On the Computational Power of Asynchronous Axon Membrane Systems

Tao Song, *Senior Member, IEEE*, Pan Zheng, *Senior Member, IEEE*, M. L. Dennis Wong, *Senior Member, IEEE*, Min Jiang, *Senior Member, IEEE*, and Xiangxiang Zeng, *Senior Member, IEEE*

***Abstract*—Axon membrane systems, also called axon P systems, are a group of neuron system inspired neural computing devices. The system are designed by the mimic of the way axon (connecting neurons in central nerves systems) processing impulse signals passing along it. In the systems, all the "computing units" are aligned one after another along the axon, achieving a linear topological structure. It was known that synchronous axon P systems can compute the families of Turing computable sets of both natural numbers and recursive functions. However, the computational power of asynchronous axon P systems is still open. In this paper, we investigate the computational power of asynchronous axon P systems, where the nonsynchronization is induced by either the node's asynchronously spiking (working in asynchronous mode) or the randomly assigned time consumption for each time spiking of the nodes (working in time-free mode). As results, it is proved that axon P systems working in either asynchronous or time-free mode are Turing universal as number generators, which indicates that the nonsynchronization will not reduce the computation power of axon P systems. It is worth noting that it needs $O(n)$ spikes to encode natural number n in asynchronous axon P systems, but it needs $O(n^2)$ spikes in Turing universal synchronous axon P systems. These results partially answer an open problem left in [IEEE NNLS 26(11): 2816-29, 2015], and may also provide some hints on designing novel learning strategies by imposing computation tasks on the synapses of neural networks models.**

Manuscript received August 17, 2018; revised December 2, 2018 and February 3, 2019; accepted February 18, 2019. The work of T. Song was supported in part by the Tai Shan Scholar Foundation, National Natural Science Foundation of China under Grant 61873280, Grant 61672033, and Grant 61672248; in part by the Key Research and Development Program of Shandong Province under Grant 2017GGX10147, in part by the Natural Science Foundation of Shandong Province under Grant ZR2017MF004, in part by the Fundamental Research Funds for the Central Universities under Grant 18CX02152A, Project TIN2016-81079-R (MINECO AEI/FEDER, Spain-EU), and the InGEMICS-CM Project (B2017/BMD-3691, FSE/FEDER, Comunidad de Madrid-EU), Research Project TIN2016-81079-R (AEI/FEDER, Spain-EU), and Grant 2016-T2/TIC-2024 from Talento-Comunidad de Madrid. The work of X. Zeng was supported in part by the Juan de la Cierva position (code: IJCI-2015-26991), and in part by the President Fund of Xiamen University under Grant 20720170054. *(Corresponding author: Xiangxiang Zeng.)*

T. Song is with the College of Computer and Communication Engineering, China University of Petroleum, Qingdao 266580, China, and also with the Department of Artificial Intelligence, Polytechnical University of Madrid, Madrid 28660, Spain (e-mail: tsong@upm.es).

P. Zheng is with the Department of Accounting and Information Systems, University of Canterbury, Christchurch 8041, New Zealand (e-mail: pzheng@swinburne.edu.my).

M. L. D. Wong is with the Heriot-Watt University, Putrajaya 62200, Malaysia (e-mail: wongmld@ieee.org).

M. Jiang is with the Department of Cognitive Science and Technology, Xiamen University, Xiamen 361005, China (e-mail: minjiang@xmu.edu.cn).

X. Zeng is with the Department of Computer Science, Xiamen University, Xiamen 361005, China, and also with the Department of Artificial Intelligence, Polytechnical University of Madrid, Madrid 28660, Spain (e-mail: xzeng@xmu.edu.cn).

Digital Object Identifier 10.1109/TETCI.2019.2907724

***Index Terms*—Natural computing, membrane computing, axon membrane system, computational power, universality, nonsynchronization.**

I. Introduction

HUMAN brain, known as "born to be powerful", provides rich ideas for computer scientists to develop powerful and practical brain-inspired computing devices and models, such as peering CNNs (Convolutional neural networks) [1] and classical neural networks models [2], [3]. Neural-like computing models are computing devices inspired from the way neurons communicating by means of spikes. A neural network can be achieved or built by connecting a series of computing units or neurons. Neural networks, i.e., neural-like computing models gain their popularity for its learning functions [4]–[9] as well as their successful applications in solving problems in practice [10]–[13]. Among the the computing units or neurons, information (represented by a number or a stack of impulse or spikes) can pass from one to another [14]–[16].

Membrane computing, initiated in 2002, is an attractive branch of bio-inspired computing. It aims to design computing models with biological information processing intelligence [17], [18]. Computing models studied in membrane computing are usually named membrane systems or P systems. There are two main classes of neural-like membrane systems: spiking neural P systems (SN P systems) and axon P systems.

– Spiking neural P systems are known as a novel candidate of the third generation of spiking neural networks [19]. The systems are proposed by modelling the way of biological neurons firing and communicating via electrical impulse or spikes [20]–[22]. Research on new variants of SN P systems is formulated in [23] as a promising branch in membrane computing [24]–[29].
– Axon P systems are obtained in a constructive way by mimic axon processing and transmitting information of impulse signals (spikes) passing along it [30]–[32].

It is shown in Figure 1 the biological structure of a neuron, where a number of computing units, namely Ranvier nodes, are with linear structure along with a axon.

In neural-like computing models, axons and synapses are abstracted as uniform edges among each pair of connecting neurons, which are only used for transmitting the spikes, and the weights (the link strength denoted by real numbers) on the synapses can be updated during the computation by different

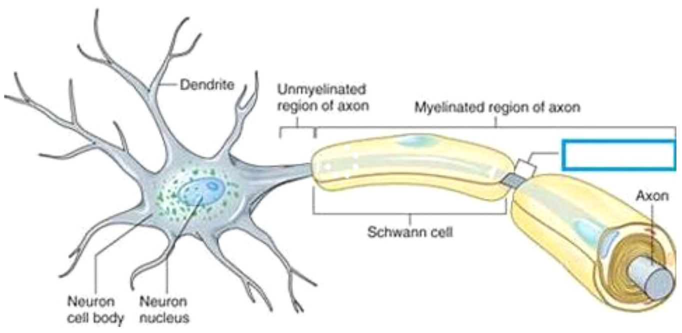

Fig. 1. The biological structure of a neuron from [33].

learning strategies. (Significant works and some peering works on learning approaches from data have been heavily investigated, see e.g. [34]–[40].) However, this is not exactly biological fact. Besides transmitting spikes, an axon has also the function of processing the spikes passed along it. Specifically, when a number of spikes pass from one Ranvier node to its left and right neighboring node, the spikes can be reduced or increased when they arrive at the target nodes, which depends on the excitatory or inhibitory function passing Ranvier node [41], [42]. Axon P systems were initialized and developed with the biological facts that Ranvier node process the spikes transmitting along axons [30]. The system has a finite set of computing units (denoting Ranvier nodes) with linear arrangement. and has the limitation that each computing unit can send spikes only to its left and right neighboring nodes, but not the other nodes.

In [30], it is reported that synchronous axon P systems cannot produce and accept the set of recursively enumerable languages. After that, an axon P system with two nodes was constructed, which can generate non-semilinear sets of natural numbers [31]. In 2015, an exciting result was obtained that axon P systems with 4 nodes can generate/compute the family of sets of Turing computable natural numbers, with 9 nodes can compute recursive functions, thus achieving Turing universality. In the systems, each computing node works in a non-synchronous mode and each information processing rule should be executed and completed in one time unit [32].

In the field of axon P systems, the computational power of using axons to do computation is not clear, particularly working in asynchronous mode. The results in the theoretical level is important for the application of the system. If we use axon P system to modelling neuro-biological systems, more computing models which are close to biological facts are needed. In this work, we consider the computational power of asynchronous axon P systems.

In neuro-biological fact, transitions of spikes among Ranvier nodes are not obligatory (even with enabled firing conditions) at any moment. Also, it is not proper to limit or request different nodes to complete their spike passing task in a uniform time unit. We investigate in this work asynchronous axon P systems. In the systems, the non-synchronization is obtained by either the obligatory use of enabled rules at any moment (non-synchronous mode from [43], [44]) or imposing a randomly execution time for each rule (the time-free mode from [45], [46]). Specifically, the computational power of asynchronous axon P systems as natural numbers computing devices is studied. As results, it is proved that axon P systems with 4 neurons in either non-synchronous or time-free working mode are Turing universal, which indicates that the non-synchronization will not reduce the power of axon P systems to compute/generate natural numbers. It is noted that in the universality proofs, the number of spikes for encoding arbitrary number n in asynchronous axon P systems is improved to $O(n)$ from $O(n^2)$ spikes used in Turing universal synchronous axon P systems [32].

The results indicate that axon P systems having 4 computing units and simple topological structure (linear structure) can achieve a "desired" computational power even working in non-synchronous modes and time-free modes. Axon P systems perform better than spiking neural P systems and certain class of artificial neural networks in terms of using a less number of computing or information processing units (such as neurons, processors or nodes) to achieve Turing universality. These results partially answer to the problem left and formulated in [32], and may provide some hints on designing novel learning strategies by imposing computation tasks on the synapses of neural networks models.

II. Axon P Systems

We firstly recall some basic concepts and notions in formal language and automata theory [18], which are helpful in understanding the definition of axon P systems. After that, asynchronous axon P systems are introduced by imposing non-synchronization on the application of rules (non-synchronous working mode) and execution time of the rules (time-free working mode).

Let V be an alphabet. We denote by V^* the set of strings produced by connection of a finite number of symbols from V. If a string has no symbol, then it is called empty and denoted by λ. The set of strings except for empty ones over V is represented by $V^+ = V^* - \lambda$. If $V = \{a\}$, V^* and V^+ can simplify written $\{a\}^*$ and $\{a\}^+$ as a^* and a^+, if respectively.

Given a finite alphabet V, the regular expressions associated with V is defined as follows.

- empty set $\emptyset$, i.e., the set containing no element is a regular expression;
- the set containing "empty" string only is a regular expression, that is, no characters contained at all;
- for any symbol a in V, the set composed of symbol a is a regular expression.

For any two regular expressions under the following operations, including concatenation, alternation and Kleene star, regular expressions can be produced. More details about regular expressions can be found in [47], [48].

The definition of regular language over an alphabet V is defined as follows, which can be referred to [49].

- The empty language $\emptyset$ is a regular language.
- For any symbol $v \in V$, the singleton language $\{v\}$ is a regular language.

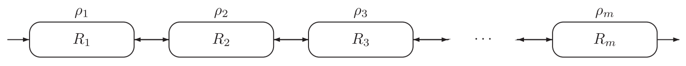

Fig. 2. The structure of an axon P system with linear m nodes.

- If A is a regular language , A^* (Kleene star) is a regular language.
- If A and B are regular languages, then $A \cdot B$ (concatenation) and $A \cup B$ are regular languages.
- No other languages over V are regular.

With each regular expression E, a regular language $L(E)$ is associated.

An axon P system (having $m \geq 1$ computing units) from [32] is defined as

$$\Pi = (O, \rho_1, \ldots, \rho_m), \text{ where:}$$

- O is an alphabet having unique symbol a, which is named as spike;
- $\rho_1, \ldots, \rho_m$ are m computing units (representing Ranvier nodes), each of which can be mathematically represented by $\rho_j = (n_j, R_j), j \in \{1, 2, \ldots, m\}$, where:
  a) n_j is the number of spikes initially placed in node ρ_j;
  b) R_j is a set of firing rules in node ρ_j, in which firing rules are of the form $E/a^c \to (a^l, a^r)$.
- ρ_1 (the leftmost node) is specified as input node to read spikes into the system from the environment, having only rules of the form $E/a^c \to (\lambda, a^r)$. Node ρ_m (the rightmost node) is the output node, which can emit spikes into the environment.

An axon P system holds a linear structure with nodes being aligned one after another along the axon. Any axon P system of degree m nodes (having m nodes) has the same topological structure, which is shown in Figure 2. The nodes aligned in linear order from ρ_1 to ρ_m. Node ρ_1, the input node, can read spikes from the environment, and node ρ_m can sends or emits spikes to its neighboring node ρ_{m-1} as well as to the environment, i.e., it is the output node.

The node transmits spikes to its neighboring nodes by using firing rule $E/a^c \to (a^l, a^r)$. The application of the firing rule in node ρ_i is as follows. At any moment t, if node ρ_i accumulates k spikes such that $a^k \in L(E)$, $k \geq c$, the firing rule is enabled. After using the firing rule, c spikes are consumed from node ρ_i (and $k - c$ spikes left), meanwhile, node ρ_i emits l spikes to its left neighboring node ρ_{i-1} and r spikes to its right neighboring node ρ_{i+1}, immediately. The spikes can be used for firing in the next step when they arrived at the target nodes. If any firing rule has $r = 0$ (resp. $l = 0$), then no spike is emitted to its right node ρ_{i+1} (resp. left node ρ_{i-1}). For the first node ρ_1, it holds rules of the form $E/a^c \to (\lambda, a^r)$. For the rightmost node ρ_m, besides sending spikes to its left node, node ρ_m will send spikes to the environment, which are counted and used to define the computation result of the system. If a firing rule has $E = a^c$, then it can simply be as $a^c \to (a^l, a^r)$.

At certain moment, it is possible to have two firing rules with common regular expressions, which means more than one firing rule in a node is applicable at that moment, but only one of the applicable firing rules can be randomly chosen and applied. In this way, the system works sequentially in the node, i.e., at most one rule can be used at any step, but for all nodes they work in a parallel manner.

The "state" (also known as configuration) of the system at a computation step is defined by the number of spikes accumulated in each node. With the notation, the initial configuration of axon P system is $\langle n_1, n_2, \ldots, n_m \rangle$. When the system halts, that is, the system reaches a "state" with no node having enabled rule(s), the number of spikes sent to the environment from the rightmost node in total is counted and taken as the computation result of the system. The nodes in the system can fire non-deterministically, thus the system can generate a set of natural numbers.

We consider in this work the computational power of asynchronous axon P systems, where the non-synchronization is induced by either the non-obligatory use of enabled rules at any moment (namely working in non-synchronous mode) or imposing a randomly execution time for time node spiking (namely working in the time-free mode).

- **Working in non-synchronous mode**
  In each time unit (marked by a global clock), if a node has an applicable firing rule, it is not obligatory to use the rule. In other words, each node can keep still/inactive in spite of having certain applicable firing rules against its spikes. For any node, it can decide to fire or not even with enabled firing rules and sufficient spikes. If the number of spikes in certain node is not changed, the applicable firing rule may be applied later. if the node receive some spikes to make the firing rule unable to use, then the computation proceeds in a new configuration. The set of natural numbers computed by asynchronous axon P system Π is denoted by $N_{all}^{asyn}(\Pi)$, where all means all the spikes emitted from the rightmost nodes into the environment are counted as the computation result. It is denoted by $N_{all}^{asyn} AxonP_m$ the family of sets of numbers generated by synchronous axon P systems having m modes.
- **Working in time-free mode**
  Before introducing the notation of time-free working mode in axon P systems, we shall revisit the notion of timed axon P systems. An axon P system is called timed, if there exits a time mapping $e : R \to \mathbb{N}$, by which each firing rule is associated with a execution time. The set of all the firing rules in the system is $R = R_1 \cup R_2 \cup \cdots \cup R_m$, and $\mathbb{N}$ is the set of natural numbers.

In timed axon P system, denoted by $\Pi(e)$, an external clock is assumed to mark time-units of equal length, starting from instant 0. The computation step t is the period of time that goes from instant $t - 1$ to t. For a firing rule r, its execution will take $e(r)$ time units to complete. In general, if the execution of rule r is started at instant j, it is completed at instant $j + e(r)$ and emitting spikes to its neighboring nodes at the beginning of moment $j + e(r) + 1$. When a rule r is started, then the node is in a "close" status and can not receive any spikes from its neighboring nodes. The spikes sending to the nodes in a "close" status will be lost, that is, be removed from the system. If there is no limitation on the time-mapping e, that is, by time-mapping

e, each firing rule can be associated with an arbitrary execution time, then the system is named time-free.

The set of all numbers computed/generated by a time-free axon P system $\Pi(e)$ is denoted by $N_{all}^{tf}(\Pi(e))$. The sets of numbers computed by time-free axon P systems of degree m is $N_{all}^{tf}AxonP_m(e)$.

III. Universality Results

In this section, it is obtained that asynchronous and time-free axon P systems having 4 nodes are Turing universal as number generators. The universality proofs are based on the computation of counter machine, which is equivalent with Turing machine as number computing device.

A counter machine M is composed by a set of counters and a set of instructions, in which the counters are used to store numbers and instructions are used to add or subtract number 1 from certain counter. The counter machine starts its computations from the initial instruction, and halts when the machine reaches the halt instruction. The computing results is the number stored in the output counter. Counter machine M can be formally denoted by $M = (m, H, l_0, l_h, I)$, where l_0 is the initial instruction, l_h is the halt label of instruction HALT; H is the set of instruction labels; m is the amount of counters; I is a set of instructions.

- ADD instruction: $l_i : (\mathtt{ADD}(r), l_j, l_k)$ (add 1 to counter r and non-deterministically go to instruction l_j or l_k);
- SUB instruction: $l_i : (\mathtt{SUB}(r), l_j, l_k)$ (if the number stored in counter r is greater than zero, then subtract 1 from it and perform instruction l_j, otherwise proceed to instruction l_k)
- HALT instruction: $l_h : \mathtt{HALT}$ (stop the computation of M).

At the beginning, a counter machine M has all counters empty (storing number zero) and performs the initial instruction l_0, and then it continues its computation by applying indicated instructions. When HALT instruction l_h is applied, the counter machine halts. Counter 1 is used as the output counter, and the number stored in it is said to be computed by M. The set of natural numbers generated by a counter machine M is written as $N(M)$.

From [50] counter machine M having 3 counters are Turing universality, which computes the family of length sets of recursively enumerable languages NRE. By RE, we denote the language generated and recognized by Turing machine, which is named recursively enumerable languages. Once a computing device can compute the length of recursively enumerable languages, it is said to be Turing universal, or can achieve Turing universality.) In the 3 counters (labeled by 1, 2 and 3), counter 1 is usually used to store the computation result. Without lose of generality, it is assumed that no SUB instruction acting on counter 1, but ADD instruction [50].

A. Working in Non-Synchronous Mode

The universality proof is achieved by constructing an asynchronous axon P systems with 4 nodes to simulate the computation of counter machine.

Theorem 3.1: $N_{all}^{asyn}AxonP_4 = NRE$.

Proof: It is sufficient to prove $NRE \subseteq N_{all}^{asyn}AxonP_4$, since the converse inclusion can be obtained directly with

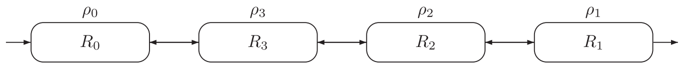

Fig. 3. The topological structure of axon P system Π.

Church-Turing thesis. An asynchronous axon P system Π with 4 nodes is constructed to simulate the computation of universal counter machine M with 3 counters, whose topological structure is shown in Figure 3. The nodes are labelled by ρ_0, ρ_1, ρ_2 and ρ_3, respectively.

In general, node ρ_0 works as a program node, whose function is similar to the state controller of automata, focusing on controlling the instruction to be simulated currently. Specifically, each instruction l_i of counter machine M is associated with a firing rule in node ρ_0. The firing rule is enabled (but not applied obligatorily due to the non-synchronous working mode) if and only if node ρ_0 contains $T + l(i)$ spikes, where $T = 2L$ with L being the number of instructions of M and $l(i) = i$, simulating counter machine M reaching instruction l_i. Node ρ_0 starts by reading $T + l(0) = 2L$ spikes from the environment, which indicates system Π has to simulate the first instruction l_0 of M.

For counters 1, 2 and 3 of counter machine M, nodes ρ_1, ρ_2 and ρ_3 are associated in system Π, and the number stored in counter r is encoded by the number of spikes in nodes ρ_1, ρ_2 and ρ_3. If counter 1 has number $n \geq 0$ at certain moment, then there are $7n$ spikes in nodes ρ_1; if counter 2 has number $n \geq 0$ at certain moment, then there are $11n$ spikes in nodes ρ_2; if counter 3 has number $n \geq 0$ at certain moment, then there are $13n$ spikes in nodes ρ_3.

During a computation, when node ρ_0 contains $T + i$ spikes at certain computation step, it fires to simulate instruction l_i : $(\mathtt{OP}(r), l_j, l_k)$ ($\mathtt{OP} \in \{\mathtt{ADD}, \mathtt{SUB}\}$) of M: starting by firing node ρ_0, adding or subtracting a number of spikes in node ρ_r, and introducing j or k spikes back to node ρ_0. In this way, node ρ_0 holds $T + j$ or $T + k$ spikes and can fire again to simulate instruction label l_j or l_k. When node ρ_1 (the output node) has $7n + T$ spikes, it means the computation of counter machine M is completely simulated. System Π needs to output the computation result.

The formal definition of the constructed axon system Π having 4 nodes is as follows.

$$\Pi = (O, \rho_0, \rho_1, \rho_2, \rho_3),$$

where the sets of rules R_0, R_1, R_2, R_3 in each node are listed in Table I.

Simulating the ADD instructions

It is obtained that 3 counters are sufficient for counter machine M to generate NRE, so the simulation of ADD instructions of M by system Π can be divided into three parts with respect to the acting counters.

(a) ADD instructions acting on counter 1

Assume at certain moment, system Π starts to simulate an ADD instruction l_i on counter 1. Node ρ_0 accumulates $T + l(i)$ spikes at that moment such that both firing rules $a^{T+l(i)}/a^{T+l(i)-l(j)} \to (\lambda, a^8)$ and $a^{T+l(i)}/a^{T+l(i)-l(k)} \to (\lambda, a^8)$ are can both be used. One of the two enabled firing rules

TABLE I
RULES IN EACH NODE OF SYSTEM Π, WHERE $T = 2L$ AND $l(i) = i$ WITH $0 \leq i \leq L$

Node	Associated rules
ρ_0	**Simulating the ADD instructions**
	$a^{T+l(i)}/a^{T+l(i)-l(j)} \to (\lambda, a^8)$ simulating ADD instructions on counter 1
	$a^{T+l(i)}/a^{T+l(i)-l(k)} \to (\lambda, a^8)$ simulating ADD instructions operated on counter 1
	$a^{T+l(i)}/a^{T+l(i)-l(j)} \to (\lambda, a^{11})$ simulating ADD instructions on counter 2
	$a^{T+l(i)}/a^{T+l(i)-l(k)} \to (\lambda, a^{11})$ simulating ADD instructions on counter 2
	$a^{T+l(i)}/a^{T+l(i)-l(j)} \to (\lambda, a^{14})$ simulating ADD instructions on counter 3
	$a^{T+l(i)}/a^{T+l(i)-l(k)} \to (\lambda, a^{14})$ simulating ADD instructions on counter 3
	Simulating SUB instructions subjecting to counter 2
	$a^{T+l(i)} \to (\lambda, a^3)$
	Simulating SUB instructions subjecting to counter 3
	$a^{T+l(i)} \to (\lambda, a^6)$
	Simulating the halting instruction
	$a^{T+l(h)} \to (\lambda, a^T)$
ρ_3	**Simulating ADD instructions subjecting to counter 1**
	$(a^{13})^*a^8/a^7 \to (\lambda, a^7)$ for ADD instructions on counter 1
	$(a^{13})^*a^2/a^2 \to (a^T, \lambda)$ for ADD instructions on counter 1
	Simulating ADD instructions subjecting to counter 2
	$(a^{13})^*a^{11}/a^{11} \to (a^T, a^{11})$ for ADD instructions on counter 2
	Simulating ADD instructions subjecting to counter 3
	$(a^{13})^*a^{14}/a \to (a^T, \lambda)$ for ADD instructions on counter 3
	Simulating SUB instructions subjecting to counter 2
	$(a^{13})^*a^3/a^3 \to (\lambda, a^3)$ for SUB instructions on counter 2
	$(a^{13})^*a^4/a^4 \to (a^{T+l(j)}, \lambda)$ for SUB instructions on counter 2 with $n > 0$
	$(a^{13})^*a^5/a^5 \to (a^{T+l(k)}, \lambda)$ for SUB instructions on counter 2 with $n = 0$
	Simulating SUB instructions subjecting to counter 3
	$(a^{13})^+a^6/a^{19} \to (a^{T+l(j)}, \lambda)$ for SUB instructions l_i acting on counter 3 with $n > 0$
	$(a^6/a^6 \to (a^{T+l(k)}, \lambda)$ for SUB instructions l_i on counter 3 with $n = 0$
	Simulating the halting instruction
	$(a^{13})^*a^T/a^T \to (\lambda, a^T)$
ρ_2	**Simulating ADD instructions acting on counter 1**
	$(a^{11})^*a^7/a^7 \to (a, a^7)$ for ADD instructions on counter 1
	Simulating SUB instructions acting on counter 2
	$(a^{11})^+a^3/a^{14} \to (a^4, \lambda)$ for SUB instructions on counter 2 with $n > 0$
	$a^3/a^3 \to (a^5, \lambda)$ for SUB instructions on counter 2 with $n = 0$
	Simulating the halting instruction
	$(a^{11})^*a^T/a^T \to (\lambda, a^T)$
ρ_1	**Outputting the computation result**
	$(a^7)^+a^T/a^7 \to (\lambda, a)$

is non-deterministically related to be applied. Since the system works in non-synchronous mode, the selected rule is not applied obligatorily at that moment. But the rule will be applied later, because no further spike can arrive at node ρ_0.

– If rule $a^{T+l(i)}/a^{T+l(i)-l(j)} \to (\lambda, a^8)$ is applied, node ρ_0 ends up with $T + l(i) - (T + l(i) - l(j)) = l(j)$ and sends 8 spikes to node ρ_3. The number of spikes in node ρ_3 becomes $13n + 8$, and firing rule $(a^{13})^*a^8/a^7 \to (\lambda, a^7)$ will be applied, consuming 7 spikes and sending 7 spikes to node ρ_2. The number of spikes in node ρ_2 becomes to $11n + 7, n \geq 0$. Firing rule $(a^{11})^*a^7/a^7 \to (a, a^7)$ will be applied in certain moment. Eventually, node ρ_2 sends 7 spikes to its right node ρ_1, and the number of spikes in node ρ_1 is increased by 7, simulating adding 1 to counter 1. Meanwhile, node ρ_2 sends one spike back to its left node ρ_3, which holds $13n + 2, n \geq 0$ spikes and will become active immediately or some steps later by using firing rule $(a^{13})^*a^2/a^2 \to (a^T, \lambda)$. When the rule is applied, node ρ_3 sends T spikes to node ρ_0 such that node ρ_0 has $T + l(j)$ spikes, which indicates system Π starts to simulate instruction l_j of M.

– If firing rule $a^{T+l(i)}/a^{T+l(i)-l(k)} \to (\lambda, a^8)$ is applied at some moment, node ρ_0 ends up with $l(k)$ and sends 8 spikes to node ρ_3. Having $13n + 8, n \geq 0$ spikes inside, node ρ_3 becomes active at certain moment by using firing rule $(a^{13})^*a^8/a^7 \to (\lambda, a^7)$, consuming 7 spikes (one spike left) and sending 7 spikes to node ρ_2. Node ρ_2 holds $11n + 7, n \geq 0$ spikes such that firing rule $(a^{11})^*a^7/a^7 \to$

(a, a^7) will be applied at that moment or some steps later. When the rule is applied, node ρ_2 sends 7 spikes to node ρ_1. The number of spikes in node ρ_1 is incremented by 7, simulating the operation of adding one to counter 1. Meanwhile, node ρ_2 sends one spike back to its left node ρ_3. With $13n + 2, n \geq 0$ spikes, node ρ_3 can become active immediately or some steps later by using rule $(a^{13})^* a^2/a^2 \rightarrow (a^T, \lambda)$. When the rule is applied, T spikes are sent to node ρ_0 such that node ρ_0 has $T + l(k)$ spikes. Asynchronous axon P system Π starts to simulate instruction l_k of M.

(b) ADD instruction acting on counter 2

Let $l_i : (\mathtt{ADD}(2),\ l_j, l_k)$ be an ADD instruction acting on counter 2. When system Π starts to simulate it, node ρ_0 contain $T + l(i)$ spikes at that moment. Having $T + l(i)$ spikes, both of firing rules $a^{T+l(i)}/a^{T+l(i)-l(j)} \rightarrow (\lambda, a^{11})$ and $a^{T+l(i)}/a^{T+l(i)-l(k)} \rightarrow (\lambda, a^{11})$ are enabled, but only one of them can be used at certain moment.

- If firing rule $a^{T+l(i)}/a^{T+l(i)-l(j)} \rightarrow (\lambda, a^{11})$ is chosen to apply, node ρ_0 ends with $l(j)$ spikes and sends 11 spikes to node ρ_3. By receiving 11 spikes, node ρ_3 contains $13n + 11, n \geq 0$ spikes and will fire at certain moment by using rule $(a^{13})^* a^{11}/a^{11} \rightarrow (a^T, a^{11})$, sending 11 spikes to node ρ_2. The number of spikes in node ρ_2 is increased by 11, simulating the number stored in counter 2 is added by 1. At the same moment, node ρ_3 ends T spikes to node ρ_0. Asynchronous anon P system Π starts to simulate instruction l_j of M.
- Similarly, if $a^{T+l(i)}/a^{T+l(i)-l(j)} \rightarrow (\lambda, a^{11})$ is chosen to apply, the number of spikes in node ρ_2 will be increased by 11, simulating adding 1 on counter 2, and the number of spikes in node ρ_0 becomes $T + l(k)$. Asynchronous axon P system Π starts to simulate instruction l_k of M.

(c) ADD instruction acting on counter 3

When system Π simulates an ADD instruction l_i acting on counter 3, node ρ_0 holds $T + l(i)$ spikes such that node ρ_0 can non-deterministically choose one of the two firing rules, $a^{T+l(i)}/a^{T+l(i)-l(j)} \rightarrow (\lambda, a^{14})$ and $a^{T+l(i)}/a^{T+l(i)-l(k)} \rightarrow (\lambda, a^{14})$ and fire, ending with $l(j)$ or $l(k)$ spikes and emitting 14 spikes to node ρ_3. With $13n + 14, n \geq 0$ spikes, firing rule $(a^{13})^* a^{14}/a \rightarrow (a^T, \lambda)$ can be applied, consuming one spike and sending T spikes to node ρ_0. The number of spikes in node ρ_3 is increased by 13, simulating the operation of adding 1 on counter 3, and node ρ_0 holds $T + l(j)$ or $t + l(k)$ spikes. At that moment, asynchronous axon P system Π proceeds to simulate instruction l_j or l_k, non-deterministically.

Simulating SUB instructions

Without lose of generality, it is assumed that there is no SUB instruction operated on counter 1, i.e., the output counter. So, there are only SUB instructions acting on counter 2 and 3.

(a) Simulating SUB instruction on counter 2

When system Π starts the simulation of SUB instruction l_i operated on counter 2, node ρ_0 has $T + l(i)$ spikes and firing rule $a^{T+l(i)} \rightarrow (\lambda, a^3)$ can be applied at certain moment. By using the rule, node ρ_0 transmits 3 spikes to node ρ_3, and a certain step later node ρ_3 transmits the 3 spikes to node ρ_2. In node ρ_2, it has the following two cases.

- If node ρ_2 accumulates $11n, n > 0$ spikes, then firing rule $(a^{11})^+ a^3/a^{14} \rightarrow (a^4, \lambda)$ is enabled and applied at certain moment. By using the rule, 14 spikes are canceled by node ρ_2 and 4 spikes are passed to node ρ_3. By removing 14 spikes, node ρ_2 ends with $11n + 3 - 14 = 11(n-1), n \geq 1$ spikes. In this way, system Π simulates that the number in counter 2 has been decremented by one. Node ρ_3 fires at certain moment by firing rule $(a^{13})^* a^4/a^4 \rightarrow (a^{T+l(j)}, \lambda)$, transmitting $T + l(j)$ spikes to node ρ_0. This indicates instruction l_j of M is to be simulated by system Π.
- If node ρ_2 has no spike (corresponding to the fact that the number stored in counter 2 is 0), then firing rule $(a^3/a^3 \rightarrow (a^5, \lambda)$ is applied at certain moment. By using the rule, 5 spikes are sent to node ρ_3 such that firing rule $(a^{13})^* a^5/a^4 \rightarrow (a^{T+l(k)}, \lambda)$ will be used in some step. In this case, node ρ_3 sends $T + l(k)$ spikes to node ρ_0. Asynchronous axon P system Π goes to simulate instruction l_k of M.

(b) Simulating SUB instruction operated counter 3

Assume at certain step, system Π starts the simulation of SUB instruction l_i acting on counter 3. At that moment, node ρ_0 has $T + l(i)$ spikes and firing rule $a^{T+l(i)} \rightarrow (\lambda, a^6)$ can be applied at certain moment to send 6 spikes to node ρ_3. In node ρ_3, it has the following two cases.

- If node ρ_3 accumulates $13n, n > 0$ spikes (corresponding to the fact that the number stored in counter 3 is larger than 0), then firing rule $(a^{13})^+ a^6/a^{19} \rightarrow (a^{T+l(j)}, \lambda)$ is applied at certain moment. With using the neuron, node ρ_3 consumes 19 spikes and transmit $T + l(j)$ spikes to node ρ_0. In node ρ_3, there are in total $13n + 6 - 19 = 13(n-1)\ (n \geq 1)$ spikes. This simulates the operation that the number in counter 2 is decremented by one. Node ρ_0 has $T + l(j)$ spikes, which means system Π starts to simulate instruction l_j of M.
- If node ρ_3 has no spike (corresponding to the fact that the number stored in counter 3 is 0), then firing rule $(a^6/a^6 \rightarrow (a^{T+l(k)}, \lambda)$ is applied at certain moment, sending node $T + l(k)$ spikes to node ρ_0. This means system Π starts to simulate instruction l_k of M.

Outputting the computation result

It is assumed that at ceratin computation step counter machine M reaches the halting instruction. In system Π, node ρ_0 accumulates $T + l(h)$ spikes. At that moment, node ρ_1 contains $7n$ spikes, i.e., counter 1 hold number n when M halts.

With $T + l(h)$ spikes in node ρ_0, firing rule $a^{T+l(h)} \rightarrow (\lambda, a^T)$ in node ρ_0 is applied, emitting T spikes to node ρ_3. Node ρ_3 sends T spikes to node ρ_2 by using firing rule $(a^{13})^* a^T/a^T \rightarrow (\lambda, a^T)$, and the the T spikes will be transmitted to node ρ_1 when node ρ_2 fires. Node ρ_1 accumulates $7n + T$ spikes and fires by using firing rule $(a^7)^+ a^T/a^7 \rightarrow (\lambda, a)$. Such firing rule will be applied in total for n times by node ρ_1. And in each time firing, node ρ_1 sends one spike into the environment. When the computation of system Π halts, n spikes are emitted into the environment, which is exactly the number stored in counter 1 of M when it halts.

TABLE II
RULES IN EACH NODE OF SYSTEM Π', WHERE $T = 2L$ AND $l(i) = i$ WITH $0 \leq i \leq L$

Node	The set of rules	Execution time
ρ_0'	**Simulating ADD instructions operated on counter 1**	
	r_{01}: $a^{T+l(i)}/a^{T+l(i)-l(j)} \to (\lambda, a^8)$	$e(r_{01})$ steps
	r_{02}: $a^{T+l(i)}/a^{T+l(i)-l(k)} \to (\lambda, a^8)$	$e(r_{02})$ steps
	Simulating ADD instructions operated on counter 2	
	r_{03}: $a^{T+l(i)}/a^{T+l(i)-l(j)} \to (\lambda, a^{11})$	$e(r_{03})$ steps
	r_{04}: $a^{T+l(i)}/a^{T+l(i)-l(k)} \to (\lambda, a^{11})$	$e(r_{04})$ steps
	Simulating ADD instructions operated on counter 3	
	r_{05}: $a^{T+l(i)}/a^{T+l(i)-l(j)} \to (\lambda, a^{14})$	$e(r_{05})$ steps
	r_{06}: $a^{T+l(i)}/a^{T+l(i)-l(k)} \to (\lambda, a^{14})$	$e(r_{06})$ steps
	Simulating SUB instructions operated on counter 2	
	r_{07}: $a^{T+l(i)} \to (\lambda, a^3)$	$e(r_{07})$ steps
	Simulating SUB instructions operated on counter 3	
	r_{08}: $a^{T+l(i)} \to (\lambda, a^6)$	$e(r_{08})$ steps
	Simulating HALT instruction l_h	
	r_{09}: $a^{T+l(h)} \to (\lambda, a^T)$	$e(r_{09})$ steps
ρ_3'	**Simulating ADD instructions operated on counter 1**	
	r_{30}: $(a^{13})^* a^8/a^7 \to (\lambda, a^7)$	$e(r_{30})$ steps
	r_{31}: $(a^{13})^* a^2/a^2 \to (a^T, \lambda)$	$e(r_{31})$ steps
	Simulating ADD instructions operated on counter 2	
	r_{32}: $(a^{13})^* a^{11}/a^{11} \to (a^T, a^{11})$	$e(r_{32})$ steps
	Simulating ADD instructions operated on counter 3	
	r_{33}: $(a^{13})^* a^{14}/a \to (a^T, \lambda)$	$e(r_{33})$ steps
	Simulating SUB instructions operated on counter 2	
	r_{34}: $(a^{13})^* a^3/a^3 \to (\lambda, a^3)$	$e(r_{34})$ steps
	r_{35}: $(a^{13})^* a^4/a^4 \to (a^{T+l(j)}, \lambda)$	$e(r_{35})$ steps
	r_{36}: $(a^{13})^* a^5/a^5 \to (a^{T+l(k)}, \lambda)$	$e(r_{36})$ steps
	Simulating SUB instructions operated on counter 3	
	r_{37}: $(a^{13})^+ a^6/a^{19} \to (a^{T+l(j)}, \lambda)$	$e(r_{37})$ steps
	r_{38}: $(a^6/a^6 \to (a^{T+l(k)}, \lambda)$	$e(r_{38})$ steps
	Simulating HALT instruction l_h	
	r_{39}: $(a^{13})^* a^T/a^T \to (\lambda, a^T)$	$e(r_{39})$ steps
ρ_2'	**Simulating ADD instructions operated on counter 1**	
	r_{20}: $(a^{11})^* a^7/a^7 \to (a, a^7)$	$e(r_{20})$ steps
	Simulating SUB instructions operated on counter 2	
	r_{21}: $(a^{11})^+ a^3/a^{14} \to (a^4, \lambda)$	$e(r_{21})$ steps
	r_{22}: $a^3/a^3 \to (a^5, \lambda)$	$e(r_{22})$ steps
	Simulating HALT instruction l_h	
	r_{23}: $(a^{11})^* a^T/a^T \to (\lambda, a^T)$	$e(r_{23})$ steps
ρ_1'	**Outputting the computation result**	
	r_{10}: $(a^7)^+ a^T/a^7 \to (\lambda, a)$	$e(r_{10})$ steps

From the above description, the computation of universal counter machine M (with 3 counters) can be correctly simulated by asynchronous axon P system Π with 4 nodes. It has $N_{all}^{asyn} AxonP_4 = N(M)$.

Therefore, it holds $NRE = N_{all}^{asyn} AxonP_4$, and this completes the proof. ■

B. Working in Time-Free Mode

In this subsection, the universality of axon P systems working in time-free mode is obtained by simulating the computation of universal counter machine.

Theorem 3.2. $N_{all}^{tf} AxonP_4(e) = NRE$.

Proof: It is sufficient to prove $NRE \subseteq N_{all}^{tf} AxonP_4(e)$. A time-free axon P system Π' with 4 nodes is constructed to simulate universal counter machine M with 3 counters. The topological structure of system Π' is the same with the one shown in Figure 3.

The constructed time-free axon P system Π' is formally defined as

$$\Pi' = (O, \rho_0', \rho_1', \rho_2', \rho_3', e).$$

The sets of firing rules R_0', R_1', R_2', R_3' are the same with the ones in system Π. A random time-mapping $e : R' \to \mathbb{N}$ is introduced, by which each rule is associated with a random execution time, instead of one time unit as working in non-synchronous mode. The rules and its execution time are shown in Table II. For counters 1, 2 and 3 of M, nodes ρ'_1, ρ'_2 and ρ'_3 are associated in system Π', and the number stored in counter r is encoded by the number of spikes in the nodes. If counter 1 stores number $n \geq 0$, there are $7n$ spikes in nodes ρ'_1; if counter 2 stores number $n \geq 0$, there are $11n$ spikes in nodes ρ'_2; if counter 3 stores number $n \geq 0$, there are $13n$ spikes in nodes ρ'_3.

In time-free axon P system Π', node ρ'_0 works to control which instruction of M to be simulated. In general, when system Π' simulates instruction l_i of M at any step, there are $T + i$ spikes in node ρ'_0. Since the initial instruction of M is l_0, node ρ'_0 starts by reading $T + l(0) = 2L$ spikes from the environment. This indicates system Π' starts to simulate the initial instruction l_0 of M. During the simulation, when node ρ'_0 has $T + i$ spikes, it fires to simulate that register machine M reaches instruction l_i. When node ρ'_1 (the output node) has $7n + T$ spikes, the computation in M is completely simulated by system Π' and it begins to output the computation result.

Different from the case of working in non-synchronous mode, node in time-free axon system Π' becomes active if it has enabled rule(s), but the execution time is determined by the time mapping e, can be random number. It is not hard to find that the universal counter machine M with 3 counters can be correctly simulated by time-free axon P system Π' having 4 nodes. The details of the proof are quite similar with Theorem 3.1, which are omitted here. ■

IV. Discussion and Conclusion

In this paper, we has obtained the computational power of axon P systems working in non-synchronous mode, where the non-synchronization can be obtained by either the obligatory use of enabled rules at any moment (namely the non-synchronous mode) or imposing a random execution time for each rule (namely time-free mode). It is proved that asynchronous and time-free axon P systems having 4 nodes with the same topological structure can compute/genrate the family of sets of length of recursively enumerable languages, i.e., the set of Turing computable natural numbers, thus achieving Turing universality. This indicates that the non-synchronization will not reduce the computational power of axon P systems. As well, it is found that the number of nodes used in non-synchronous axon P systems is not more than the ones needed for synchronous axon P systems.

In [30], the encoding of value n is done by $3Ln$ spikes, where L is amount of instructions in the register machine. The encoding mechanism is still linear ($O(n)$), not quadratic. In our systems, we use $13n$ spikes to encode number n, which also improves the information encoding way from [30].

Moreover, in the universality proofs, the number of spikes used to encode arbitrary number n in nodes associated with counters of universal counter machine in asynchronous and time-free axon P systems are $7n, 11n, 13n$, i.e., it is $O(n)$. This improved the strategy in universal synchronous axon P systems, wherein $O(n^2)$ spikes are used to represent number n in the nodes associated with the counters [32]. Comparing with SN P systems (4 neurons with request rules [39] or white hole rules [40]) and recurrent neural networks (886 units needed [51]) to achieve Turing universality, axon P systems doing computation along axon are with a smaller number of computing units to achieve Turing universality. Moreover, this is the first attempt to do computation along with axon working in non-synchronous mode in axon P systems.

These results give a partial answer to an open problem left in [32], and may provide some hints on designing novel learning strategies or training method for neural-like computing models by imposing some computation tasks on the synapses of neural networks models.

In 2015, memcomputing machines were proposed in [52]. The machine has linear structure with units in memory, and can run multiple instructions in a parallel manner. It is of interests to design memcomputing machines with firing rules and linear structure as axon P systems working in synchronous or non-synchronous mode, which will increase the level of realism a neural simulation, thus resulting power brain-inspired computing machines or systems.

In the theoretical level, axon P systems working in both synchronous and asynchronous modes perform better than artificial neural networks [51] in terms of using less number of computing units to achieve Turing universality. Few results focus on solving real-life problems, like natural language processing or optimization problems by axon P systems It is quite an interesting research topic for further research. One of the most vital problems is that the data structure of axon P system are discrete spikes, and it needs to find a way to represent natural languages and other problems by spikes, and then axon P systems can be used to deal with the information represented by the spikes or spike trains.

In 2019, spiking neural P systems with learning functions are proposed in [53], in which synapses and their weights can be updated during the computation. The idea is directly from the neural systems in human brain for learning. This can also be useful in developing learning axon P systems, such as spiking rules can be updated during the computation of axon P system.

References

[1] P. Swietojanski, A. Ghoshal, and S. Renals, "Convolutional neural networks for distant speech recognition," *IEEE Signal Process. Lett.*, vol. 21, no. 9, pp. 1120–1124, Sep. 2014.

[2] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús, *Neural Network Design*, vol. 20. Boston, MA, USA: PWS-Kent, 1996.

[3] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, 1997.

[4] G. A. Carpenter, "Neural network models for pattern recognition and associative memory," *Neural Netw.*, vol. 2, no. 4, pp. 243–257, 1989.

[5] G. Deco and B. Schürmann, "The coding of information by spiking neurons: An analytical study," *Netw., Comput. Neural Syst.*, vol. 9, no. 3, pp. 303–317, 1998.

[6] X. Zhang, K. Zhou, H. Pan, L. Zhang, X. Zeng, and Y. Jin, "A network reduction based multi-objective evolutionary algorithm for community detection in large-scale complex networks," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2018.2871673.

[7] N. K. Kasabov, "Neucube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data," *Neural Netw.*, vol. 52, pp. 62–76, 2014.

[8] Y. Tian, R. Cheng, X. Zhang, F. Cheng, and Y. Jin, "An indicator based multi-objective evolutionary algorithm with reference point adaptation for better versatility," *IEEE Trans. Evol. Comput.*, vol. 22, no. 4, pp. 609–622, Aug. 2018.

[9] Y. Tian, R. Cheng, X. Zhang, Y. Su, and Y. Jin, "A strengthened dominance relation considering convergence and diversity for evolutionary many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 2, pp. 331–345, Apr. 2019.
[10] D. V. Buonomano and M. Merzenich, "A neural network model of temporal code generation and position-invariant pattern recognition," *Neural Comput.*, vol. 11, no. 1, pp. 103–116, 1999.
[11] F. Ponulak and A. Kasinski, "Supervised learning in spiking neural networks with resume: Sequence learning, classification, and spike shifting," *Neural Comput.*, vol. 22, no. 2, pp. 467–510, 2010.
[12] T. Song, S. Pang, S. Hao, A-R. Paton, and P. Zheng, "A parallel image skeletonizing method using spiking neural P systems with weights," *Neural Process. Lett.*, 2019, doi: 10.1007/s11063-018-9947-9.
[13] X. Wang, P. Zheng, T. Ma, and T. Song, "Computing with bacteria conjugation: Small universal systems," *Moleculer*, vol. 23, no. 6, pp. 1307–1313, 2018.
[14] S. M. Bohte, J. N. Kok, and H. La Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, no. 1, pp. 17–37, 2002.
[15] C. W. Eurich and S. D. Wilke, "Multidimensional encoding strategy of spiking neurons," *Neural Comput.*, vol. 12, no. 7, pp. 1519–1529, 2000.
[16] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "A decision variable clustering based evolutionary algorithm for large-scale many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 97–112, Feb. 2018.
[17] G. Păun, "Computing with membranes," *J. Comput. Syst. Sci.*, vol. 61, no. 1, pp. 108–143, 2000.
[18] G. Păun, G. Rozenberg, and A. Salomaa, *The Oxford Handbook of Membrane Computing*. London, U.K.: Oxford Univ. Press, 2010.
[19] T. Song, X. Zeng, Z. Pan, J. Min, and R.-P. Alfonso, "A parallel workflow pattern modeling using spiking neural P systems with colored spikes," *IEEE Trans. Nanobiosci.*, vol. 17, no. 4, pp. 474–484, Oct. 2018.
[20] M. Ionescu, G. Păun, and T. Yokomori, "Spiking neural P systems," *Fundamenta Informaticae*, vol. 71, no. 2, pp. 279–308, 2006.
[21] A. Păun and G. Păun, "Small universal spiking neural P systems," *BioSyst.*, vol. 90, no. 1, pp. 48–60, 2007.
[22] X. Zeng, T. Song, X. Zhang, and L. Pan, "Performing four basic arithmetic operations with spiking neural P systems," *IEEE Trans. NanoBiosci.*, vol. 11, no. 4, pp. 366–374, Dec. 2012.
[23] S. Pang, T. Ding, A. Rodriguez-Paton, T. Song, and P. Zheng, "A parallel bioinspired framework for numerical calculations using enzymatic P system with an enzymatic environment," *IEEE Access*, vol. 6, pp. 65548–65556, 2018, doi: 10.1109/ACCESS.2018.2876364.
[24] T. Song, P. Zheng, D. M. Wong, and X. Wang, "Design of logic gates using spiking neural P systems with homogeneous neurons and astrocytes-like control," *Inf. Sci.*, vol. 372, 2016, Art. no. 380C391.
[25] X. Zeng, X. Zhang, and L. Pan, "Homogeneous spiking neural P systems," *Fundamenta Informaticae*, vol. 97, no. 1, pp. 275–294, 2009.
[26] A. Leporati, G. Mauri, C. Zandron, G. Păun, and M. J. Pérez-Jiménez, "Uniform solutions to SAT and subset sum by spiking neural P systems," *Natural Comput.*, vol. 8, no. 4, pp. 681–702, 2009.
[27] T. Song, L. Pan, K. Jiang, B. Song, and W. Chen, "Normal forms for some classes of sequential spiking neural P systems," *IEEE Trans. NanoBiosci.*, vol. 12, no. 3, pp. 255–264, Sep. 2013.
[28] T. Song, J. Xu, and L. Pan, "On the universality and non-universality of spiking neural P systems with rules on synapses," *IEEE Trans. Nanobiosci.*, vol. 14, no. 8, pp. 960–966, Dec. 2015.
[29] X. Wang, T. Song, F. Gong, and P. Zheng, "On the computational power of spiking neural P systems with self-organization," *Sci. Rep.*, vol. 6, 2016, Art. no. 27624.
[30] C. Haiming, T.-O. Ishdorj, and G. Paun, "Computing along the axon," *Prog. Natural Sci.*, vol. 17, no. 4, pp. 417–423, 2007.
[31] X. Zhang, J. Wang, and L. Pan, "A note on the generative power of axon P systems," *Int. J. Comput. Commun. Control*, vol. 4, no. 1, pp. 92–98, 2009.
[32] X. Zhang, L. Pan, and A. Păun, "On the universality of axon P systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 11, pp. 2816–2829, Nov. 2015.
[33] 2018. [Online]. Available: https://www.studyblue.com/notes/michigan-schools/us/mi
[34] A. Krogh *et al.*, "Neural network ensembles, cross validation, and active learning," *Adv. Neural Inf. Process. Syst.*, vol. 7, pp. 231–238, 1995.
[35] S. Haykin, "A comprehensive foundation," *Neural Network*, Englewood Cliffs, N.J.: Prentice-Hall, 2004.
[36] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. London, U.K.: Psychology Press, 2005.
[37] T. Song, X. Wang, Z. Zhang, and Z. Chen, "Homogenous spiking neural P systems with anti-spikes, neural computing and applications," *Neural Comput. Appl.*, vol. 24, no. 7–8, pp. 1833–1841, 2014.
[38] B. Wang and K. Liu, "Designing DNA code: Quantity and quality," *Int. J. Adaptive Innov. Syst.*, to be published.
[39] T. Song and L. Pan, "Spiking neural P systems with request rules," *Neurocomputing*, vol. 193, pp. 193–200, 2016.
[40] T. Song, F. Gong, X. Liu, Y. Zhao, and X. Zhang, "Spiking neural P systems with white hole rules," *IEEE Trans. Nanobiosci.*, vol. 15, no. 7, pp. 666–673, Oct. 2016.
[41] B. D. Clark, E. M. Goldberg, and B. Rudy, "Electrogenic tuning of the axon initial segment," *The Neurosci.*, vol. 15, no. 6, pp. 651–668, 2009.
[42] D. Khodagholy *et al.*, "Neurogrid: Recording action potentials from the surface of the brain," *Nature Neurosci.*, vol. 18, no. 2, pp. 310–315, 2015.
[43] M. Cavaliere, O. H. Ibarra, G. Păun, O. Egecioglu, M. Ionescu, and S. Woodworth, "Asynchronous spiking neural P systems," *Theor. Comput. Sci.*, vol. 410, no. 24, pp. 2352–2364, 2009.
[44] T. Song, L. Pan, and G. Păun, "Asynchronous spiking neural P systems with local synchronization," *Inf. Sci.*, vol. 219, pp. 197–207, 2012.
[45] L. Pan, X. Zeng, and X. Zhang, "Time-free spiking neural P systems," *Neural Comput.*, vol. 23, no. 5, pp. 1320–1342, 2011.
[46] T. Song, L. F. Macías-Ramos, L. Pan, and M. J. Pérez-Jiménez, "Time-free solution to sat problem using P systems with active membranes," *Theor. Comput. Sci.*, vol. 529, pp. 61–68, 2014.
[47] M. Sipser, *Introduction to the Theory of Computation*. Boston, MA, USA: Cengage Learning, 2012.
[48] J. E. Hopcroft, *Introduction to Automata Theory, Languages, and Computation*. London, U.K.: Pearson, 1979.
[49] R. Mitkov, *The Oxford Handbook of Computational Linguistics*. London, U.K.: Oxford Univ. Press, 2005.
[50] M. L. Minsky, *Computation: Finite and Infinite Machines*. Upper Saddle River, NJ, USA: Prentice-Hall, 1967.
[51] H. T. Siegelmann and E. D. Sontag, "On the computational power of neural nets," *J. Comput. Syst. Sci.*, vol. 50, no. 1, pp. 132–150, 1995.
[52] F. L. Traversa and M. Di Ventra, "Universal memcomputing machines," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 11, pp. 2702–2715, 2015.
[53] S. Tao, P. Linqiang, Z. Pan, W. D. Mouling, and R.-P. Alfonso, "Spiking neural P systems with learning functions," *IEEE Trans. Nanobiosci.*, to be published, doi: 10.1109/TNB.2019.2896981.