Zhang, Y. Wang, Q. Wang, X. Ma, Study of the effect of membrane thickness on microcapsule strength, permeability, and cell proliferation, Journal of Biomedical Materials Research Part A 101A (2013) 1–9.

[38] M. L. Minsky, Computation: finite and infinite machines, Prentice-Hall, Inc., 1967.

[39] R. G. Michael, S. J. David, Computers and Intractability: A Guide to the Teory of NP-Completeness, WH Freeman and Company, New York, 1979.

[40] L. Wang, X. Liu, Y. Zhao, Universal nonlinear spiking neural P systems with delays and weights on synapses, Computational Intelligence and Neuroscience 2021 (2021) 1–15.

[41] X. Tian, X. Liu, Q. Ren, Y. Zhao, Spiking neural P systems with enzymes, IEEE Transactions on NanoBioscience 21 (4) (2022) 575–587.

[42] L. Zhang, F. Xu, Spiking neural P systems with cooperative synapses, Neurocomputing 501 (2022) 222–230.

[43] F. G. C. Cabarle, R. T. A. D. L. Cruz, D. P. P. Cailipan, D. Zhang, X. Zeng, On solutions and representations of spiking neural P systems with rules on synapses, Information Sciences 501 (2019) 30–49.

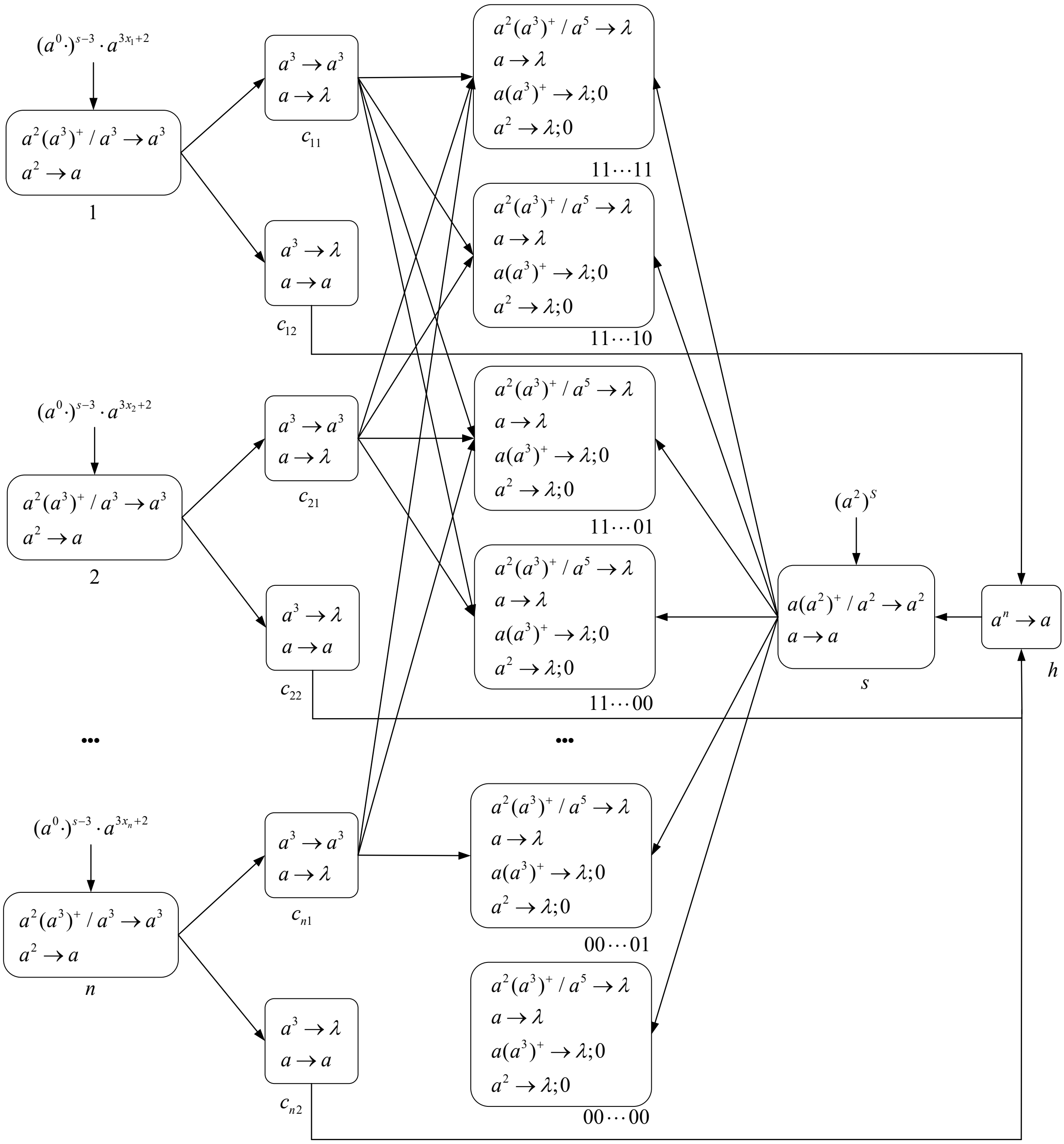

Figure 7: NP-SNP system Π_s for solving Subset Sum problem

Declaration of Interest Statement

The authors declare that there are no conflicts of interest in the implementation of this work.