Inf. Comput. 311 (2026) 105472

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/yinco

Spiking neural P systems incorporating winner-take-all mechanism

Tingting Bao[a,b], Bifan Wei[c,d,*], Bo Li[a,b], Lingling Zhang[a,b,*], Hong Peng[e], Xiaoqing Zhang[f], Jun Liu[a,d]

[a] School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, Shaanxi, 710049, China
[b] Ministry of Education Key Laboratory of Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, 710049, China
[c] School of Continuing Education, Xi'an Jiaotong University, Xi'an, 710049, China
[d] Shaanxi Province Key Laboratory of Big Data Knowledge Engineering, Xi'an Jiaotong University, Xi'an, 710049, China
[e] School of Computer and Software Engineering, Xihua University,Chengdu, Sichuan, 610039, China
[f] School of Medicine, Tongji University, Shanghai, 200092, China

ARTICLE INFO

Keywords:
Neural computation
Spiking neural p systems
Computational efficiency
Winner-take-all mechanism
Turing universality

ABSTRACT

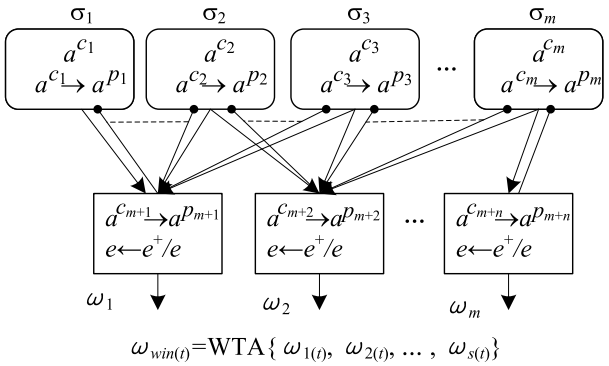
Spiking neural P systems (SNP systems), a class of parallel distributed computational models inspired by biological neurons, have become an important research direction in biocomputing in recent years due to their biological interpretability and low-power computing. There are many studies on the expressive power of SNP system variants, but their computational efficiency is not high in terms of resource overhead. Inspired by the biological winner-take-all (WTA) computation mechanism, this study proposes spiking neural P systems incorporating WTA mechanism (WTASNP systems). Competing neuron nodes are introduced into the WTASNP systems to realize the competitive selection and inhibition control mechanisms of the WTA computation. After competition, only the winner neuron is allowed to emit spikes, while the loser neurons are inhibited. It is proved that the WTASNP systems have Turing universality. Furthermore, it reduces the computational resource requirements for solving the NP-complete SAT problem with SNP systems from $O(n^2)$ or $O(2^n)$ complexity levels down to linearly solvable $O(n)$. The WTASNP systems not only effectively preserve the preamble spike information, but also inhibits the loser neuron spike issuance by the WTA computational mechanism of competing neurons, reduces redundant computation to avoid neuron over-excitation, and improves the computational efficiency and expressive power.

1. Introduction

Inspired by biological neurons, a variety of models have emerged in the field of biocomputing that attempt to simulate neural information processing processes. Among them, membrane computing system is a parallel computing model inspired by the structure and function of biological cells [1,2]. SNP system is an important branch of membrane computing inspired by biological neural system [3,4]. SNP systems are parallel computational models in which information is conveyed by spikes, and have become a hot research topic in recent years due to their ability to mimic biological neural mechanisms and their scalability in dealing with real-world problems [5,6]. SNP systems use neuron as the basic computing unit, and spikes and rules can be stored in the neuron. The

* Corresponding author.
E-mail addresses: weibifan@xjtu.edu.cn (B. Wei), zhanglling@xjtu.edu.cn (L. Zhang).

https://doi.org/10.1016/j.ic.2026.105472
Received 12 June 2025; Received in revised form 21 April 2026; Accepted 11 May 2026
Available online 14 May 2026



(a) A WTASNP system

Models	Neurons
SNP	$3n^2+8m+5$
DBSNP	$3*(2^n+n+1)$
ECSNP-ER	$n^2+2n+4m+1$
MSNP	2^n+3n+2
ACSNP	$2^{n+1}+3n+1$
WTASNP (Proposed)	$\mathbf{3m+2}$

n variables, m clauses.

(b) Computational resources

Fig. 1. Compared to different SNP system variants. (a) A WTASNP system with a system organization architecture abstracted from the winner-take-all (WTA) mechanism. (b) Computational resources, the number of neurons for solving the $SAT(n, m)$ problem for the WTASNP systems compared to other variants.

neurons perform state update and communication according to the rules issued by the spikes. The system as a whole evolves according to the parallelism [7].

Based on the application of some specific scenarios, researchers will introduce some biological properties from different perspectives to further enhance the expression ability of the system. For example, the MSNP systems focus on the microglia mechanism [8], the ECSNP-ER systems consider the energy request mechanism [9] and the NGCSNP systems focus on non-gated channels mechanism [10]. All of these mechanisms enhance the expression capacity of the system. Despite the enhancement in expressive power of these SNP system variants, the computational efficiency remains at the $O(n^2)$ or $O(2^n)$ level when solving the SAT problem in the NP-complete problem (as shown in Fig. 1(b)). This implies that despite the progress in expressive power of the system, its computational resource overhead is still large and is limiting for solving applications. Research on improving the computational efficiency of SNP systems is still in the development stage, so it is necessary to introduce an efficient computational approach, which is the focus of this paper.

The winner-take-all (WTA) mechanism provides an approach to reduce the computational resource expense of SNP systems. The WTA mechanism is a fundamental competitive computational model derived from biological neural systems to model the competitive dynamics between neurons, and is inspired by lateral inhibition and competition between biological neurons in the brain [11,12]. The central feature of the mechanism is that, among a set of units jointly involved in computation, only the activation behavior of the strongest response is retained, while the rest of the units are suppressed or silenced through mutual inhibition. The WTA mechanism is oriented to the strongest response, and achieves selective control of spiking disbursement through local inhibition. Because it uses only the strongest activation behavior, it is able to reduce computational redundancy to save on system overhead, and it can further improve the efficiency of the computation. The superiority of the WTA computerized system has been verified and extended in several fields. For example, applications to audio scene analysis [13] and pattern recognition based on DNA neural networks [14], and others.

For this reason, this paper introduces the WTA computational mechanism into the SNP systems and proposes the SNP systems incorporating WTA mechanism (WTASNP systems). The WTA computational system improves computational efficiency more substantially under the condition of enhanced system expressiveness. The WTASNP systems to achieve local competitive activation and inhibition behaviors (the abstract simulation structure of the system is shown in Fig. 1(a)). In WTASNP systems, neurons are divided into two categories: ordinary neurons σ_i and competing neurons ω_i. These two types of neurons cooperate to accomplish the overall computation of the system. Specifically, the ordinary neurons σ_i are responsible for input encoding and preliminary computation, whereas the competition neurons ω_i perform WTA competition and inhibitory control. For any step t, all ω_i satisfying the firing condition form a candidate set. The system then performs WTA computation on this set and selects the neuron with the largest number of spikes at step t as the winner neuron ω_{W_i}. The remaining neurons participating in the competition are regarded as loser neurons ω_{L_j}. The winner neuron fires and imposes inhibition, while the loser neurons execute forgetting and clear their internal states. The structural characteristics and computerized mechanism of this system are mainly reflected in the following aspects:

- Proving the Turing universality of the WTASNP systems. First, a formal definition of the systems is provided to clarify the basic components and operation mechanism of the system. Subsequently, by constructing system instances to generate natural numbers and simulating the instruction operation of the register as a number acceptor and generator, it is further proved that WTASNP systems have the ability to simulate the universality of the Turing machine to achieve the computational completeness in the theory.
- The superiority of the computational efficiency of WTASNP in solving SAT problems is demonstrated, and the computational resource complexity is reduced from $O(n^2)$ or $O(2^n)$ to $O(n)$. In contrast to some existing SNP variants that use 2^n or n^2 levels of computational resources when dealing with such problems, the WTASNP systems rely only on linear levels of neurons and time steps to accomplish the solving task. In the solution process, the basic concepts of the SAT problem are first formally explained. Subsequently, the solution process of the WTASNP systems is analyzed in detail, which mainly includes (1) converting Boolean

clauses into the spike input structure in the system, (2) realizing the two-layer judgment of the truth value of the variable x_i in the processing module, and (3) executing the WTA mechanism through competing neurons to realize selective activation and path inhibition. Due to the introduction of structural innovations and local WTA computation rule, the WTASNP systems exhibit higher efficiency in resource scheduling and computational path control.

2. Related work

2.1. SNP systems

Nowadays, the research on SNP focuses on the following four aspects.

Model expansion and diversity. Researchers have continuously introduced new neural mechanisms to extend the model function, including bio-inspired features such as threshold regulation [15,16], polarization [17,18], dendritic structure [19,20], inhibitory mechanism [21,22], and changes in synaptic structure [4,23]. Meanwhile, some scholars have also proposed SNP systems with nonlinear processing capability and numerical computation function from mathematical and computational perspectives [24,25]. In addition, the system is endowed with certain learning ability and adaptability by introducing weight dynamic adjustment mechanisms, such as Hebb learning rule or LTP/LTD mechanism [26,27].

Computational capability and efficiency. Related research covers the Turing completeness of SNP systems [28], function computation ability (small generality) [23,29] and language generation ability (finite and infinite languages) [30]. In recent years, the research focus has been gradually extended to the level of computational efficiency, especially in solving NP-complete problems (e.g., SAT, subset sum problems) [9,31–35]. Some variant systems, such as models introducing microglia properties and coupled neurons, are also exploring the construction of a unified framework for solving SAT [8,36]. Moreover, variant model has been constructed to investigate the efficient resolution of problems within the class PSPACE [37].

Application expansion. Thanks to the strong generalization and abstraction capabilities of SNP systems, studies have been conducted to apply them to several practical scenarios, including grid fault diagnosis [38,39], image fusion [40–43], medical image segmentation [44] classification tasks [45,46], robot control [47,48], network security [49] and time series analysis [5].

System implementation and simulation. Some studies have realized SNP variants through hardware platforms (e.g., FPGAs and GPUs) [50–55]. Some scholars have also used the P-Lingua standard framework simulator to implement the simulation of SNP variant models [4,56,57]. In addition, the Haskell language has been used to formally define the syntax and semantics of SNP systems and to implement model simulations that support arithmetic operations [58,59].

2.2. Winner-Take-All

The WTA competition mechanism is inspired by the lateral inhibition and competition of neurons in the biological brain. The WTA competition mechanism can improve the expressive power of the model. To be more specific, a WTA strategy combines spectral clustering masks from self-supervised features, leading to the SELF-MASK detector, which outperforms previous unsupervised SOD methods [60]. In neuromorphic computing, a WTA mechanism with spintronic neurons in a spiking neural network has improved digit recognition accuracy and energy efficiency [61]. An adaptive WTA framework, Neuronal Competition Groups, resolves competitive imbalances and boosts image recognition on CIFAR benchmarks [62]. A recurrent spiking WTA architecture, integrated with an SVPG-driven R-STDP framework, shows superior robustness in vision-based navigation and robotic control [63]. A decentralized k-WTA framework for robotic task allocation enables finite-time convergence and reduces motion costs in dynamic target tracking [64]. In this paper, our aim is to integrate the WTA mechanism into the SNP system to optimize the neuron excitation control through selective activation and inhibition, thus enhancing the computational efficiency of the model.

3. SNP Systems with winner-Take-All

3.1. Problem description

The existing variants of the SNP systems, when solving NP-complete problems (such as SAT), possess expressive capability. However, their resource consumption remains high, with the scale of neurons growing to $O(n^2)$ or even $O(2^n)$ as the input increases, which limits their deployment under practical resource constraints. To reduce the computational and communication overhead, this paper introduces a competitive sparse activation mechanism. The WTA mechanism helps to reduce redundant evolution from parallel branches by retaining the activation of only the dominant units within the same competitive set while suppressing the rest. Inspired by this, the paper introduces the WTA competitive mechanism and proposes the WTASNP model with WTA competition and suppression semantics. The formal definition and rule construction details of the model are provided below.

3.2. WTASNP systems definition

The following is the definition of a WTASNP system of degree $m_1 + m_2$:

$$\Pi = (O, \sigma_1, \cdots, \sigma_{m_1}, \omega_1, \cdots, \omega_{m_2}, syn, syn', in, out),$$

where:

1. $O = \{a\}$ denotes an alphabet consisting of the spike a.
2. $\sigma_i = (n_i, R_i), 1 \le i \le m_1$ denotes neurons,where: n_i denotes the number of spikes in σ_i. R_i denotes firing rules contained in σ_i in the form of $E/a^c \to a^p$. E is a regular expression over O, and $c \ge p \ge 0$.
3. $\omega_i = (n_i, R_i', R_i''), 1 \le i \le m_2$ denotes competing neurons, where $R_i' = (R_{win}, R_{los})$ represents firing rules contained in ω_i same with σ_i. R_{win} contains rules applied when ω_i as winner neurons, and R_{los} denotes forgotten rules executed as loser neurons form as $a^c \to \lambda$. R_i'' consists of rules such as $e \leftarrow e$ and $e \leftarrow e^+/e$ that allow winner neurons to release inhibitory factor e at the corresponding inhibitory synapses, blocking spikes from ordinary neurons to the competing neurons along that pathway.
4. $syn = \{(i, j)\}, 1 \le i, j \le m_1, i \ne j$ represents synaptic connections between neurons.
5. $syn' = \{\langle i, j\rangle\}, 1 \le i, j \le m_2, i \ne j$ denotes inhibitory synaptic connections between competing neurons ω_i and ordinary neurons σ_i.
6. in, out are the labels of the input and output neurons.

A WTASNP system consists of m_1 interconnected ordinary neurons, and m_2 competing neurons connected to the ordinary neurons. The input neurons receive spike inputs from external sources, while the output neurons output the spikes to the environment, thus reflecting the computational results of the system.

Each neuron σ_i contains a number of rules R_i and spikes n_i. These rules are usually of the form $E/a^c \to a^p$, where E is a regular expression that needs to be satisfied in order to use the rule. The rule indicates that the neuron when executed, consumes c spikes and transmits p spikes to the succeeding neurons of the neuron σ_i. In addition, there are other forms of representation of the rule within the neuron: first, when $E = a^c$ the rule can be simplified as $a^c \to a^p$. second, when $a^p = a^0$, a^p can be represented as a λ symbol, in which case the rule representation is of the form $E/a^c \to \lambda$, known as the rule of forgetting. It means that the neuron σ_i, although it consumes c spikes, does not deliver spikes to its successor neuron, but only performs an spike-consuming action.

The competing neuron ω_i has the same rule form and spikes as the ordinary neuron σ_i, but there are differences in the application of the rules. Specifically, the WTA scope is defined as a competition group $G = \{\omega_1(a^{n_1}), \omega_2(a^{n_2}), \ldots, \omega_m(a^{n_m})\}$. The system enters the WTA decision phase if and only if, at the same global time step t , there are at least two candidate neurons in the group that simultaneously satisfy the firing condition. That is, there exists a usable rule R_i' such that ω_i has firing eligibility at time t. Let $A(t) = \{\omega_i \in G \mid \exists R_i' \text{ enabled at } t\}$. When $|A(t)| = 0$, there is no firing within the group. When $|A(t)| = 1$, no competition is needed, and the only candidate neuron directly fires. When $|A(t)| \ge 2$, the WTA function is triggered: $WTA(A(t)) = \omega_{W_i}$, which means that for each candidate neuron in $A(t)$, the number of competitive spikes $a^{n_i}(t)$ is calculated. The neuron with the maximum number of spikes is selected as the winner neuron ω_{W_i} , while the others are considered loser neurons ω_{L_i}. During each competition, the comparison results of the competing neurons can be categorized into the following two cases:

1. When a winner neuron is selected: systems calculate $WTA\{\omega_1(a^{n_1}) , \omega_2(a^{n_2}), \cdots, \omega_m(a^{n_m})\}$ which results in selecting a winner neuron. All the winner neurons ω_{W_i} will then trigger the R_{win} rule, while all other loser neurons ω_{L_j} execute the R_{los} $(a^k \to \lambda)$ rule, consuming their respective spikes. In addition, for ω_{w_i} will release the inhibitory factor e via R'' and activate the corresponding inhibitory synapses, thus preventing ordinary neurons in that pathway from transmitting spikes to neurons in the competing neurons.
2. When a winner neuron cannot be selected: when the calculation of $WTA\left\{\omega_1(a^{n_1}) , \omega_2(a^{n_2}), \cdots, \omega_m(a^{n_m})\right\}$ fails to select the winner neuron, it means that the number of spikes of the competing neurons ω_i is equal. In this case, all ω_i initiate R_{win} rule but do not release the inhibitory factor e.In subsequent steps, the states of the neurons σ_i and ω_i will continue to update according to the predefined rules until ω_i executes the WTA computation to select ω_{W_i}.

Usually, the synaptic connection syn denotes the directed edge (i, j) from σ_i to σ_j. In the WTASNP system, the inhibitory synapses syn' controlled by the winner neuron ω_{w_i} as the initiator are categorized into two types: first-degree inhibitory synapses and cascade inhibitory synapses. 1) First-degree inhibitory synapses: when ω_{w_i} has been selected as the winning neuron after the WTA calculation, if in the previous step, the neuron σ_i had spikes occurring to it through the connection $(\sigma_i, \omega_{W_i}) \in syn$, the corresponding inhibitory synapse $\left\langle \omega_{W_i}, \sigma_i \right\rangle \in syn'$ is activated. This synapse then receives the inhibitory factor e from ω_{w_i}, which inhibits the issuance of further spikes from σ_i to ω_{W_i} until the end of the system computation. 2) Cascade inhibitory synapses: when the σ_i end of a primary inhibitory synapse $\left\langle \omega_{W_i}, \sigma_i \right\rangle \in syn'$ is connected by a dotted line to another inhibitory synapse $\left\langle \omega_{L_j}, \sigma_j \right\rangle$, the two form a cascade inhibitory structure $\left\{ \left\langle \omega_{W_i}, \sigma_i \right\rangle, \left\langle \omega_{L_j}, \sigma_j \right\rangle \right\}$. In this structure, after either end is activated and receives the inhibitory factor e, the other end will also be linked to wake up and inhibit the spikes delivery from σ_j to ω_{L_j} until the end of system computation.

The system represents the result of the computation by the time difference between two consecutive spikes sent by the neuron σ_{out} to the environment. For example, the time difference can be expressed as $(t + n + 1) - (t + 1) = n$.

A WTASNP system is parallel. From the dimension of system neurons, at each time step, both ordinary neurons and competing neurons can operate synchronously at both levels as long as the ignition condition is satisfied. From the perspective of rule execution, at each time step, all ignitable neurons execute a rule to facilitate spike issuance and state update. However, due to the nondeterministic nature of the system, when multiple executable rules exist for a neuron within the same time step, only one of them will be randomly selected for execution.

The family of natural number sets generated by the WTASNP systems with at most m neurons is denoted by $N_2WTASNP_m$ denotes. The family of natural number sets accepted by the WTASNP systems with at most m neurons is denoted by $N_{acc}WTASNP_m$. If the number of neurons is countable but unknown, m is usually replaced by $*$.

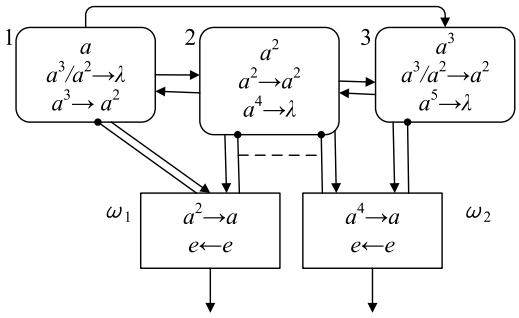


Fig. 2. An illustrative example of WTASNP systems.

Table 1
Generate all natural number $n = (n+2) - 2$.

Step	σ_1	σ_2	σ_3	ω_1	ω_2	en
0	1	2	3	$(0, e)$	$(0, e)$	—
1	1	2 $a^2 \to a^2$	3 $a^3/a^2 \to a^2$	$(0, e)$	$(0, e)$	—
2	3 $a^3/a^2 \to \lambda$	2 $a^2 \to a^2$	3 $a^3/a^2 \to a^2$	$(2, e)$ R_{los}	$(4, e)$ R_{win}	1, first
3	3 $a^3/a^2 \to \lambda$	2 $a^2 \to a^2$	3 $a^3/a^2 \to a^2$	$(0, e)$	—	—
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$n+1$	3 $a^3 \to a^2$	2 $a^2 \to a^2$	3 $a^3/a^2 \to a^2$	$(0, e)$	—	—
$n+2$	2	4 $a^4 \to \lambda$	5 $a^5 \to \lambda$	$(2, e)$ R_{win}	—	1, second
$n+3$	2	0	0	—	—	—

3.3. Example

In this subsection, the operation mechanism of the WTASNP systems will be analyzed in stepwise manner through a specific case of generating natural numbers shown in Fig. 2.

In the initial state, neurons σ_1, σ_2, and σ_3 contain a, a^2 and a^3 spikes respectively. At step 1, σ_2 sends a^2 spikes to σ_3 and ω_1, and ω_2 using the rule $a^2 \to a^2$. At the same time, σ_3 sends a^2 spikes to σ_2 and ω_2 using the rule $a^3/a^2 \to a^2$.

At step 2, the competing neurons ω_1 and ω_2 receive a^2 and a^4 spikes respectively. They are in an excitable state and therefore belong to $A(2) = \{\omega_1, \omega_2\}$, thus starting the WTA competition computation, $WTA\{\omega_1(a^2), \omega_2(a^4)\}$. Competition result: ω_2 becomes the winner neuron, denoted as ω_{W_2}, while ω_1 becomes the loser neuron, denoted as ω_{L_1}. ω_{L_1} will execute the rule of forgetting to consume a^2 spikes, and ω_{W_2} will execute the rule $a^4 \to a$ to send the first spike to the environment. At the same time, cascading inhibitory synapses $\left\{\left\langle \omega_{W_2}, \sigma_3 \right\rangle, \left\langle \omega_{W_2}, \sigma_2 \right\rangle, \left\langle \omega_{W_1}, \sigma_2 \right\rangle\right\}$ is activated and receives an inhibitory factor from ω_2 to prevent σ_2 from sending spikes to ω_1 and ω_2, and σ_3 from sending spikes to ω_2. Therefore, although σ_2 and σ_3 will excite the same spike rules as at step 1, they will not be able to send spikes to the competing neurons. In this case, if σ_1 applies the forgetting rule $a^3/a^2 \to \lambda$ for $n-1$ consecutive times since step 2. At step $n+1$, σ_1 executes the spike rule $a^3/a^2 \to a$ and sends a spike to σ_2 and ω_1. Subsequently, at step $n+2$, the spike cycle that originally existed between σ_1, σ_2 and σ_3 will be broken. At this point, σ_1 has only a^2 spikes, which can no longer be energized, and σ_2 and σ_3 have a^4 and a^5 spikes, respectively, which will be consumed by the forgetting rule. Eventually, ω_1 has a^2 spikes and becomes the only winner neuron, noted as ω_{W_1}, and excites the spike rule to send spikes to the environment for a second time. After the system has gone through the above process, the computational calculations satisfy the relation: $n = (n+2) - 2$. Therefore, the system can generate all natural numbers. Table 1 shows the spike and rule evolution process of ordinary and competing neurons when this example generates natural numbers.

4. Turing universality of WTASNP systems

Turing universality is an important measure of the capabilities of a computing system. Since the register is a Turing complete model of computation, becomes a key criterion for testing the Turing generality of a system [65]. In addition, SNP systems are often demonstrated by emulating a register as a number generating/accepting device.

A register machine is formally defined as $M = (m, H, l_0, l_h, I)$, where m denotes the number of registers and I represents a finite set of instructions. Each instruction is labeled with an identifier from the set H, in which l_0 and l_h correspond to the initial and stopping instructions, respectively. The instruction set includes the following types:

1. ADD: $l_i : (ADD(r), l_j, l_k)$. This instruction increases the value in the register r by one and proceeds non-deterministically to l_j or l_k.
2. SUB: $l_i : (SUB(r), l_j, l_k)$. If the current value in the register $r \geq 1$, it is decreased by one and the next instruction is l_j; otherwise, the control passes to l_k.
3. HALT: l_h : HALT. This marks the end of the computation process.

The computation begins with instruction l_0 and progresses according to the defined semantics. It terminates when the control reaches the halting instruction l_h. While ADD and SUB operations may be invoked multiple times during execution, the HALT instruction is applied exactly once.

4.1. WTASNP systems as number-generating devices

Theorem 1. $N_2WTASNP_*^4 = NRE$.

Proof. One can easily observe that $N_2WTASNP_*^4 \subseteq NRE$. To demonstrate $N_2WTASNP_*^4 \supseteq NRE$. We constructed a WTASNP system that simulates M operating in generative mode capable of generating NRE the set of Turing computable numbers. At the beginning of the computation, all registers do not store spikes, in digital generation mode. During computation, the ordinary neuron σ_i, the neuron σ_{l_i} and the competing neuron ω_i work together to complete the simulation task. Where neuron σ_i denotes the register r and neuron σ_{l_i} contains the instruction l_i. The competing neuron ω_i denotes the competitive computation of the algorithm is usually linked to the next instruction through instructions l_i and l_k. As the simulation progresses, the ADD and SUB instructions may be executed multiple times. When the system reaches instruction l_h, the computation terminates, and the final result is stored in register 1. The numeric value in a register is represented by the number of spikes it contains: for example, four spikes encode the value 1.

ADD module for simulating ADD instruction $l_i : (ADD(r), l_j, l_k)$ (Fig. 3):

In the starting state, both competing neurons ω_1 and ω_2 carry a^4 spikes. Table 2 shows the system execution process of the system simulating $l_i : (ADD(r), l_j, l_k)$ instructions. Assuming that the neuron σ_{l_i} receives the a^4 spikes at step $t+1$, it fires and the spikes is delivered to the neurons σ_r, $\sigma_{l_{i1}}$ and $\sigma_{l_{i2}}$. In step $t+2$, the number of spikes in neuron σ_r is increased by four, while the rule $a^4 \to a^3$ in $\sigma_{l_{i2}}$ is energized, and there are two possible state transition paths for $\sigma_{l_{i1}}$:

1. If $a^4 \to a^4$ is executed, the a^4 spikes goes to ω_1. In step $t+3$, ω_2 also receives the a^3 spikes. At this point, both competing neurons ω_1 and ω_2 are activated and enter WTA competition $WTA\{\omega_1(a^8), \omega_2(a^7)\}$. Competition result: ω_2 becomes the loser neuron ω_{L_2}, enforcing the forgetting rule for a^7 spikes. ω_1 becomes the winner neuron ω_{W_1} , triggers the spike rule $a^8 \to a^4$ and sends a^4 spikes to σ_{l_j} so that σ_{l_j} is activated to simulate l_j. At the same time, the inhibitory synapse $\langle \omega_{W_1}, \sigma_{l_{i1}} \rangle$ is activated and receives inhibitory factor e released from rule $e \leftarrow e$ in ω_{W_1}.
2. If $a^4 \to a$ is executed, the a spike goes to ω_1. In step $t+3$, ω_2 also receives the a^3 spikes. At this point, the competing neurons ω_1 and ω_2 are still activated and enter WTA competition $WTA\{\omega_1(a^5), \omega_2(a^7)\}$. Competition result: ω_1 becomes the loser neuron ω_{L_1}, enforcing the forgetting rule for the a^5 spikes. ω_2 becomes the winner neuron ω_{W_2} , triggers the spike rule $a^7 \to a^4$, and sends a^4 spikes to σ_{l_k}, causing σ_{l_k} to be activated, thus simulating l_k. At the same time, inhibitory synapses $\langle \omega_{W_2}, \sigma_{l_{i2}} \rangle$ is activated and receives inhibitory factor e released from rule $e \leftarrow e$ in ω_{W_2}.

SUB module for simulating an SUB instruction $l_i : (SUB(r), l_j, l_k)$(Fig. 4):

This module is initialized with the competing neuron ω_1 having a^3 spikes and the neuron σ_r having $a^{4n} (n \in \mathbb{N})$ spikes. The dynamic process of the system begins with step $t+1$, which assumes that the neuron $\sigma_{l_{i1}}$ receives a^4 spikes and triggers the ignition of the rule $a^4 \to a^3$ ignition process. In step $t+2$, ω_1 has a^6 spikes and σ_r has a^{4n+3} spikes. Therefore, the competing neurons ω_1 and σ_r are activated and start the WTA competition, which is formally represented as $WTA\{\omega_1(a^6), \sigma_r(a^{4n+3})\}$. The outcome of the competition varies depending on the number of spikes within σ_r, as analyzed below (Table 3):

1. If $\sigma_r(a^{4n+3})(n > 0)$, then ω_1 becomes the loser neuron ω_{L_1} and its a^6 spikes will be consumed. After σ_r becomes a winner neuron ω_{W_r} and triggers the rule $a^3(a^4)^+/a^7 \to a^4$, the number of spikes in σ_r decreases from $4n+3$ to $4(n-1)$, i.e., the value decreases by one. At the same time, the inhibitory synapse $\langle \omega_{W_r}, \sigma_{l_i} \rangle$ is activated and receives the inhibitory factor released from rule $e \leftarrow e$ in ω_{W_r}. In addition, σ_{l_j} receives a^4 spikes from σ_r and is activated to simulate l_j.
2. If σ_r has a^3 spikes, it becomes the loser neuron ω_{L_r}, and the a^3 spikes are consumed. ω_1 becomes the winner neuron ω_{W_1}, and the rule $a^6 \to a^4$ is applied to transmit a^4 spikes to σ_{l_k}. At this point, the inhibitory synapse $\langle \omega_{W_1}, \sigma_{l_i} \rangle$ is activated and receives the inhibitory factor e released from rule $e \leftarrow e$ in ω_{W_1}. Finally, σ_{l_k} receives a^4 spikes and is activated to simulate l_k.

FIN module for outputting the result of a computation (Fig. 5):

Table 2
$l_i : (ADD(r), l_j, l_k)$ is simulated by the ADD module and the computations are transmitted to the instructions l_j and l_k.

Step	σ_{l_i}	$\sigma_{l_{i1}}$	$\sigma_{l_{i2}}$	ω_1	ω_2	σ_r
t	0	0	0	$(4, e)$	$(4, e)$	$4n$
$t+1$	4 $a^4 \to a^4$	0	0	$(4, e)$	$(4, e)$	$4n$
$t+2$	0	4	4 $a^4 \to a^3$	$(4, e)$	$(4, e)$	$4(n+1)$
$\sigma_{l_{i1}} : a^4 \to a^4$, σ_{l_j} will simulate instruction l_j:						
$t+3$	0	0	0	$(8, e)$ R_{win}	$(7, e)$ R_{los}	$4(n+1)$
$t+4$	0	0	0	0	$(0, e)$	$4(n+1)$
$\sigma_{l_{i1}} : a^4 \to a$, σ_{l_k} will simulate instruction l_k:						
$t+3$	0	0	0	$(5, e)$ R_{los}	$(7, e)$ R_{win}	$4(n+1)$
$t+4$	0	0	0	$(0, e)$	0	$4(n+1)$

Table 3
$l_i : (SUB(r), l_j, l_k)$ is simulated by the SUB module and the computations are transmitted to the instructions l_j and l_k.

Step	σ_{l_i}	σ_r	ω_1	σ_{l_j}	σ_{l_k}
t	0	$(4n, e)$	$(3, e)$	—	—
$t+1$	4 $a^4 \to a^3$	$(4n, e)$	$(3, e)$	—	—
$t+2$	0	$(4n+3, e)$	$(6, e)$	—	—
when σ_r has $4n+3$ $n>1$ spikes:					
$t+2$	0	$(4n+3, e)$ R_{win}	$(6, e)$ R_{los}	—	—
$t+3$	0	$4(n-1)$	$(0, e)$	4	—
when σ_r only has 3 spikes:					
$t+2$	0	$(3, e)$ R_{los}	$(6, e)$ R_{win}	—	—
$t+3$	0	$(0, e)$	0	—	4

Table 4
Case 1: FIN model to simulate instruction l_h, when σ_1 has $4n+1 (n>1)$ spikes.

Step	σ_{l_h}	σ_1	σ_{out}	en
t	0	$4n$	$(0, e^+)$	—
$t+1$	4	$4n$	$(0, e^+)$	—
$t+2$	0	$4n+1$ $a^5(a^4)^+/a^4 \to \lambda$	$(1, e^+)$ $a \to a$	1,first
$t+3$	0	$4n-3$ $a^5(a^4)^+/a^4 \to \lambda$	$(0, e^+)$	—
⋮	⋮	⋮	⋮	⋮
$t+n+1$	0	5 $a^5 \to a$	$(0, e^+)$	—
$t+n+2$	0	0	$(1, e^+)$ $a \to a$	1,second
$t+n+3$	0	0	$(0, e^+)$	—

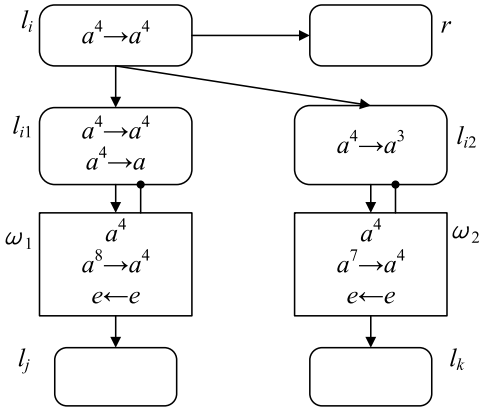


Fig. 3. ADD module to simulate instruction $l_i : (ADD(r), l_j, l_k)$.

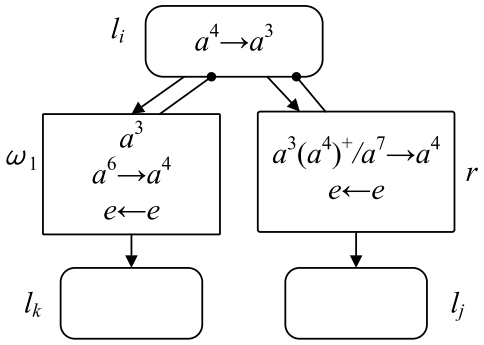


Fig. 4. SUB module to simulate instruction $l_i : (SUB(r), l_j, l_k)$.

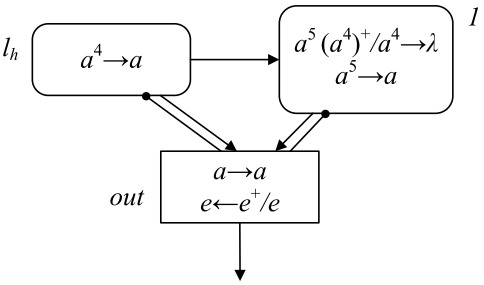


Fig. 5. FIN moudule.

Table 5
Case 2: FIN moudule to simulate instruction l_h, when σ_1 has 4 spikes.

Step	σ_{l_h}	σ_1	σ_{out}	en
t	0	4	$(0, e^+)$	—
$t+1$	4 $a^4 \to a$	4	$(0, e^+)$	—
$t+2$	0	5 $a^5 \to a$	$(1, e^+)$ $a \to a$	1,first
$t+3$	0	0	$(1, e^+)$ $a \to a$	1,second
$t+4$	0	0	$(0, e^+)$	—

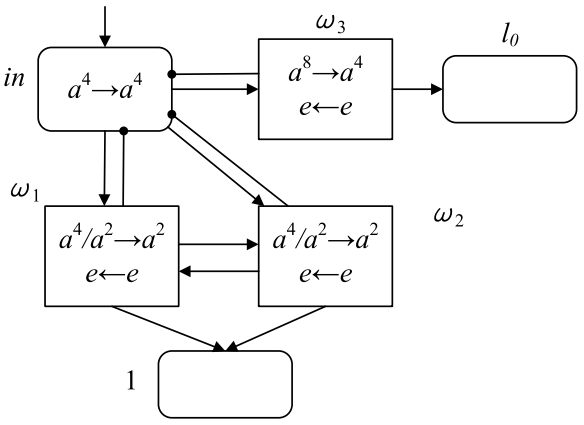


Fig. 6. INPUT module.

The main function of this module is to receive the stop command and output the calculation result. In the initial state, the neuron σ_1 contains a^{4n} spikes, while the competing neuron contains only output neuron σ_{out}. Therefore σ_{out} is always the winner neuron $\omega_{W_{out}}$ in the module. Since there are no other competing neurons in the system, σ_{out} is always used as the winner neuron and is written as $\omega_{W_{out}}$. The computational process is analyzed as follows.

Assuming that at step $t+1$, σ_{l_h} receives a^4 spikes, the simulation process of the departure stop command l_h. σ_{l_h} sends a spike to σ_1 and σ_{out} by rule $a^4 \to a$. In step $t+2$, σ_{out} is activated upon receipt of the spike and becomes the winner neuron $\omega_{W_{out}}$, and because of its winning status, $\omega_{W_{out}}$ triggers rule $a \to a$ to send the first output spike to the environment. At this point, the inhibitory synapse $\left\langle \omega_{W_{out}}, \sigma_{l_h} \right\rangle$ is activated and receives the inhibitory factor e (generated by rule $e \leftarrow e^+/e$) released from $\omega_{W_{out}}$. The number of spikes of the neuron σ_1 is increased to a^{4n+1}, and depending on the number of spikes, the computational process is divided into two cases:

1. (Table 4) If the number of spikes in σ_1 is 5 ($n=1$) the rule $a^5 \to a$ will be applied to deliver a spike to σ_{out}. In step $t+3$, σ_{out} is again activated as the winner neuron $\omega_{W_{out}}$ and the rule $a \to a$ is applied, sending a second output spike to the environment. The result of the calculation is $3-2=1$. At the same time, the inhibitory synapse $\left\langle \omega_{W_{out}}, \sigma_1 \right\rangle$ is activated and receives the inhibitory factor e released from $\omega_{W_{out}}$.
2. (Table 5) If the number of spikes in σ_1 is $4n+1 (n>1)$, the rule $a^5(a^4)^+/a^4 \to \lambda$ is continually applied until the following condition is met. At step $t+n+1$, the number of spikes in σ_1 drops to 5, then a spike is delivered to σ_{out}. At step $t+n+2$, σ_{out} again becomes the winner neuron $\omega_{W_{out}}$ and applies the rule $a \to a$ and sends second output spike. The result is calculated as $(n+2)-2=n$. Meanwhile, the inhibitory synapse $\left\langle \omega_{W_{out}}, \sigma_1 \right\rangle$ is activated and receives inhibitory factor e released from $\omega_{W_{out}}$.

Based on the above analyses, the system is able to generate NRE sets. □

4.2. WTASNP systems as number-accepting devices

Theorem 2. $N_{acc}WTASNP_*^2 = NRE$

Proof. Similar to the proof of Theorem 1, $N_{acc}WTASNP_*^2 \subseteq NRE$ is straightforward. To prove $N_{acc}WTASNP_*^2 \supseteq NRE$, we construct a WTASNP system operating in digital acceptance mode to simulate the machine M. The construction consists of the following three modules: 1) Input module: This module receives a spike sequence as input, where the input number is stored in register 1 in the form of spikes. Initially, all registers are empty and contain no spikes. 2) ADD module: This module simulates instruction $l_i : (ADD(r), l_j)$, which increases the value in register r by one and then deterministically branches to instruction l_j. This behavior differs from the instruction $l_i : (ADD(r), l_j, l_k)$ described in Theorem 1. 3) SUB module: As illustrated in Fig. 3, this module handles the subtraction operation. The simulation of the corresponding instructions is described in detail below.

INPUT module for initializing a computation (Fig. 6):

Assuming that the input value is n, the value is encoded as spike sequence of the form $a^4 0^{n-1} a^4$. The input signal is received by the neuron σ_{in}, and the competing neurons includes ω_1, ω_2 and ω_3. The calculation process is shown in Table 6. In step $t+1$, when σ_{in} receives a^4 from the spike sequence, it indicates that the computation of the input value n begins. Subsequently, σ_{in} sends spikes to the competing neurons by rule $a^4 \to a^4$.

In step $t+2$, ω_1, ω_2 and ω_3 all receive a^4 spikes and are activated for WTA calculation, $WTA\{\omega_1(a^4), \omega_2(a^4), \omega_3(a^4)\}$. Since there are no clear winner and loser neurons during competition, the spike rules of all neurons can be ignited without triggering the release of inhibitory synapses and inhibitory factors. As a result, the cascade inhibitory synapses $\{\langle \omega_1, \sigma_{in} \rangle, \langle \omega_2, \sigma_{in} \rangle, \langle \omega_3, \sigma_{in} \rangle\}$, cannot be energized, and the competing neurons can continue to receive spikes from σ_{in}. Where the a^4 spikes in ω_3 is unable to ignite the spike rule, so the spike remains inside the neuron. Instead, ω_1 and ω_2 send a^2 spikes to each other using the rule $a^4/a^2 \to a^2$.

Table 6
INPUT module accepts $a^4 0^{n-1} a^4$ spike train.

Step	σ_{in}	ω_1	ω_2	ω_3	σ_1	σ_{l_0}
t	0	$(0,e)$	$(0,e)$	$(0,e)$	—	—
$t+1$	4 $a^4 \to a^4$	$(0,e)$	$(0,e)$	$(0,e)$	—	—
$t+2$	0	$(4,e)$ R_{win}	$(4,e)$ R_{win}	$(4,e)$	—	—
$t+3$	0	$(4,e)$ R_{win}	$(4,e)$ R_{win}	$(4,e)$	4	
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$t+n+1$	4 $a^4 \to a^4$	$(4,e)$ R_{win}	$(4,e)$ R_{win}	$(4,e)$	$4(n-1)$	—
$t+n+2$	0	$(6,e)$ R_{los}	$(6,e)$ R_{los}	$(8,e)$ R_{win}	$4n$	—
$t+n+3$	0	$(0,e)$	$(0,e)$	0	$4n$	4

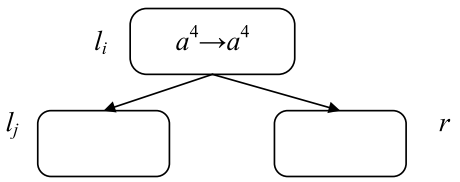


Fig. 7. ADD moudule to simulate instruction $l_i : (ADD(r), l_j)$.

Table 7
The process of executing $l_i : (ADD(r), l_j)$.

step	σ_{l_i}	σ_r	σ_{l_j}
t	0	$4n$	0
$t+1$	4 $a^4 \to a^4$	$4n$	0
$t+2$	0	$4(n+1)$	4

In step $t+3$, σ_1 receives a^4 spikes from ω_1 and ω_2 and they will continue to fire and send spikes. Until step $t+n+1$, the second a^4 spikes in the spike sequence is input to σ_{in}, causing σ_{in} to fire again and send spikes to the succeeding neurons.

In step $t+n+2$, σ_1 receives a^{4n} spikes in total, indicating that the input value n has been delivered. At the same time, all competing neurons receive a^4 spikes from σ_{in}, triggering a new WTA competition $WTA\{\omega_1(a^6), \omega_2(a^6), \omega_3(a^8)\}$. At this point, ω_3 becomes the winner neuron, notated as ω_{W_3}, while ω_1 and ω_2 become loser neurons and consume the a^6 spikes. ω_{W_3} will initiate rule $a^8 \to a^4$ to output a^4 spikes to σ_{l_0}. At the same time, the inhibitory synapse $\langle \omega_{W_3}, \sigma_{in} \rangle$ is activated and receives the inhibitory factor released from rule $e \leftarrow e$ in ω_{W_3}. Eventually, σ_{l_0} receives the a^4 spikes and initiates the analog instruction l_0.

ADD module for simulating a deterministic ADD instruction $l_i : (ADD(r), l_j)$ (Fig. 7)(Table 7):

When the neuron σ_{li} receives a^4 spikes, it means that this module starts to start the rule $a^4 \to a^4$ to simulate the instruction l_i. Afterwards, σ_r receives a^4 spikes proves that the value in register r increases by 1, σ_{l_j} receives a^4 spikes can start to simulate the instruction l_j.

SUB module:

The SUB module is the same as the SUB module in Theorem 1, and the arithmetic procedure is not repeated here.

Based on the above analyses, the system is able to generate NRE sets. □

5. WTASNP systems to solve SAT problems

5.1. The SAT problems

The Boolean satisfiability problem (SAT problem) is a classical NP-complete problem in computational complexity theory [66]. It asks whether there exists an assignment of Boolean values to a given formula such that the formula evaluates to true. SAT instances are

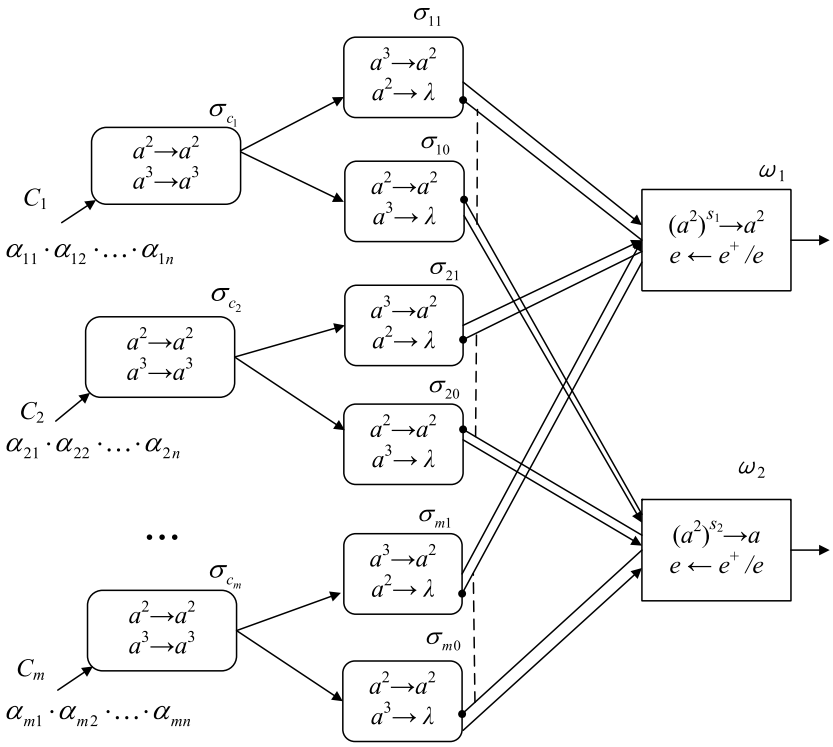


Fig. 8. Π_{sat} for SAT problem.

typically expressed in Conjunctive Normal Form (CNF), which is a conjunction of multiple clauses, where each clause is a disjunction of literals ($l_i = x_i$ or $\neg x_i$).

Let the Boolean variable set be $\{x_1, x_2, \cdots, x_n\}$, and each clause is written as $C_i = (l_1 \vee l_2 \vee \cdots \vee l_k), 1 \le i \le n$, where l_i denotes a literal. A CNF formula is then expressed as $C_1 \wedge C_2 \wedge \cdots \wedge C_m$. A formula is satisfiable if there exists at least one assignment of variables such that every clause C_i contains at least one literal l_i evaluated true.

Based on the computational structure of the WTASNP systems, we designed system Π_{sat} (Fig. 8) to model and solve the SAT problem. The input parameters in Π_{sat} are the same as those of the unified solution model, both containing variables x_i and clauses C_m.

5.2. Problem encoding

In the system Π_{sat}, it is the neuron σ_{c_i} $(1 \le i \le m)$ that is responsible for receiving the input signal. Specifically, at each moment, σ_{c_i} is used to obtain the literal information (x_i or $\neg x_i$) input in the corresponding clause C_i $(1 \le i \le m)$. For each variable x_i, neuron σ_{i1} $(1 \le i \le m)$ is used to determine whether the true value of x_i is true, and neuron σ_{i0} $(1 \le i \le m)$ is used to determine whether the true value of x_i is false. In the competing neurons ω_1 and ω_2 are filtered through the WTA competition mechanism, and finally neuron ω_{W_i} outputs the truth value of variable x_i that can make the maximum number of clauses C_i true.

Since the input parameters of the SAT problem need to be converted into the form of spike signals for input to the system Π_{sat}. The variable x_i in clause C_i is represented by the corresponding literal information $\alpha_{ij}, (1 \le i \le m, 1 \le j \le n)$ and is encoded by different numbers of spikes. To ensure that neighboring variables x_i and x_{i+1} can be correctly distinguished in the input signal, $\}\}0\varepsilon$ is inserted between them as a separator. For example, for the Boolean formula $\gamma = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_2)$, the corresponding input spike encoding is $\{C_1 = 30202, C_2 = \ 00200\}$.

$$\alpha_{ij} = \begin{cases} 0, & \text{if } C_i \text{ doesn't contains } x_i \text{ or } \neg x_i; \\ a^3, & \text{if } C_i \text{ contains } x_i; \\ a^2, & \text{if } C_i \text{ contains } \neg x_i. \end{cases}$$

When the computation starts, the spike encoding of each clause C_i is input to the corresponding neuron σ_{ci}. First, the system judges the first literal α_{i1} in clause C_i: (i) if its corresponding variable is x_1, this input is encoded as a^3 spikes and triggers the rule $a^3 \to a^3$. (ii) if its corresponding variable is $\neg x_1$, it is encoded as a^2 spikes and triggers the rule $a^2 \to a^2$. Next, the neurons σ_{i1} and σ_{i0} receive a^2 or a^3 spikes from σ_{c_i} to further determine the truth state of the variable x_1:

If σ_{i1} and σ_{i0} receive a^3 spikes, it means that the variable of input literal α_{i1} is x_1:

- σ_{i1}: x_1 truth value is true (α_{i1} is *true*), then σ_{i1} triggers the rule $a^3 \to a^2$ and sends a^2 spikes to ω_1.
- σ_{i0}: x_1 truth value is false (α_{i1} is $false$), σ_{i1} executes the forgetting rule.

If σ_{i1} and σ_{i0} receive a^2 spikes, the variable that indicates the input literal α_{i1} is $\neg x_1$:

- σ_{i1}: x_1 truth value is true (α_{i1} is $false$), σ_{i1} performs the forgetting rule.
- σ_{i0}: x_1 truth value is false (α_{i1} is *true*), then σ_{i0} triggers the rule $a^2 \to a^2$ and sends a^2 spikes to ω_2.

Through the double-level judgment mechanism described above, the variable x_i in each clause outputs the truth value that makes α_{ij} true. The reason for this judgment is that the CNF corresponding to each clause C_i can be proved to be true if and only if there is at least one literal α_{ij} in that clause that is true.

From that, neuron ω_1 receives a^{2s_1} spikes from σ_{i1}, while neuron ω_2 receives a^{2s_2} spikes from σ_{i0}. Subsequently, the competing neurons are activated to perform the competitive computation $WTA\left\{\omega_1(a^{2s_1}), \omega_2(a^{2s_2})\right\}$. This competitive process essentially compares the number of cases that make clause C_i true when the variable x_i takes *true* or $false$ truth values to determine the final result of the computation.

- When ω_1 is the winner neuron ω_{W_1}, it represents that the number s_1 of clauses that make the clause true when the x_i truth value is *true* is greater than the number s_2 when the x_i truth value is $false$. According to the WTA rule, ω_{L_2} consumes the a^{2s_2} spike while ω_{W_1} releases the spike to the environment *en* by the rule $a^{2s_1} \to a^2$ to release the spike. In addition, cascade inhibitory synapses $\left\{\left\langle\omega_{W_1}, \sigma_{i1}\right\rangle, \left\langle\omega_{W_2}, \sigma_{i0}\right\rangle \ldots, \left\langle\omega_{W_1}, \sigma_{j1}\right\rangle, \left\langle\omega_{W_2}, \sigma_{j0}\right\rangle\right\}$ $(1 \le i, j \le m)$ is activated and receives a signal from ω_{W_1} in rule $e \leftarrow e^+/e$ released inhibitory factor. These inhibitory synapses act to prevent σ_{i1} from continuing to send spikes to ω_1. However, the system can still receive other spike inputs that are not part of the cascade inhibitory path.
- When ω_2 is the winner neuron ω_{W_2}, denotes that the number s_2 of clauses that make the clause true when the truth value of x_i is $false$ is greater than the number s_1 when the truth value of x_i is *true*. According to the WTA rule, ω_{L_1} consumes the spike of a^{2s_1} while ω_{W_2} releases a spike to the environment *en* through the rule $a^{2s_2} \to a$ release spike. At the same time, the cascade inhibits synapses $\left\{\left\langle\omega_{W_2}, \sigma_{i0}\right\rangle, \left\langle\omega_{W_1}, \sigma_{i1}\right\rangle, \ \ldots, \left\langle\omega_{W_2}, \sigma_{j0}\right\rangle, \left\langle\omega_{W_1}, \sigma_{j1}\right\rangle\right\}$ $(1 \le i, j \le m, i \ne j)$ is activated and receives the data from ω_{W_2} in rule $e \leftarrow e^+/e$ released inhibitory factor. These cascading inhibitory synapses act to prevent σ_{i0} from continuing to send spikes to ω_2 and σ_{i1} to ω_1. However, the system can still receive other spike inputs that are not part of the cascade inhibition pathway.
- If the competing neurons fails to produce winner neurons and loser neurons, it means that the number of clauses that make α_{ij} true is equal when x_i is *true* or $false$, i.e., $s_2 = s_1$. In this case, ω_1 and ω_2 activate the spike rule at the same time and send a total of a^3 spikes to the environment *en*. In addition, since competition does not produce a dominant outcome, inhibitory factor is not triggered and inhibitory synapses are not activated.

When the environment *en* receives different spikes from ω_1 and ω_2, it indicates different truth cases of α_{ij} when solving:

$$en = \begin{cases} a^2, & \text{the truth value of } x_i \text{ is } true \text{ making } C_i \ true, \\ a, & \text{the truth value of } x_i \text{ is } false \text{ making } C_i \ true, \\ a^3, & x_i \text{ truth values of } true/false \text{ both make } C_i \ true. \end{cases}$$

Therefore, the system will evaluate x_i to be evaluated in order to find a solution that satisfies the Boolean formula γ as soon as possible. Based on the above analysis, the system Π_{sat} can solve the SAT problem with total $3m + 2$ neurons. Under the optimal case, the system can realize a 3-step solution when the input literal α_{i1} for each clause C_i of the input Boolean formula is the same (both x_i or $\neg x_i$). The maximum solving time limit is $2n + 1$ steps, and a judgment is performed on each variable x_i in order to make the input Boolean formula true.

5.3. Example - detailed analysis

In order to further elucidate the operational mechanisms of the system, the following is illustrated by means of specific Boolean formula examples. Consider a Boolean formula containing three clauses and four variables:

$$\gamma_1 = (x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee x_4)$$

During the computation, neurons $\sigma_{c_1}, \sigma_{c_2}$ and σ_{c_3} are responsible for receiving and processing the spike inputs of the three clauses, whose corresponding encodings are $\left\{C_1 = 3020003, C_2 = 0020303, C_3 = 2,030,303\right\}$. Table 8 depicts the cooperation process of each neuron when γ_1 is solved in Π_{sat}.

In step 1, the system accepts the first literal α_{i1}(x_1 or $\neg x_1$) in each clause and begins to compute the state of the neuron σ_{ci}. In step 2, σ_{i1} and σ_{i0} receive a^2 or a^3 spikes from σ_{c_i} and initiate the process of determining the truth value of the variable x_1. During the process, σ_{11} sends a^2 spikes to ω_1 based on the rule $a^3 \to a^2$, while σ_{30} sends a^2 spikes to ω_2 using the rule $a^2 \to a^2$ to further advance the calculation process shown in Fig. 9.

1. Competition determination of variable x_1: In step 3, the competing neurons turn on the WTA competition computation, i.e., they execute $WTA\left\{\omega_1(a^2), \omega_2(a^2)\right\}$, and since the intensity of the spikes received by ω_1 and ω_2 are the same, they are in a state of

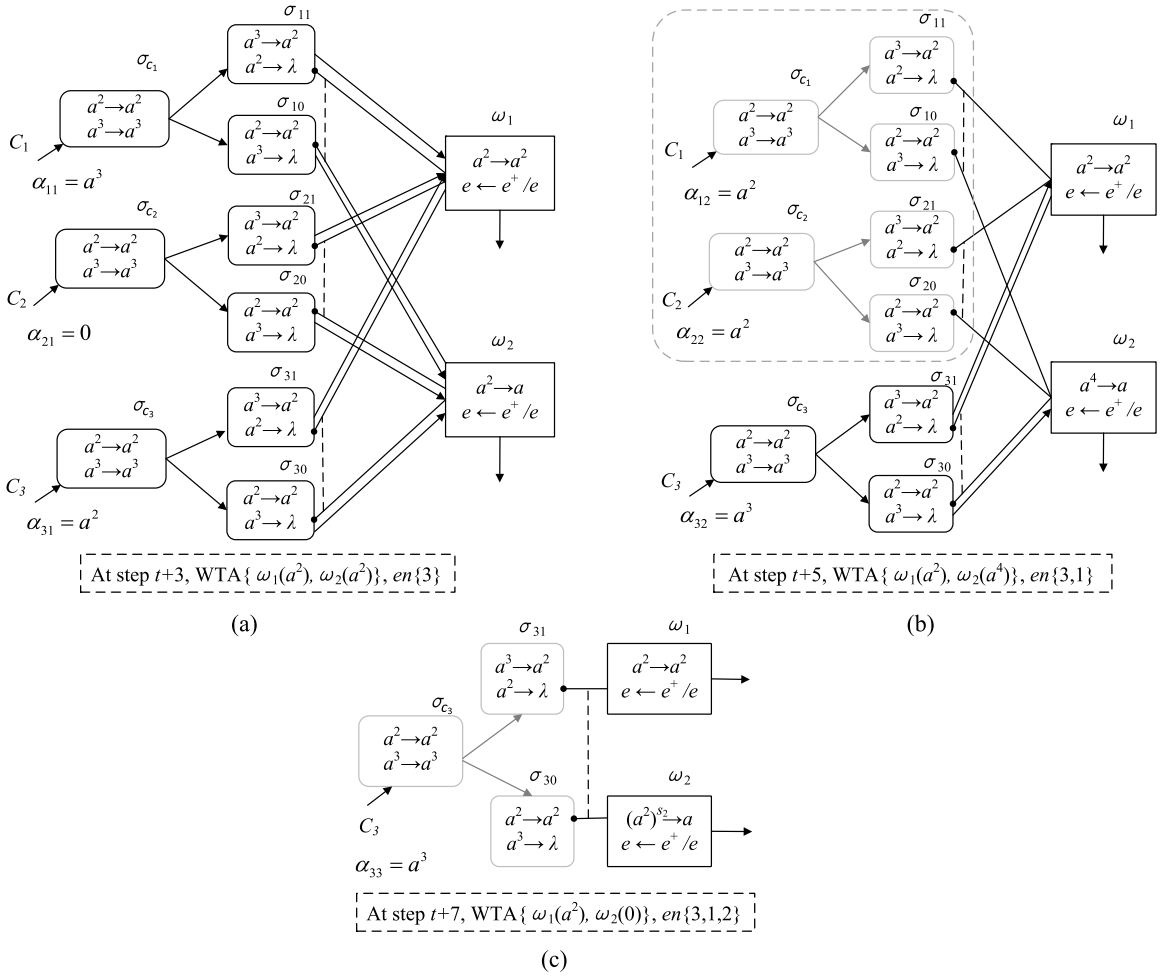


Fig. 9. The Boolean formula γ_1 is solved in Π_{sat} for the inputs of clauses C_1, C_2 and C_3. (a) Π_{sat} for x_1 truth judgment. (b) Π_{sat} for x_2 truth judgment. (c) Π_{sat} for x_3 truth judgment.

competitive equilibrium. Therefore, the spike rule is executed and they output a^2 and a spikes to the environment en respectively. At this point, the environment receives a spike sequence of $en\{3\}$, indicating that when the truth value of x_1 is taken as *true* or $false$, the number that makes C_i true is equal, and a deterministic solution cannot be obtained.

2. Competition determination of variable x_2: In step 5, ω_2 receives a^2 spikes from σ_{20} and σ_{10}, while ω_1 receives only a^2 spikes from σ_{31}. Therefore, the $WTA\{\omega_1(a^2), \omega_2(a^4)\}$ calculation shows that ω_1 is ω_{L_1} and ω_2 is ω_{W_2}. In this state, the cascade inhibitory synapses $\{\langle\omega_2,\sigma_{20}\rangle, \langle\omega_2,\sigma_{10}\rangle, \langle\omega_1,\sigma_{21}\rangle, \langle\omega_1,\sigma_{11}\rangle\}$ of ω_{W_2} are activated and receive inhibitory factor released by the ω_2 rule $e\leftarrow e^+/e$. These synaptic activations prevent ω_2 from continuing to receive spikes from σ_{20} and σ_{10}, and also prevent ω_1 from continuing to receive spikes from σ_{21} and σ_{11}. At the same time, ω_2 applies the spike rule $a^4\to a$ and outputs a spike to the environment en. At this point, the spike sequence received by the environment is updated to $en\{3,1\}$, indicating that the clauses C_1, C_2 can be made true when $x_2 = false$.
3. Competition determination for variable x_3: In step 7, ω_1 receives a^2 spikes from σ_{31}. Since ω_2 did not receive any more input spikes from the upper layer at this point, ω_1 is the only winner neuron ω_{W_1}. Then the cascade of inhibitory synapses $\{\langle\omega_2,\sigma_{30}\rangle, \langle\omega_1,\sigma_{31}\rangle\}$ of ω_{W1} is activated and receives the ω_1 released inhibitory factor, inhibitory synapses function, thereby preventing ω_1 from continuing to receive spikes from σ_{31}. In addition, ω_1 applies the spike rule $a^2\to a^2$ to output a^2 spikes to the environment en. Up to this point, the sequence of spikes received by the environment is updated to $en\{3,1,2\}$, indicating that the clause C_3 can be made true when $x_3 = true$.

At this time inhibitory synapses connected to competing neurons ω_1 and ω_2 are all excited. Therefore ω_1, ω_2 no longer receive spikes from σ_{i1}, σ_{i0}. This also means that the system finds a solution for γ_1, and the truth value of the variable x_4 being true or false does not affect γ_1 being *true*. Based on the sequence of spikes received by the environment $en\{3,1,2\}$, it can be seen that the formula $\gamma_1 = C_1\wedge C_2\wedge C_3$ is true when $x_2 = false$ and $x_3 = true$. Therefore, the solutions that satisfy $\gamma_1 = true$ can be identified. The set of solutions for which the variable x_i is assigned such that γ_1 holds is as follows: $(x_1,x_2,x_3,x_4) = \{(0,0,1,1),(1,0,1,0),(1,0,1,1),(0,0,1,0)\}$.

Table 8
The solving process for the Boolean formula $\gamma_1 = (x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee x_4)$.

Neurons	Steps: 1	2	3	4	5	6	7
σ_{c_1}	3 $a^3 \to a^3$	0 3	2 $a^2 \to a^2$	0			
σ_{11}	—	$a^3 \to a^2$ 3	0	2	×		
σ_{10}	—	$a^3 \to \lambda$	0	2	×		
σ_{c_2}	0	—	2 $a^2 \to a^2$	0			
σ_{21}	—	—	—	2	×		
σ_{20}	—	—	—	2	×		
σ_{c_3}	2 $a^2 \to a^2$	0 2	3 $a^3 \to a^3$	0 3	2 $a^2 \to a^2$	0 2	
σ_{31}	—	$a^2 \to \lambda$ 2	0	$a^3 \to a^2$ 3	0	$a^2 \to \lambda$ 2	×
σ_{30}	—	$a^2 \to a^2$	0	$a^3 \to \lambda$	0	$a^2 \to a^2$	×
ω_1	—	—	2 $a^2 \to a^2$	0	2 R_{los}	0	2 $R_{win} : a^2 \to a$
ω_2	—	—	2 $a^2 \to a^2$	0	4 $R_{win} : a^4 \to a$	0	R_{los}
en	—	—	{3}	{3}	{3,1}	{3,1}	{3,1,2}

Table 9
Comparison with other uniform solutions to SAT(n, m) problems.

SNP-based Model	Optimization method	Synaptic control	Neurons	Time steps	Rules
SNP [3]	–	Conn. & trans.	$3n^2 + 8m + 5$	$2n + 2$	4
SARPPD-SNP [67]	Refractory period and propagation delay	Conn. & trans.	–	$n + m + 3$	2
ECSNP-ER [9]	Energy req. & evo-comm.	Conn. & trans.	$n^2 + 2n + 4m + 1$	$2n + 4$	4
MSNP [8]	Microglia	Conn. & trans.	$2^n + 3n + 2$	$m + 4$	3
NGCSNP [10]	Non-gated channels	Conn. & trans.	$2^{n+1} + 3n + 2$	$m + 5$	–
WSNP [68]	Weights and thresholds	Fixed weights	–	$2n + m + 3$	2
LTPD-SNP [27]	LTPD	Weight modulation	$2^n + 3n + 3$	$m + 4$	3
DBSNP [34]	Division and budding	Established Syn.	$3(2^n + n + 1)$	$2n + mn + 6$	5
SNPSDBC [37]	Division, budding and colored spikes	Established Syn.	2^n	$O(n + m)$	–
ACSNP [36]	Cell-autonomous and synchronized	Create/delete Syn.	$2^{n+1} + 3n + 1$	$2m + 4$	4
WTASNP	WTA	Inhibition	$3m + 2$	max: $2n + 1$ min: 3	2

n and m denote the numbers of variables and clauses, respectively, where $n, m \in \mathbb{N}^*$.
Abbreviations: Conn. = connection; Trans. = transmission; Syn. = synaptic; Req. = request; Evo-comm. = evolution-communication.

The above solution set shows that the Boolean formula γ_1 always holds under these combinations of variable assignments. Note that the solution assignment for γ_1 will be different when different system input orders are given for x_i.

5.4. Comparisons

To evaluate the computational efficiency of the WTASNP systems, we conducted a comparative analysis of our approach against ten existing SNP variant methods (Table 9). These models employ diverse optimization strategies to construct variants with varying synaptic control capabilities, divisible into three groups based on synaptic functionality: 1) Synapses only facilitate connection and spike transmission. 2) Synaptic weight mechanisms are introduced beyond connection and transmission. 3) Synapses can be dynamically created or removed. Due to differing neuronal firing rules and synaptic control capabilities, the models exhibit disparate performance when solving SAT problems. Overall, all models can solve SAT problems within finite time steps, though the required neural network size often exhibits exponential growth. In contrast, WTASNP employs WTA operations to realise synaptic inhibition capabilities in competing neurons for SAT , demonstrating significantly improved computational efficiency and time complexity while exhibiting excellent solving capabilities.

1. Computational resource consumption: In solving the SAT problem, the number of neurons directly affects the computational resource consumption. A smaller number of neurons means lower storage requirements and computational burden, and higher solving efficiency. Compared with the initial SNP model and later variants, WTASNP reduces the space complexity of neuron

requirement from $O(2^n)$ or $O(n^2)$ to $O(n)$, effectively reducing computational resource consumption. The smaller number of neurons makes it easier to implement in hardware and reduces the energy consumption of the system.

2. Time efficiency: Time efficiency is an important factor in the SAT solving process, the fewer time steps means the faster the computation speed, and the more efficient to find the feasible solution. WTASNP can solve SAT in 3 steps in the optimal case, which is faster than the other methods, and it is suitable for the fast solving scenarios. Even at the maximum time overhead, the number of solution steps is lower than those listed in the literature [3,9,34], which improves the solution time efficiency. The optimization of computation time enables WTASNP to better adapt to large-scale SAT problem and meet the requirements of real engineering applications on the solution speed.

The WTASNP systems demonstrate excellent performance in both computational resource consumption and computational time efficiency. By reducing the number of neurons, optimizing the computational structure, and shortening the computation time, WTASNP systems provides an efficient method for solving SAT problem.

6. Conclusions and discussions

In this work, we employed the WTA computation mechanism inspired by the mechanism of lateral inhibition and competition between neurons in the brain, and constructed the WTASNP systems based on it. The WTASNP systems consist of two types of neurons: ordinary neurons and competing neurons. Competing neurons are selected by WTA calculation, and the neuron with the strongest expression ability is selected as the winner, which emits spikes and inhibits other neurons. The inhibitory ability of the winner neuron is reflected by the combined action of inhibitory synapses and inhibitory factors. Specifically, winner neurons release inhibitory factors, which are transmitted through inhibitory synapses to other neurons, thereby blocking the transmission of spikes from ordinary neurons along that pathway to the winner neurons. It should be noted that if the inhibitory factors propagate along the cascade inhibitory synapses, the entire cascade link becomes inhibited, preventing all ordinary neurons along that link from sending spike signals to the competing neurons.

To verify the computational power of the systems, the WTASNP systems are used as numerical generation/acceptance devices to simulate the register commands and prove their Turing completeness. Benefiting from the WTA mechanism, the WTASNP systems have good sparsity, reduces redundancies, improves information transfer efficiency, and demonstrates the potential to reduce energy consumption. Therefore, the system shows obvious advantages in solving NP-complete problems (e.g., SAT problem). Compared to existing SNP variants, WTASNP systems reduce the computational units from exponential to linear in terms of space complexity and achieves a 3-step solution in the optimal case.

We believe that the advantages of the WTASNP systems not only promote the development of the theory of SNP systems, but also provide new ideas for solving practical problems. Future research can further explore its adaptability to different types of SAT problem and optimize its ability to solve problem in specific application scenarios. Considering that the current system is limited to a single-layer competitive mechanism, we plan to extend it to a multilayer structure to further enhance the computational capability and explore the application of the mechanism to NP-complete problems such as Subset Sum or TSP.

CRediT authorship contribution statement

Tingting Bao: Writing – original draft, Methodology, Formal analysis, Conceptualization; **Bifan Wei:** Supervision, Formal analysis, Conceptualization; **Bo Li:** Validation, Software, Formal analysis; **Lingling Zhang:** Writing – review & editing, Writing – original draft, Supervision, Formal analysis, Conceptualization; **Hong Peng:** Formal analysis; **Xiaoqing Zhang:** Methodology, Conceptualization; **Jun Liu:** Writing – review & editing, Visualization, Formal analysis, Conceptualization.

Data availability

No data was used for the research described in the article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (JYB2025XDXM116), National Natural Science Foundation of China (No. 62137002, 62293553, 62293554, 62450005, 62477036, 62577043), the Fundamental Research Funds for the Central Universities (xzy022025037).

References

[1] G. Păun, Computing with membranes, J. Comput. Syst. Sci. 61 (1) (2000) 108–143.
[2] B. Song, C. Hu, D. Orellana-Martín, A. Ramírez-de Arellano, M.J. Pérez-Jiménez, X. Zeng, The computational properties of p systems with mutative membrane structures, Inf. Comput. 303 (2025) 105277.
[3] M. Ionescu, G. Păun, T. Yokomori, Spiking neural p systems, Fundam. Inform. 71 (2–3) (2006) 279–308.
[4] T. Wu, L. Valencia-Cabrera, M.J. Pérez-Jiménez, L. Pan, Spiking neural p systems with mute rules, Inf. Comput. 299 (2024) 105179.
[5] 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 Trans. Neural Netw. Learn. Syst. 34 (9) (2021) 6227–6236.
[6] B. Li, L. Zhang, T. Bao, Y. Lei, X. Zhang, J. Liu, When multi-Focus image fusion meets nonlinear spiking neural p systems, IEEE Trans. Multimed. (2025).
[7] L. Zhang, T. Wu, Homogeneous spiking neural p systems with synaptic failure, Inf. Comput. (2025) 105281.
[8] Y. Zhao, X. Liu, Spiking neural p systems with microglia, IEEE Trans. Parallel Distrib. Syst. (2024).
[9] L. Wang, X. Liu, M. Sun, Y. Zhao, Evolution-communication spiking neural p systems with energy request rules, Neural Netw. 164 (2023) 476–488.
[10] Y. Zhao, Z. Yang, Y. Luo, H. Li, W. Zang, Spiking neural p systems with non-gated channels, Inf. Comput. (2025) 105399.
[11] A.L. Yuille, N.M. Grzywacz, A winner-take-all mechanism based on presynaptic inhibition feedback, Neural Comput. 1 (3) (1989) 334–347.
[12] P. Redgrave, T.J. Prescott, K. Gurney, The basal ganglia: a vertebrate solution to the selection problem?, Neuroscience 89 (4) (1999) 1009–1023.
[13] V. Letzelter, M. Fontaine, M. Chen, P. Pérez, S. Essid, G. Richard, Resilient multiple choice learning: a learned scoring scheme with application to audio scene analysis, Adv. Neural Inf. Process. Syst. 36 (2023) 6001–6013.
[14] K.M. Cherry, L. Qian, Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks, Nature 559 (7714) (2018) 370–376.
[15] H. Peng, J. Wang, M.J. Pérez-Jiménez, A. Riscos-Núñez, Dynamic threshold neural p systems, Knowl.-Based Syst. 163 (2019) 875–884.
[16] T. Bao, N. Zhou, Z. Lv, H. Peng, J. Wang, Sequential dynamic threshold neural p systems, J. Membr. Comput. 2 (2020) 255–268.
[17] T. Wu, A. Păun, Z. Zhang, L. Pan, Spiking neural p systems with polarizations, IEEE Trans. Neural Netw. Learn. Syst. 29 (8) (2017) 3349–3360.
[18] S. Jiang, Z. Shen, B. Xu, X. Zhu, T. Liang, Spiking neural p systems with polarizations and astrocytes, J. Membr. Comput. 5 (1) (2023) 55–68.
[19] T. Bao, Q. Yang, H. Peng, X. Luo, J. Wang, X. Song, Computational power of sequential dendrite p systems, Theor. Comput. Sci. 893 (2021) 133–145.
[20] L. Garcia, G. Sanchez, J.-G. Avalos, E. Vazquez, Spiking neural p systems with myelin and dendritic spines, Neurocomputing 552 (2023) 126522.
[21] T. Bao, N. Zhou, H. Peng, Q. Yang, J. Wang, Computational completeness of sequential spiking neural p systems with inhibitory rules, Inf. Comput. 281 (2021) 104786.
[22] Y. Liu, Y. Zhao, Spiking neural p systems with lateral inhibition, Neural Netw. 167 (2023) 36–49.
[23] T. Bao, H. Peng, H. Zhou, Y. Liu, B. Zhou, Computational completeness of sequential spiking neural p systems with autapses with partial synchronization, J. Membr. Comput. (2024) 1–13.
[24] 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, Int. J. Neural Syst. 30 (10) (2020) 2050008.
[25] T. Wu, L. Pan, Q. Yu, K.C. Tan, Numerical spiking neural p systems, IEEE Trans. Neural Netw. Learn. Syst. 32 (6) (2020) 2443–2457.
[26] Y. Chen, Y. Chen, G. Zhang, T. WU, X. ZHANG, H. RONG, X. MA, et al., A survey of learning spiking neural p systems and a novel instance, Int. J. Unconv. Comput. 16 (2021).
[27] Y. Zhao, Y. Shen, X. Liu, Y. Luo, W. Zang, X. Liu, Spiking neural p systems with long-term potentiation and depression, Inf. Sci. 640 (2023) 119082.
[28] L. Wang, X. Liu, Z. Han, Y. Zhao, Spiking neural p systems with neuron permeability, Neurocomputing 576 (2024) 127351.
[29] T. Wu, L. Pan, Spiking neural p systems with communication on request and mute rules, IEEE Trans. Parallel Distrib. Syst. 34 (2) (2022) 734–745.
[30] N. Zhou, H. Peng, J. Wang, Q. Yang, X. Luo, Computational completeness of spiking neural p systems with inhibitory rules for generating string languages, Theor. Comput. Sci. 920 (2022) 64–75.
[31] P. Paul, P. Sosík, Solving the SAT problem using spiking neural p systems with coloured spikes and division rules, J. Membr. Comput. 6 (3) (2024) 222–233.
[32] X. Tian, X. Liu, Q. Ren, Y. Zhao, Spiking neural p systems with enzymes, IEEE Trans. NanoBioscience 21 (4) (2022) 575–587.
[33] A. Leporati, G. Mauri, C. Zandron, G. Păun, M.J. Pérez-Jiménez, Uniform solutions to SAT and subset sum by spiking neural p systems, Nat. Comput. 8 (4) (2009) 681–702.
[34] L. Pan, G. Păun, M.J. Pérez-Jiménez, Spiking neural p systems with neuron division and budding, Sci. China Inf. Sci. 54 (2011) 1596–1607.
[35] D. Orellana-Martín, C. Zandron, A. Leporati, A solution to SAT with virus machines with pre-computed resources, J. Membr. Comput. (2025) 1–10.
[36] D. Li, X. Liu, Y. Zhao, Autonomous spiking neural p systems with coupled neurons, Neural Netw. (2025) 108186.
[37] A. Grillo, C. Zandron, On the computational complexity of spiking neural membrane systems with colored spikes, Int. J. Neural Syst. (2025) 2550035.
[38] Z. Huang, T. Wang, W. Liu, L. Valencia-Cabrera, M.J. Pérez-Jiménez, P. Li, A fault analysis method for three-phase induction motors based on spiking neural p systems, Complexity 2021 (1) (2021) 2087027.
[39] H. Wu, T. Wang, A novel fault diagnosis method of smart grids based on artificial immune spiking neural p systems considering false data injection attacks, Electr. Power Syst. Res. 244 (2025) 111572.
[40] B. Li, H. Peng, X. Luo, J. Wang, X. Song, M.J. Pérez-Jiménez, A. Riscos-Núñez, Medical image fusion method based on coupled neural p systems in nonsubsampled shearlet transform domain, Int. J. Neural Syst. 31 (01) (2021) 2050050.
[41] B. Li, L. Zhang, J. Liu, H. Peng, Q. Wang, J. Liu, Multi-focus image fusion with parameter adaptive dual channel dynamic threshold neural p systems, Neural Netw. 179 (2024) 106603.
[42] B. Li, Y. Lei, T. Bao, Y. Wang, L. Zhang, J. Liu, Neurodynamics-Driven Coupled Neural P Systems for Multi-Focus Image Fusion, (2025). arXiv:2509.17704
[43] B. Li, L. Zhang, T. Bao, Y. Lei, X. Zhang, J. Liu, When multi-focus image fusion meets nonlinear spiking neural p systems, IEEE Trans. Multimed. (2025).
[44] J. Xue, L. Ren, B. Song, Y. Guo, J. Lu, X. Liu, G. Gong, D. Li, Hypergraph-based numerical neural-like p systems for medical image segmentation, IEEE Trans. Parallel Distrib. Syst. 34 (4) (2023) 1202–1214.
[45] I. Ermini, C. Zandron, Modular spiking neural membrane systems for image classification, Int. J. Neural Syst. 34 (06) (2024) 2450021.
[46] G. Zhang, S. Verlan, T. Wu, F.G.C. Cabarle, J. Xue, D. Orellana-Martín, J. Dong, L. Valencia-Cabrera, M.J. Pérez-Jiménez, Spiking Neural P Systems: Theory, applications and implementations, Springer Nature, 2024.
[47] L. Zhang, F. Xu, D. Xiao, J. Dong, G. Zhang, F. Neri, Enzymatic numerical spiking neural membrane systems and their application in designing membrane controllers, Int. J. Neural Syst. 32 (11) (2022) 2250055.
[48] X. Liu, H. Rong, F. Neri, Z. Yu, G. Zhang, Entropy-weighted numerical gradient optimization spiking neural system for biped robot control, Int. J. Neural Syst. 34 (06) (2024) 2450030.
[49] M.-I. Pleșa, M. Gheorghe, F. Ipate, G. Zhang, Applications of spiking neural p systems in cybersecurity, J. Membr. Comput. (2024) 1–8.
[50] J.L.I. Rangel, M.I. Arroyo, E. Vázquez, J.G. Avalos, G. Sánchez, New compact finite-Field arithmetic circuits over GF (p) based on spiking neural p systems with communication on request implemented in a low cost FPGA, IEEE Embed. Syst. Lett. 16 (2024) 295–298.
[51] J.L.G. Peña, T.E.F. Carmona, M.A.A. Rodriguez, C.A.D. Rodriguez, T.J. Borgonio, Efficient FPGA implementation of circuits based on spiking neural p systems, in: 2019IEEE International Fall Meeting on Communications and Computing (ROC&C), IEEE, 2019, pp. 51–54.
[52] Z. Shang, S. Verlan, G. Zhang, H. Rong, FPGA Implementation of numerical p systems, Int. J. Unconv. Comput. 16 (2–3) (2021) 279–302.
[53] R.V. Gungon, K.K.M. Hernandez, F.G.C. Cabarle, R.T.A. De la Cruz, H.N. Adorna, M.Á. Martínez-del Amor, D. Orellana-Martín, I. Pérez-Hurtado, GPU Implementation of evolving spiking neural p systems, Neurocomputing 503 (2022) 140–161.
[54] A.A.M. Valdez, F. Wee, A.N.L. Odasco, M.L.M. Rey, F.G.C. Cabarle, Gpu simulations of spiking neural p systems on modern web browsers, Nat. Comput. 22 (1) (2023) 171–180.
[55] G. Zhang, Z. Shang, S. Verlan, M.Á. Martínez-Del-Amor, C. Yuan, L. Valencia-Cabrera, M.J. Perez-Jimenez, An overview of hardware implementation of membrane computing models, ACM Comput. Surv. 53 (4) (2020) 1–38.

[56] L.F. Macías-Ramos, I. Pérez-Hurtado, M. García-Quismondo, L. Valencia-Cabrera, M.J. Pérez-Jiménez, A. Riscos-Núñez, A P–Lingua based simulator for spiking neural p systems, in: Membrane Computing: 12th International Conference, CMC 2011, Springer, 2012, pp. 257–281.
[57] G. Zhang, M.J. Pérez-Jiménez, A. Riscos-Núñez, S. Verlan, S. Konur, T. Hinze, M. Gheorghe, Membrane computing models: implementations, Springer, 2021.
[58] G. Ciobanu, E.N. Todoran, Spiking neural p systems and their semantics in haskell, Nat. Comput. 22 (1) (2023) 41–54.
[59] B. Aman, G. Ciobanu, Arithmetic abilities of SNP systems with astrocytes producing calcium, Neural Netw. 183 (2025) 106913.
[60] G. Shin, S. Albanie, W. Xie, Unsupervised salient object detection with spectral cluster voting, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 3971–3980.
[61] D. Wang, R. Tang, H. Lin, L. Liu, N. Xu, Y. Sun, X. Zhao, Z. Wang, D. Wang, Z. Mai, et al., Spintronic leaky-integrate-fire spiking neurons with self-reset and winner-takes-all for neuromorphic computing, Nat. Commun. 14 (1) (2023) 1068.
[62] G. Goupy, P. Tirilly, I.M. Bilasco, Neuronal competition groups with supervised STDP for spike-Based classification, in: NeurIPS Neural Information Processing Systems, 2024.
[63] Z. Yang, S. Guo, Y. Fang, Z. Yu, J.K. Liu, Spiking variational policy gradient for brain inspired reinforcement learning, IEEE Trans. Pattern Anal. Mach. Intell. 47 (03) (2025) 1975–1990.
[64] K. Liu, Y. Zhang, Distributed dynamic task allocation for moving target tracking of networked mobile robots using k-WTA network, IEEE Trans. Neural Netw. Learn. Syst. (2024).
[65] I. Korec, Small universal register machines, Theor. Comput. Sci. 168 (2) (1996) 267–301.
[66] M.R. Garey, D.S. Johnson, Computers and intractability: a guide to the theory of NP-Completeness (1990).
[67] Y. Zhao, Y. Liu, X. Liu, M. Sun, F. Qi, Y. Zheng, Self-adapting spiking neural p systems with refractory period and propagation delay, Inf. Sci. 589 (2022) 80–93.
[68] J. Wang, H.J. Hoogeboom, L. Pan, G. Păun, M.J. Pérez-Jiménez, Spiking neural p systems with weights, Neural Comput. 22 (10) (2010) 2615–2646.