

Received 21 August 2023, accepted 2 January 2024, date of publication 12 January 2024,
date of current version 19 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3353619

RESEARCH ARTICLE

On the Universality of Spiking Neural P Systems With Multiple Channels and Autapses

XIN NING^{1,2}, GUANYI YANG^{3,4}, ZHANG SUN^{1,3,4,5}, AND XIAOXIAO SONG^{1,3,4}

¹State Grid Sichuan Electric Power Research Institute, Sichuan 610041, China

²Power Internet of Things Key Laboratory of Sichuan Province, Sichuan 610041, China

³School of Electrical Engineering and Electronic Information, Xihua University, Chengdu, Sichuan 610039, China

⁴Key Laboratory of Fluid and Power Machinery, Ministry of Education, Xihua University, Chengdu, Sichuan 610039, China

⁵School of Electrical Engineering, Southwest Jiaotong University, Chengdu, Sichuan 611756, China

Corresponding author: Xiaoxiao Song (songxx_xhu@163.com)

This work was supported in part by the Sichuan Provincial Department of Science and Technology under Grant 2022YFS0538, Grant 2022NSFSC0025, Grant 2021YFG0204, Grant 2022YFG0117, and Grant 2022YFG0061.


ABSTRACT Spiking neural P systems (SN P systems) abstract the structure and function of neurons and nervous systems. By adopting some biological observations or mathematical considerations in SN P systems, many variants have been proposed. In this work, through combining two interesting variants, SN P systems with multiple channels (SNP-MC systems) and SN P systems with autapses (SNP-AU systems), we propose SN P systems with multiple channels and autapses (SNP-MCA systems) and research their universality. SNP-MCA systems are proved to be universal in the generation of numbers. For computing functions, a universal SNP-MCA system with 25 neurons is constructed, which requires fewer neurons than the two variants.

INDEX TERMS Membrane computing, spiking neural P systems, multiple channels, autapses, universality.

I. INTRODUCTION

Membrane computing (also called P systems) simulates the structure, function, and communication of cells in animals [1]. P systems can be classified as cell-like [2], [3], [4], tissue-like [5], [6], [7], and neural-like. In 2003, as considered by Web of Sciences, membrane computing was a fast-rising topic in computer science, and in 2020, it was incorporated in the AMS Mathematics Subjects Classification under Computer Science.

Spiking neural P systems (SN P systems) were proposed by Ionescu et al. in 2006, and are regarded as a type of neural-like P system. Directed graphs are usually used to describe SN P systems, where nodes are neurons, and directed arcs are synapses (connecting neurons). Some rules and spikes are usually contained in neurons, where spikes can also be transmitted through synapses. There are generally two kinds of rules in neurons: *spiking rules* and *forgetting rules*. *Spiking rules* have the form $E/a^c \rightarrow a^p; d$, where E represents the firing condition, $c(p)$ is the number of spikes (denoted by a)

The associate editor coordinating the review of this manuscript and approving it for publication was Claudia Raibulet .

the rule consumes (generates), and d is the delay time; and *forgetting rules* remove c spikes and have the form $a^c \rightarrow \lambda$. In the last 15 years, much research has been focused on SN P systems, and some variants have been proposed.

SN P systems can generate and accept any recursively enumerable set of numbers [8], can compute any Turing-computable function [9], are capable of generating some languages [10], and solving some practical problems [11].

Some variants based on biological observations have been proposed, such as those with rules using the computational power of synapses [12], [13], which are moved from neurons onto synapses. SN P systems with delays on synapses were proposed by considering the length of axons [14], where postsynaptic neurons get spikes at different times. By introducing astrocytes, a kind of special cell in neuron systems, the spike transmission in synapses can be controlled [15], [16], and SN P systems with weights have been proposed for multiple connections between two neurons, [17], [18]. We use the weight w to denote w synapses between neurons. SN P systems with white-hole rules [19], [20] have been proposed based on neural information rejection. When neurons fire, all the spikes are consumed.

Some variants are inspired by mathematical/computer considerations. In SN P systems with thresholds [21], [22], neurons fire when they have more spikes than the threshold. Inspired by the concept of “positive” and “negative” numbers, SN P systems with anti-spikes were researched [23], [24]. Spikes and anti-spikes cannot coexist in a neuron. Fuzzy SN P systems use fuzzy logic and reasoning in spiking rules [25]. Adopting request-response methods from computers in communication, SN P systems with communication on request [26] and with request rules [27] were researched. Analogous to homogeneous units in computer science, homogeneous SN P systems were proposed, where neurons [28] and even synapses [29] are unified.

Most such variants consider a global clock, i.e., they work in synchronous mode. It has been proved that SN P systems can also work in asynchronous or sequential mode, where enabled rules can be kept unfired in asynchronous mode [30], [31], and only one rule can and must be used in sequential mode, no matter how many available rules exist [32], [33].

Two kinds of biologically inspired variants were proposed in recent years: SN P systems with multiple channels (SNP-MC systems) (2017) [34], [35], [36] and SN P systems with autapses (SNP-AU systems) (2021) [37]. SNP-MC systems introduce biological observations of multiple ion channels, whose labels distinguish synapses. The spiking rules have the form $E/a^c \rightarrow a^p(l)$, where the generated p spikes are transmitted through synapses labeled l . Autapses are one kind of special synapse, where synapses lead spikes back to themselves. For SNP-AU systems, we remove the constraint $i \neq j$ in the definition of synapses; hence one neuron can transmit spikes to itself.

We propose a variant of SN P systems, *spiking neural P systems with multiple channels and autapses* (SNP-MCA systems), and research their computational power both in generating numbers and computing functions.

The remainder of this work is structured as follows. Some background knowledge is introduced in section II. SNP-MCA systems are defined in section III. We prove the universality of SNP-MCA systems in generating numbers in section IV, and build a small universal SNP-MCA system in computing functions in section V. Section VI reviews our conclusions.

II. PRELIMINARIES

We relate some knowledge in automata theory and register machines.

In sections IV and V, the register machine is used in proving the universality of SNP-MCA systems. It has the form $M = (m, H, l_0, l_h, I)$, where m is the number of registers, H is the set of instruction labels, l_0 is a starting label, l_h is a halting label, and I represents the instruction set. There are *ADD*, *SUB*, and *HALT* instructions with respective forms $l_i : (ADD(r), l_j, l_k)$, $l_i : (SUB(r), l_j, l_k)$, and $l_h : HALT$. Instruction $l_i : (ADD(r), l_j, l_k)$ increases the number stored in r by one, and activates either l_j or l_k . Instruction $l_i : (SUB(r), l_j, l_k)$ subtracts one from r , activates l_j if r is not empty, and

$$\begin{aligned}
 l_0 &: (SUB(1), l_1, l_2), & l_1 &: (ADD(7), l_0), \\
 l_2 &: (ADD(6), l_3), & l_3 &: (SUB(5), l_2, l_4), \\
 l_4 &: (SUB(6), l_5, l_3), & l_5 &: (ADD(5), l_6), \\
 l_6 &: (SUB(7), l_7, l_8), & l_7 &: (ADD(1), l_4), \\
 l_8 &: (SUB(6), l_9, l_0), & l_9 &: (ADD(6), l_{10}), \\
 l_{10} &: (SUB(4), l_0, l_{11}), & l_{11} &: (SUB(5), l_{12}, l_{13}), \\
 l_{12} &: (SUB(5), l_{14}, l_{15}), & l_{13} &: (SUB(2), l_{18}, l_{19}), \\
 l_{14} &: (SUB(5), l_{16}, l_{17}), & l_{15} &: (SUB(3), l_{18}, l_{20}), \\
 l_{16} &: (ADD(4), l_{11}), & l_{17} &: (ADD(2), l_{21}), \\
 l_{18} &: (SUB(4), l_0, l_h), & l_{19} &: (SUB(0), l_0, l_{18}), \\
 l_{20} &: (ADD(0), l_0), & l_{21} &: (ADD(3), l_{18}), \\
 l_h &: HALT
 \end{aligned}$$

FIGURE 1. Universal register machine M_u .

otherwise activates l_k . The instruction $l_h : HALT$ represents the end of computation.

Register machines have been proved to characterize *NRE* [38], which means they can generate Turing-computable numbers, and deterministic register machines can compute Turing-computable functions, where *ADD* has the form $l_i : (ADD(r), l_j)$. A register machine $M_u = (8, H, l_0, l_h, I)$ has 8 registers and 23 labels, as shown in Figure 1 [39]. With input $g(x)$ in register 1 and y in register 2, M_u can compute the function $\varphi_x(y) = M_u(g(x), y)$, whose result is stored in register 0.

III. SPIKING NEURAL P SYSTEMS WITH MULTIPLE CHANNELS AND AUTAPSES

An SNP-MCA system is defined as a tuple,

$$\Pi = (O, L, \sigma_1, \dots, \sigma_m, syn, in, out),$$

where

- 1) $O = \{a\}$ represents the alphabet and a denotes a spike;
- 2) $L = \{1, \dots, N\}$ represents the set of channel labels;
- 3) $\sigma_i = (n_i, R_i)$ stands for the i^{th} neuron, $1 \leq i \leq m$, where
 - (a) n_i is the initial number of spikes in σ_i ;
 - (b) R_i , in the form of $E/a^c \rightarrow a^p(l)$, denotes the rules in σ_i , where E is a regular expression over O , and l is the channel label ($l \in L$), $c \geq p \geq 0$;
- 4) The synapses are represented by $syn \subseteq \{1, 2, \dots, m\} \times \{1, 2, \dots, m\} \times L \times K$, where K is the set of numbers of synapses; hence a synapse is denoted by (i, j, l, k) , where $1 \leq i, j \leq m$, and $k \in K$ is the number of synapses between neurons σ_i and σ_j ;
- 5) in and out stand for the input and output neuron, respectively.

In SNP-MCA systems, rule $E/a^c \rightarrow a^p(l)$ fires when $a^n \in L(E)$, and $n(n \geq c)$ is the number of spikes in σ_i . In this way, using c spikes, $p \times k > 0$ spikes are sent to the postsynaptic neurons $\sigma_j \in \{\sigma_j | (i, j, l, k) \in syn\}$. Additionally, rules can be expressed as $a^c \rightarrow a^p(l)$ for brevity when $n = c$. When $p = 0$, after eliminating c spikes from neurons, no spikes are sent out from neuron σ_i . Thus, channel labels are omitted

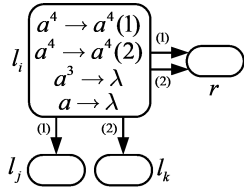


FIGURE 2. ADD module based on SNP-MCA system.

when $p = 0$, and rules are represented by $E/a^c \rightarrow \lambda$. For simplicity, when there is only one synapse labeled l between neurons (or one autapse labeled l on a neuron), the expression $k = 1$ can be omitted in directed graphs.

Autapses are special synapses that can lead the generated spikes back to a neuron. Thus, the classical constraint $(i, i) \notin \text{syn}$, as adopted in many SN P systems, is removed, and in SNP-MCA systems, i can equal j when a synapse has the form (i, j, l, k) .

Similar to most SN P systems, SNP-MCA systems work in synchronous mode, and rules work sequentially in a single neuron. Thus, at each instant, one rule must be used if it is the only one available in a neuron, and if more than one rule is available, one of them is used non-deterministically.

In this work, $N_{gen}(\Pi)$ denotes a Turing-computable number generated by Π , and $N_{gen}(SNP_m^n MC_{lA_a})$ stands for numbers generated by SNP-MCA systems with m neurons, each with up to n rules, l channels, and a autapses.

IV. COMPUTATIONAL COMPLETENESS

We prove that SNP-MCA systems are capable of generating all sets of NRE (The family of recursively enumerable languages is denoted by RE and the family of Turing computable sets of natural numbers is denoted by NRE).

Theorem 1: $N_{gen}(SNP_m^4 MC_{2A_1}) = NRE$.

Proof: It is straightforward that $N_{gen}(SNP_m^4 MC_{2A_1}) \subseteq NRE$. Hence we focus on proving $N_{gen}(SNP_m^4 MC_{2A_1}) \supseteq NRE$. We construct an SNP-MCA system with ADD, SUB, and OUTPUT modules to simulate a register machine. $4n$ spikes in neuron σ_r corresponds to the value n stored in register r . * means the SNP-MCA system is constructed with an unbounded number of neurons.

A. ADD MODULE

The ADD module in Figure 2 simulates the ADD instruction of the SNP-MCA system in the form of $l_i : (ADD(r), l_j, l_k)$.

Let us assume exactly four spikes exist in σ_{l_i} at a certain instant. Then two rules are available: $a^4 \rightarrow a^4(1)$ and $a^4 \rightarrow a^4(2)$. One of them will be used non-deterministically. If the first is selected, four spikes are sent to both neuron σ_{l_i} (denoting instruction l_i) and neuron σ_r (denoting register r), simultaneously, through synapses $(l_i, l_j, 1, 1)$ and $(l_i, r, 1, 1)$. In this way, the number stored in register r is increased by one, and instruction l_j is activated. Similarly, if the second rule is selected, then the number is increased by one, and instruction l_k is enabled.

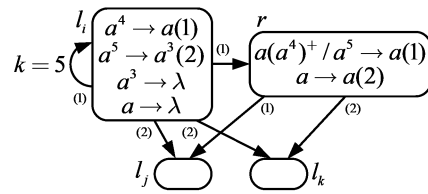


FIGURE 3. SUB module based on SNP-MCA system.

TABLE 1. Computation of SUB module.

Instant	Register r is not empty				Register r is empty			
	l_i	r	l_j	l_k	l_i	r	l_j	l_k
t	4	$4n$	0	0	4	0	0	0
$t+1$	5	$4n+1$	0	0	5	1	0	0
$t+2$	0	$4(n-1)$	4	3	0	0	3	4
$t+3$	0	$4(n-1)$	(4)	0	0	0	0	(4)

B. SUB MODULE

The SUB instruction has the form $l_i : (SUB(r), l_j, l_k)$, and is shown in Figure 3.

With four spikes at instant t , using rule $a^4 \rightarrow a(1)$, a spike is sent from neuron σ_{l_i} to neurons σ_r and σ_{l_i} through synapse $(l_i, r, 1, 1)$ and autapse $(l_i, l_i, 1, 5)$, respectively. Thus, neuron σ_{l_i} gets five spikes and σ_r gets one spike at the next instant, and $a^5 \rightarrow a^3(2)$ in neuron σ_{l_i} is available, which produces three spikes and sends them to neurons σ_{l_j} and σ_{l_k} . Depending on the number of spikes stored in register r , there are two possible computations:

- 1) If $4n + 1$ ($n \geq 1$) spikes are contained in register r at instant $t + 1$, rule $a(a^{4n})^+ / a^5 \rightarrow a(1)$ is active. A single spike is sent to neuron σ_{l_i} after consuming five spikes (only $4(n - 1)$ spikes are left). Thus, the number in register r is reduced by one. With this spike and the three spikes gotten from neuron σ_{l_j} , l_j is active, and the remaining spikes in neuron σ_{l_k} are removed by the rule $a^3 \rightarrow \lambda$.
- 2) If the register contains a single spike at instant $t + 1$, rule $a \rightarrow a(2)$ is enabled, the generated spike is sent to neuron σ_{l_k} , and instruction l_k is active at the next instant.

Table 1 shows the computation of the SUB module. The two numbers in parentheses at the bottom of the table indicate that four spikes should be changed at this instant. However, these operations are the computation of some subsequent instruction.

We can find there would be some interference between the SUB modules, as shown in Figure 3. If one register could be used by more than one instruction, the register would send spikes to more than one post-synaptic neuron. For example, SUB instructions $l_4 : (SUB(6), l_5, l_3)$ and $l_8 : (SUB(6), l_9, l_0)$ both act on register 6. If instruction l_4 is active, the generated spike is sent to neuron σ_{l_5} (neuron σ_{l_3}) if register 6 is not

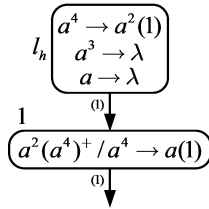


FIGURE 4. OUTPUT module based on SNP-MCA system.

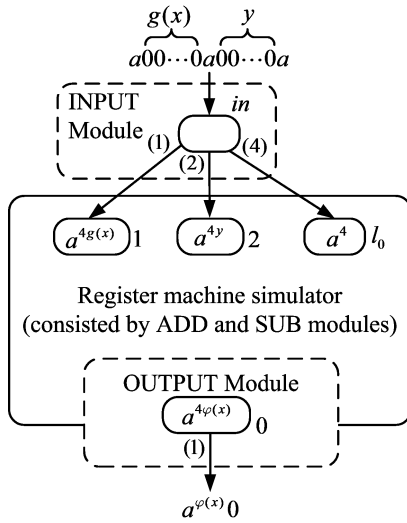


FIGURE 5. Universal SNP-MCA system.

empty (empty) one instant later. Thus, with the three spikes obtained from neuron σ_{l_4} , four spikes will be contained in neuron σ_{l_5} (neuron σ_{l_3}), which enables instruction l_5 (l_3). One spike, as the interference, will also be simultaneously sent from neuron σ_6 to neuron σ_{l_0} (neuron σ_{l_0}). Different from neuron σ_{l_5} (neuron σ_{l_3}), no other spikes will arrive at neuron σ_{l_0} (neuron σ_{l_0}). Thus, the single spike will be removed by rule $a \rightarrow \lambda$, and no undesired effect will appear.

C. OUTPUT MODULE

When instruction l_h gets four spikes, the computation comes to an end, and the result will be output through the OUTPUT module, as shown in Figure 4.

After using four spikes, neuron σ_1 gets two generated spikes from neuron σ_h , enabling rule $a^2(a^{4n})^+ / a^4 \rightarrow a(1)$. In this way, one spike is output for every four spikes, and the result is the number of spikes that are sent out.

It can be seen that SNP-MCS systems can simulate register machines. Hence Theorem 1 holds.

V. SMALL UNIVERSAL SNP-MCA SYSTEM FOR COMPUTING FUNCTIONS

We build an SNP-MCA system based on a strong M_u [39].

Theorem 2: A small universal SNP-MCA system with 25 neurons can be built to compute any set of Turing-computable functions.

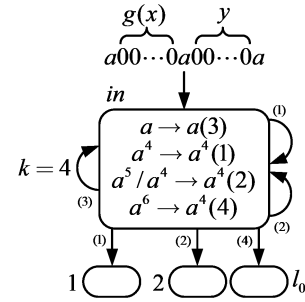


FIGURE 6. INPUT module based on SNP-MCA system in computing functions.

Proof: Figure 5 gives the framework of the SNP-MCA system, which includes four classes of modules. Similar to the previous section, $4n$ spikes stand for the number n in registers.

A. INPUT MODULE

The INPUT module introduces $a(0)^{g(x)}a(0)^{y-1}a$ to the system and enables instruction l_0 , where $(a)^{g(x)}$ means that only one spike arrives at neuron σ_{in} at each step within $g(x)$ steps. Because four spikes denote the number one, $4g(x)$ and $4y$ spikes should be loaded into neurons σ_1 (denoting register 1) and σ_2 (register 2), respectively. The module is shown in Figure 6.

When one spike arrives at neuron σ_{in} , using the rule $a \rightarrow a(3)$, a spike is sent through autapse $(in, in, 3, 4)$. Then four spikes are sent back to neuron σ_{in} , which enables rule $a^4 \rightarrow a^4(1)$. Four spikes are transmitted to neuron σ_1 through synapse $(in, 1, 1, 1)$, and four spikes simultaneously return to neuron σ_{in} through autapse $(in, in, 1, 1)$. Thus, register 1 gets four spikes at each instant until the second spike gets to the neuron. In this way, register 1 contains $4g(x)$ spikes. After the second spike arrives from the environment, the rule $a^5/a^4 \rightarrow a^4(2)$ is available. Similarly, at each instant, neuron σ_2 gets four spikes through synapse $(in, 2, 2, 1)$, and four spikes return to neuron σ_{in} through $(in, in, 2, 1)$. When the third spike arrives, a total of $4y$ spikes are contained in register 2. In this way, the rule $a^6 \rightarrow a^4(4)$ is enabled, which consumes the last six spikes in neuron σ_{in} , sends four spikes to neuron σ_{l_0} through synapse $(in, l_0, 4, 1)$, and activates instruction l_0 . This means that the system begins to compute.

B. ADD MODULE

Different from non-deterministic mode, in computing mode, after adding one to register r , the ADD module directly activates instruction l_j . Thus, with the module in Figure 7, $l_i : (ADD(r), l_j)$ is realized successfully.

For computing functions, the SUB module in a small universal SNP-MCA system is the same as the module in Figure 3. The OUTPUT module in computing functions is similar to the OUTPUT module in generating numbers in Figure 4. The difference is that the computation result is stored in register 0 instead of register 1.

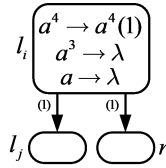


FIGURE 7. ADD module based on SNP-MCA system in computing functions.

TABLE 2. Number of neurons in building the small universal SNP-MCA system.

Modules and registers	Required number of neurons
INPUT	1
ADD	9
SUB	13
OUTPUT	1
registers	8
Sum	32

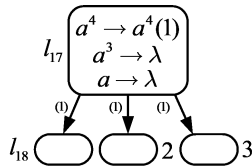


FIGURE 8. Instructions $l_{17}(ADD(2), l_{21})$ and $l_{21}(ADD(3), l_{18})$.

Thus, we can find the SNP-MCA system has a total of 32 neurons: eight neurons corresponding to the eight registers, 23 neurons for the 23 instruction labels, and 1 additional neuron in the INPUT module.

With some analysis, we can find that some optimization modules can be used to save neurons.

C. ADD-ADD

In the small universal register machine proposed by Korec [39], there is only one pair of consecutive ADD instructions, l_{17} and l_{21} , whose function can be realized by a single ADD-ADD module, as shown in Figure 8.

When four spikes arrive at neuron $\sigma_{l_{17}}$, rule $a^4 \rightarrow a^4(1)$ is enabled. Then four spikes are sent through synapses $(l_{17}, l_{18}, 1, 1)$, $(l_{17}, 2, 1, 1)$, and $(l_{17}, 3, 1, 1)$. Simultaneously, neurons $\sigma_{l_{18}}$, σ_2 , and σ_3 each get four spikes at the next instant. Thus, the numbers in registers 2 and 3 are increased by one, and instruction l_{18} is active. In this way, the ADD-ADD module realizes the function of these two instructions, saving neuron $\sigma_{l_{21}}$.

When we combine the consecutive SUB $(l_i : (SUB(r_1), l_j, l_k))$ and ADD $(l_l : (ADD(r_2), l_g))$ instructions, there are two cases: $l_j = l_l$ and $l_k = l_l$.

D. SUB-ADD-1: $l_j = l_l$

In the first case, $l_j = l_l$; hence, after reducing the number in register r_1 by one (assuming it is not empty), the number in register r_2 is increased by one and l_g is enabled. If register r_1 is

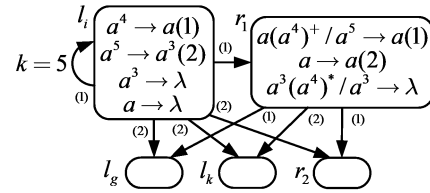


FIGURE 9. SUB-ADD-1 module.

TABLE 3. Computation of SUB-ADD-1 module.

Instant	Register r_1 is not empty					Register r_1 is empty				
	l_i	r_1	l_g	l_k	r_2	l_i	r_1	l_g	l_k	r_2
t	4	$4n$	0	0	$4m$	4	0	0	0	$4m$
$t+1$	5	$4n+1$	0	0	$4m$	5	1	0	0	$4m$
$t+2$	0	$4n-4$	4	3	$4m+4$	0	0	3	4	$4m+3$
$t+3$	0	$4n-4$	(4)	0	$4m+4$	0	0	0	(4)	$4m$

empty, instruction l_k is activated, with no changes in registers r_1 and r_2 . We adopt the SUB-ADD-1 module in Figure 9 to carry out these pairs of instructions.

When four spikes reach neuron σ_{l_i} at instant t , one spike is sent to register r_1 and to itself through synapse $(l_i, r_1, 1, 1)$ and autapse $(l_i, l_i, 1, 5)$. Thus, one spike arrives at neuron σ_{r_1} , and five spikes arrive at neuron σ_{l_i} (with weight $k = 5$ on the autapse) at instant $t + 1$. With the five spikes in neuron σ_{l_i} , rule $a^5 \rightarrow a^3(2)$ is enabled, which sends three spikes to neurons σ_{l_g} , σ_{l_k} , and σ_{r_2} at the next instant. For neuron σ_{r_1} , a different rule would be available.

- 1) If neuron σ_{r_1} has $4n + 1$ ($n \geq 1$) spikes, $a(a^4)^+ / a^5 \rightarrow a(1)$ is enabled. After using five spikes, a spike is sent to both σ_{l_g} and r_2 through synapses $(r_1, l_g, 1, 1)$ and $(r_1, r_2, 1, 1)$. With the three spikes from σ_{r_1} , a total of four spikes arrive at these two neurons at instant $t + 2$. Thus, the number stored in register r_2 is increased, and instruction l_g is activated. The only three spikes in neuron σ_{l_k} , obtained from neuron σ_{r_1} , are removed by using rule $a^3 \rightarrow \lambda$ at the next instant.
- 2) If the neuron has only one spike, then register r_1 had no spikes at the previous instant. One spike is transmitted to σ_{l_k} using rule $a \rightarrow a(2)$. Thus, neuron σ_{l_k} gets a total of four spikes, which activate instruction l_k . Neurons σ_{l_g} and σ_{r_2} each have only three spikes, which are eliminated by using rules $a^3 \rightarrow \lambda$ and $a^3(a^4)^* / a^3 \rightarrow \lambda$, respectively.

To demonstrate the function of the SUB-ADD-1 module, Table 3 shows its computation.

Through the above analysis, we can see that instructions $l_i : (SUB(r_1), l_j, l_k)$ and $l_l : (ADD(r_2), l_g)$, where $l_j = l_l$, are simulated by the optimization module. As five pairs of these consecutive instructions can be found in M_u (l_0 and l_1 , l_4 and l_5 , l_6 and l_7 , l_8 and l_9 , and l_{14} and l_{16}), neurons σ_{l_1} , σ_{l_5} , σ_{l_7} , σ_{l_9} , and $\sigma_{l_{16}}$ can be saved.

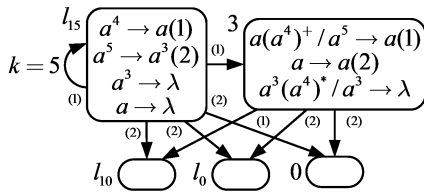


FIGURE 10. SUB-ADD-2 module.

TABLE 4. Number of neurons saved with optimization modules.

Modules	Number of Saved neurons
ADD-ADD	-1
SUB-ADD-1	-5
SUB-ADD-2	-1
Sum	-7

TABLE 5. Number of neurons in universal SN P systems and some variants in computing functions.

Variants of SN P systems	Number of neurons
SN P systems [9]	41
SNP-MC systems [40]	39
SNP-AU systems [37]	53
SNP-MCA systems	25

E. SUB-ADD-2: $l_k = l_l$

If $l_k = l_l$, only one pair of instructions can be found in M_u : l_{15} and l_{20} . Similar to SUB-ADD-1, the function of these instructions can be carried out by the SUB-ADD-2 module in Figure 10, and neuron $\sigma_{l_{20}}$ can be removed.

By adopting these three optimization modules, seven neurons can be saved.

Thus, for computing functions, 25 neurons are adopted to build the small universal SNP-MCA system, and Theorem 2 holds.

VI. CONCLUSION

Dozens of variants of SN P systems have been studied during the last 15 years, most based on biological observations or mathematical considerations. Both SNP-MC systems [34] and SNP-AU systems [37] adopt some new biological discoveries in nervous systems and nerve cells. We combined these two mechanisms in one type of SN P system, SNP-MCA. In section IV, SNP-MCA systems were proved capable of generating any Turing-computable set of numbers. To compute functions, we constructed a universal SNP-MCA system, which has good computation ability. It requires only 25 neurons, which is fewer than SN P systems, SNP-MC systems, and SNP-AU systems.

In this work, the SNP-MCA systems worked in synchronous mode. However, it has been proved that some SN P systems can work in sequential or asynchronous mode.

SNP-MCA systems show their powerful computation ability in synchronous mode. But what about sequential or asynchronous mode? This would be interesting to explore.

Here, we only discussed the small universality of SNP-MCA systems. The borderline between universality and non-universality is also an interesting and challenging problem to research, i.e., “big non-universal” SNP-MCA systems. In addition, SNP-MCA systems have good computation ability, which provides a possible way to deal with some complex real-world problems. Therefore, our future work will focus on the application of SNP-MCA systems in some real-world problems.

REFERENCES

- [1] G. Păun, “A dozen of research topics in membrane computing,” *Theor. Comput. Sci.*, vol. 736, pp. 76–78, Aug. 2018.
- [2] S. Jiang, T. Liang, B. Xu, Z. Shen, X. Zhu, and Y. Wang, “Cell-like P systems with channel states and synchronization rule,” *Mathematics*, vol. 11, no. 1, p. 117, Dec. 2022.
- [3] Y. P. Ceon, H. C. Anandharaj, S. Jebasingh, and D. A. Chandy, “Generation of chain code pictures using cell-like spiking neural P system with several types of spikes,” *J. Membrane Comput.*, vol. 4, no. 3, pp. 243–250, Sep. 2022.
- [4] Y. Zhao, W. Zhang, M. Sun, and X. Liu, “An improved consensus clustering algorithm based on cell-like P systems with multi-catalysts,” *IEEE Access*, vol. 8, pp. 154502–154517, 2020.
- [5] D. Díaz-Pernil, H. A. Christinal, and M. A. Gutiérrez-Naranjo, “Solving the 3-COL problem by using tissue P systems without environment and proteins on cells,” *Inf. Sci.*, vols. 430–431, pp. 240–246, Mar. 2018.
- [6] Y. Luo, Y. Zhao, and C. Chen, “Homeostasis tissue-like P systems,” *IEEE Trans. Nanobiosci.*, vol. 20, no. 1, pp. 126–136, Jan. 2021.
- [7] Y. Luo, P. Guo, Y. Jiang, and Y. Zhang, “Timed homeostasis tissue-like P systems with evolutionary symport/antiport rules,” *IEEE Access*, vol. 8, pp. 131414–131424, 2020.
- [8] Z. Sun, L. Valencia-Cabrera, G. Ning, and X. Song, “Spiking neural P systems without duplication,” *Inf. Sci.*, vol. 612, pp. 75–86, Oct. 2022.
- [9] A. Păun and G. Păun, “Small universal spiking neural P systems,” *Biosystems*, vol. 90, no. 1, pp. 48–60, Jul. 2007.
- [10] L. Zhang, F. Xu, A. Ramanujan, and K. G. Subramanian, “Control languages accepted by labeled spiking neural P systems with rules on synapses,” *Theor. Comput. Sci.*, vol. 893, pp. 60–71, Nov. 2021.
- [11] M. Zhu, Q. Yang, J. Dong, G. Zhang, X. Gou, H. Rong, P. Paul, and F. Neri, “An adaptive optimization spiking neural P system for binary problems,” *Int. J. Neural Syst.*, vol. 31, no. 1, Jan. 2021, Art. no. 2050054.
- [12] F. G. C. Cabarle, R. T. A. de la Cruz, D. P. P. Cailipan, D. Zhang, X. Liu, and X. Zeng, “On solutions and representations of spiking neural P systems with rules on synapses,” *Inf. Sci.*, vol. 501, pp. 30–49, Oct. 2019.
- [13] L. Zhang and F. Xu, “Asynchronous spiking neural P systems with rules on synapses and coupled neurons,” *Knowl.-Based Syst.*, vol. 257, Dec. 2022, Art. no. 109896.
- [14] X. Song, L. Valencia-Cabrera, H. Peng, J. Wang, and M. J. Pérez-Jiménez, “Spiking neural P systems with delay on synapses,” *Int. J. Neural Syst.*, vol. 31, no. 1, Jan. 2021, Art. no. 2050042.
- [15] S. Jiang, Z. Shen, B. Xu, X. Zhu, and T. Liang, “Spiking neural P systems with polarizations and astrocytes,” *J. Membrane Comput.*, vol. 5, no. 1, pp. 55–68, Mar. 2023.
- [16] B. Aman, “Solving subset sum by spiking neural P systems with astrocytes producing calcium,” *Natural Comput.*, vol. 22, no. 1, pp. 3–12, Mar. 2023.
- [17] M. Dalvand, A. Fathi, and A. Kamran, “Spiking neural P system with weight model of majority voting technique for reliable interactive image segmentation,” *Neural Comput. Appl.*, vol. 35, no. 12, pp. 9035–9051, Dec. 2022.
- [18] Y. Liu and Y. Zhao, “Weighted spiking neural P systems with polarizations and anti-spikes,” *J. Membrane Comput.*, vol. 4, no. 4, pp. 269–283, Dec. 2022.
- [19] A. Alhazov, R. Freund, S. Ivanov, M. Oswald, and S. Verlan, “Extended spiking neural P systems with white hole rules and their red–green variants,” *Natural Comput.*, vol. 17, no. 2, pp. 297–310, Jun. 2018.

- [20] T. Song, F. Gong, X. Liu, Y. Zhao, and X. Zhang, "Spiking neural P systems with white hole neurons," *IEEE Trans. Nanobiosci.*, vol. 15, no. 7, pp. 666–673, Oct. 2016.
- [21] H. Peng, J. Wang, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "Dynamic threshold neural P systems," *Knowledge-Based Syst.*, vol. 163, pp. 875–884, Jan. 2019.
- [22] X. Zeng, X. Zhang, T. Song, and L. Pan, "Spiking neural P systems with thresholds," *Neural Comput.*, vol. 26, no. 7, pp. 1340–1361, Jul. 2014.
- [23] M.-I. Plesa, M. Gheoghe, F. Ipate, and G. Zhang, "A key agreement protocol based on spiking neural P systems with anti-spikes," *J. Membrane Comput.*, vol. 4, no. 4, pp. 341–351, Dec. 2022.
- [24] T. Ma, S. Hao, X. Wang, A. Alfonso Rodríguez-Patón, S. Wang, and T. Song, "Double layers self-organized spiking neural P systems with anti-spikes for fingerprint recognition," *IEEE Access*, vol. 7, pp. 177562–177570, 2019.
- [25] X. Yin, X. Liu, M. Sun, J. Dong, and G. Zhang, "Fuzzy reasoning numerical spiking neural P systems for induction motor fault diagnosis," *Entropy*, vol. 24, no. 10, p. 1385, Sep. 2022.
- [26] T. Song and L. Pan, "Spiking neural P systems with request rules," *Neurocomputing*, vol. 193, pp. 193–200, Jun. 2016.
- [27] L. Pan, G. Păun, G. Zhang, and F. Neri, "Spiking neural P systems with communication on request," *Int. J. Neural Syst.*, vol. 27, no. 8, Dec. 2017, Art. no. 1750042.
- [28] K. Jiang, W. Chen, Y. Zhang, and L. Pan, "Spiking neural P systems with homogeneous neurons and synapses," *Neurocomputing*, vol. 171, pp. 1548–1555, Jan. 2016.
- [29] X. Zeng, X. Zhang, and L. Pan, "Homogeneous spiking neural P systems," *Fundamenta Informaticae*, vol. 97, nos. 1–2, pp. 275–294, 2009.
- [30] S. Jiang, Y. Liu, B. Xu, J. Sun, and Y. Wang, "Asynchronous numerical spiking neural P systems," *Inf. Sci.*, vol. 605, pp. 1–14, Aug. 2022.
- [31] T. Wu, L. Zhang, Q. Lyu, and Y. Jin, "Asynchronous spiking neural P systems with local synchronization of rules," *Inf. Sci.*, vol. 588, pp. 1–12, Apr. 2022.
- [32] F. G. C. Cabarle, H. N. Adorna, and M. J. Pérez-Jiménez, "Sequential spiking neural P systems with structural plasticity based on max/min spike number," *Neural Comput. Appl.*, vol. 27, no. 5, pp. 1337–1347, Jul. 2016.
- [33] Z. Lv, Q. Yang, H. Peng, X. Song, and J. Wang, "Computational power of sequential spiking neural P systems with multiple channels," *J. Membrane Comput.*, vol. 3, no. 4, pp. 270–283, Dec. 2021.
- [34] H. Peng, J. Yang, J. Wang, T. Wang, Z. Sun, X. Song, X. Luo, and X. Huang, "Spiking neural P systems with multiple channels," *Neural Netw.*, vol. 95, pp. 66–71, Nov. 2017.
- [35] X. Song, J. Wang, H. Peng, G. Ning, Z. Sun, T. Wang, and F. Yang, "Spiking neural P systems with multiple channels and anti-spikes," *Biosystems*, vols. 169–170, pp. 13–19, Jul. 2018.
- [36] X. Song, H. Peng, J. Wang, G. Ning, and Z. Sun, "Small universal asynchronous spiking neural P systems with multiple channels," *Neurocomputing*, vol. 378, pp. 1–8, Feb. 2020.
- [37] X. Song, L. Valencia-Cabrera, H. Peng, and J. Wang, "Spiking neural P systems with autapses," *Inf. Sci.*, vol. 570, pp. 383–402, Sep. 2021.
- [38] M. Minsky, *Computation: Finite and Infinite Machines*. Hoboken, NJ, USA: Prentice-Hall, 1967.
- [39] I. Korec, "Small universal register machines," *Theor. Comput. Sci.*, vol. 168, no. 2, pp. 267–301, Nov. 1996.
- [40] X. Song, H. Peng, J. Wang, G. Ning, T. Wang, Z. Sun, and Y. Xia, "On small universality of spiking neural P systems with multiple channels," in *Proc. Int. Conf. Membrane Comput.*, Sep. 2018, pp. 229–245.



XIN NING received the B.Sc. degree in hydropower engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2009, and the Ph.D. degree in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 2015.

He has been an Engineer with the State Grid Sichuan Electric Power Research Institute, since 2015. He has been a Senior Engineer, since 2018. His research interests include health status evaluation of power equipment, prevent and mitigate natural disasters, and optimization algorithm.



GUANYI YANG received the B.Sc. degree in electronic science and technology from Hangzhou Dianzi University, Hangzhou, China, in 2020. She is currently pursuing the M.E. degree in electronic information with the School of Electrical and Electronic Information, Xihua University, Chengdu, China.

Her research interests include membrane computing, image processing, and deep learning.



ZHANG SUN received the B.Sc. degree in electrical engineering from Hubei Minzu University, Enshi, China, in 2010, and the M.E. degree in electrical engineering from Xihua University, Chengdu, China, in 2013. He is currently pursuing the Ph.D. degree in electrical engineering with Southwest Jiaotong University, Chengdu.

He has been a Lecturer with the School of Electrical Engineering and Electronic Information, Xihua University, since 2015. He has been an Associate Professor, since 2021. His research interests include power converter, DC-microgrid, and hybrid energy storage technology.



XIAOXIAO SONG received the B.Sc. degree in automation from the University of Electronic Science and Technology of China, Chengdu, China, in 2003, the M.E. degree in automation from Guangxi University, Nanning, China, in 2007, and the Ph.D. degree in automation from Chongqing University, Chongqing, China, in 2010.

He was a Lecturer with Xihua University, China, from 2011 to 2017. He has been an Associate Professor with the School of Electrical Engineering and Electronic Information, Xihua University, from 2018 to 2022. He has been a Professor, since 2023. His research interests include membrane computing, intelligent control, and optimization algorithm.

• • •