1

Introduction

1.1 Background

Natural computing is a computing paradigm abstracted from nature, aimed at imitating the inherent phenomena and laws of nature or human society, exploring new algorithms for problems, and better solving them [1]. Natural computing typically encompasses adaptive computational frameworks capable of autonomous organization and continuous improvement, addressing intricate challenges beyond conventional computing paradigms. So far, natural computing has proposed numerous algorithms corresponding to various origins of inspiration, such as swarm intelligence algorithms (biological populations), artificial life (biological individuals), immune algorithms (immune systems), artificial neural networks (biological nervous systems), membrane computing (living cells), and DNA computing (DNA molecules), as well as some methods derived from physical and chemical phenomena.

Proposed by Gh. Păun in 1998, membrane computing represents an innovative computational framework within the broader field of natural computing [2]. Membrane computing aims to abstract computing models from the structure and function of living cells as well as tissues and organs that collaborate to process information. Membrane systems, alternatively called P systems, constitute a class of inherently parallel and spatially distributed computational models in membrane computing. In P systems, the interior of the cell is divided into several regions separated by the cell membranes, which are considered independent computing units. Each region contains multisets of objects (corresponding to compounds that are constantly evolving inside the cell) as carriers of information, and each computational module's handling of object multisets (analogous to cellular biochemical reactions) constitutes its autonomous information processing mechanism. Intercellular object transfer across membranes embodies the computational units' information-sharing mechanism. With distributed processing

Spiking Neural P Systems for Time Series Analysis, First Edition. Jun Wang and Hong Peng.

elements working in parallel, P systems achieve significantly greater theoretical efficiency than von Neumann-type computers [3]. Their computational power has been mathematically proven equivalent to Turing machines, with indications of transcending these classical constraints.

Drawing upon the electrophysiological properties of neural signal transmission, Ref. [4] introduced spiking neural P systems (SNP systems) in 2006 – a novel membrane computing framework mimicking neuronal architectures. The structural foundation of SNP systems consists of directed graphs wherein neuronal units serve as computational nodes interconnected by oriented synaptic pathways, facilitating concurrent distributed processing. For each neuron, there exists a state variable representing the number of spikes within it (simulating the membrane potential of biological neuron). Moreover, a set of spiking/firing rules is defined to update the state of the neuron (simulating the growth or decay of membrane potential). Due to common features with spiking neural networks [5], SNP systems are classified among third-generation neural network architectures. These research results have demonstrated that SNP systems can achieve Turing-complete computational power in many cases and have also demonstrated excellent performance and potential in solving computationally difficult problems.

Within the domain of biologically-inspired computation, membrane computing represents a significant advancement, offering fresh modeling techniques and computational methodologies for computing research. Concurrently, it furnishes innovative theoretical frameworks, methodological advancements, and practical instruments to address real-world challenges.

1.2 Membrane Computing

During his 1998 academic stay in Finland, Gh. Păun initially formulated the principles of membrane computing, which were subsequently documented in his pioneering 2000 publication [2]. Numerous researchers have focused their attention on membrane computing because of its extensive potential for real-world applications in diverse fields such as computer science, systems biology, and linguistics. The Institute for Scientific Information (ISI) recognized membrane computing in 2003 as an emerging and fast-growing discipline within computer science. The pioneering membrane computing paper emerged as both an immediate scientific breakthrough and one of ISI's most frequently referenced publications during its publication year. Springer introduced the field's first comprehensive publication on membrane computing during 2002, titled "*Membrane Computing – An Introduction*" [6]. Oxford University Press published "*The Oxford Handbook of Membrane Computing*" in 2010 [3]. In 2017, Springer published the book "*Real Life Application with Membrane Computing*" [7]. In 2024, Springer published the book "*Advanced Spiking Neural P Systems: Models and*

Applications" [8]. The membrane computing domain has yielded substantial academic output to date, including 40+ journal special issues, 50+ conference proceedings volumes, 70+ doctoral dissertations, and numerous review articles.

In membrane computing community, theoretical research, application research, and software and hardware implementation are the three main contents. Below, we will briefly review their research status.

1.2.1 Theoretical Research

With over two decades of continuous advancement, membrane computing has established a substantial theoretical foundation. P systems exhibit three primary architectural variants based on their topological configurations: cell-like (hierarchical organization), tissue-like (networked arrangement), and neural-like (directed graph architecture). The biological diversity observed in living cells has led to the creation of numerous P system architectures, exemplified by catalytic variants, membrane transport systems utilizing symport/antiport mechanisms, and promoter/inhibitor-regulated models [9, 10]. Drawing on fundamental concepts in computer science, P systems exhibiting varied operational mechanisms and parallel processing capabilities have been constructed [11]. By simulating with computing devices in classical computational theory, including Turing machines, registration machines, etc., it was confirmed that most P systems are Turing complete, meaning that the computing power of these systems is equivalent to that of Turing machines. By introducing mechanisms such as cell division, cell separation, and cell generation, P systems can generate their own computational space during the computation process. This allows these systems to effectively solve some classic NP-problems, such as PSPACE, knapsack, and SAT problems, by exchanging space for time [12, 13].

1.2.2 Application Research

Due to some characteristics of membrane computing, such as distribution, uncertainty, and scalability, the computational framework of P systems demonstrates particular efficacy in resolving specific real-world problems. In the early stages of applications, the focus was mainly on the research of membrane algorithms. The fusion between the naturally parallelized framework of P systems and established optimization approaches (genetic algorithm [GA], particle swarm optimization [PSO], differential evolution [DE]) gave rise to an innovative algorithmic paradigm. The membrane algorithm was initially developed by Ref. [14] as a computational approach for combinatorial optimization challenges, with its practical implementation demonstrated through successful application to the traveling salesman problem. The integration of membrane computing with established

optimization techniques has spawned multiple algorithmic derivatives, demonstrating effectiveness across diverse problem domains ranging from theoretical models to real-world applications. Beyond its core applications, membrane computing demonstrates significant utility across interdisciplinary domains including ecological system simulations [15], visual data analysis [16], and intelligent algorithm design [17].

In recent years, membrane computing learning models have received attention, focusing on how to construct learning models using the mechanisms of membrane computing. The unsupervised learning model for membrane computing focuses on clustering algorithms. By utilizing the structures of different P systems and the mechanisms of cells (or membranes), some membrane inspired clustering algorithms have been proposed [18–22], known as membrane clustering algorithms. These algorithms discussed problems such as data clustering, fuzzy clustering, multi-objective clustering, and automatic clustering.

1.2.3 Software and Hardware Implementation

With the deepening of research, the software and hardware implementation of membrane computing has made certain progress. Several simulation software have been developed for various membrane systems, including Membrane Simulator based on C++ for catalytic P systems and active membrane P systems [23], and SNUPS based on Java for numerical P systems [24]. However, current P system simulation tools predominantly exhibit limited functionality, typically restricted to emulating specific membrane system configurations. A research team specializing in natural computation at Spain's Seville University developed P-Lingua, an integrative modeling framework [25]. The framework enables comprehensive P system modeling and execution through Java-based pLinguaCore kernel invocations. For hardware-level realization, the team employed CUDA parallel computing architecture to enable GPU-accelerated simulation of membrane computing systems [26] and proved that GPU parallel computing can greatly improve computational efficiency by solving difficult problems on the P system's GPU simulator.

1.3 Spiking Neural P Systems

SNP systems are computing systems derived from the way neurons in biological neural networks process information in the form of spikes and interact with each other using pulses, and were proposed by Ref. [4]. Neural computing is a hot research field at the intersection of neuroscience and computational science, and as a new type of third-generation neural computing models, SNP systems have become a research hotspot since their proposal, achieving numerous

research results. Simultaneously, the neural computing research community has disseminated significant findings through high-impact journal publications, such as *Neural Networks and Neurocomputing* under Elsevier, *International Journal of Neural Systems* under World Scientific, and *IEEE Transactions on Neural Networks and Learning Systems*.

SNP systems represent a category of bio-inspired computing architectures that exhibit both parallel processing and distributed characteristics within membrane computing paradigms [8, 27]. SNP systems fundamentally comprise three key components: a directed graph architecture, neuronal units containing discrete spike counts, and operational rules governing both spike consumption and generation processes. The operational rules fundamentally determine the spiking dynamics that define SNP systems' computational behavior.

In the past decade, the investigation of SNP systems has yielded substantial findings across fundamental theory, practical applications, and software and hardware aspects. Below, we will introduce each of these three aspects separately.

1.3.1 Theoretical Research Work

The theoretical work of SNP systems mainly involves establishing various computational models and analyzing their computational capabilities, including two key aspects: Turing-equivalent computational capability assessment and the feasibility of deriving efficient algorithms for NP-hard problems.

Since Ref. [4] proposed SNP systems, numerous adaptations have been derived from diverse biological discoveries in the past decade, particularly those observed in neuroscience. Here are some discoveries listed: multiple channels [19], anti-spikes [28], astrocytes [29], polarization [30], rules on synapses [31], structural plasticity [32], communications on requests [33], inhibitory rules [34], autapses [35], dendritic mechanism [34], delay on synapses [36], nonlinear mechanism [37], coupling modulation [38], dynamic threshold [39], plasmids [40], etc. Recent studies on SNP systems reveal an increasing number of biologically inspired mechanisms.

The first theoretical question to address after building the model is its computational completeness, specifically examining if it achieves the same level of computation as a Turing machine. When examining computational completeness, computational models to analyze encompass digital generators/acceptors, functional computing devices, and formal language generators. Many results demonstrate that these models exhibit Turing-complete capabilities, such as digital generators/acceptors and functional computing devices [3]. In addition, it has been proven that some variants can generate regular languages and recursively enumeration languages [41].

SNP models generally operate under three distinct execution modes: sequential, asynchronous, and synchronous. For synchronous case, a global clock synchronizes each neuron. Therefore, neurons operate concurrently, however, every neuron uses the rules in sequence. In previous variants, most work in synchronous case. For asynchronous case, all neurons work asynchronously without considering the global clock [42]. The past few years have seen theoretical verification of computation universality in some asynchronous models. For sequential case, processing scheme requires neurons to activate one after another, while maintaining rule execution order within individual processing units [43]. Researchers have introduced various sequential models that demonstrate computational universality, serving as both digital generators/acceptors and functional computing devices.

Computational theorists continue developing scaled-down universal systems, exemplified by space-efficient Turing machines and register machines with minimal instruction sets. The investigation of minimal universal models has become an active research direction in SNP systems, with Gh. Păun pioneering the foundational work on their computational completeness at small scales [29].

Another key theoretical focus involves analyzing the computational efficiency of SNP system architectures. This study aims to evaluate computational efficiency through the application of SNP architectures and their modified forms in addressing some NP-hard problems. The purpose is to test their efficiency by using SNP systems and variants to solve some NP-hard problems, including PSPACE and SAT problems [44]. SNP systems show great potential in solving these NP-hard problems. Currently, multiple approaches exist for addressing this challenge: (i) Using exponential scale precomputing resources (i.e., the system has exponential neurons) to solve the NP-hard problem through spatial exchange time; (ii) The polynomial scale SNP systems can solve NP-hard problems in a nondeterministic manner in polynomial time; (iii) Enhancing neuronal density in computational architectures facilitates a space-for-time exchange strategy, which proves effective against NP-complete challenges.

1.3.2 Model Application Work

With their distributed framework, massive parallel processing capacity, and tolerance for ambiguous dynamics, SNP systems demonstrate unique advantages in handling nonlinear real-life challenges that require flexible computational approaches.

The computational properties of SNP systems have enabled the development of diverse hardware components [45], including elementary logic elements, digital circuit implementations, natural number arithmetic units, and probabilistic computing modules.

Researchers have employed the nondeterministic nature of SNP systems to create enhanced algorithmic solutions [46], where refined SNP architectures address complex combinatorial optimization challenges.

The incorporation of fuzzy set theory into SNP architectures has led to the creation of new computational models with enhanced uncertainty handling capabilities. Subsequently, these enhanced SNP models with fuzzy capabilities found practical implementation in diagnosing faults within electrical transformers and broader power grid systems [47, 48].

Nonlinear spiking neural P (NSNP) systems, coupled neural P (CNP) systems, and dynamic threshold neural P (DTNP) systems were applied in multiple image processing tasks, including image fusion [49], image segmentation [50], and edge detection [51, 52].

The nonlinear spiking mechanism of NSNP systems has been used to develop several recurrent-like models and convolutional models, including long- and short-term memory-like spiking neural P systems (or LSTM-SNP model) [53], nonlinear spiking neural P systems with autapses (or NSNP-AU model) [54], gated spiking neural P systems (or GSNP model) [55], and ConvSNP model [56]. These models were applied in time series [8, 57], image processing [52, 58], sentiment analysis [59, 60], and recommendation systems [61, 62].

1.3.3 Hardware and Software Simulation

Significant advancements have been achieved in creating computational tools for simulating SNP systems. As an illustration, Ref. [63] developed computational models within the P-Lingua environment to simulate both conventional and nonsynchronous SNP systems. Ref. [64] created Pnet Lab, a computational platform designed for the analysis and emulation of both standard SNP systems and their anti-spike variants. Furthermore, Ref. [65] engineered an online interactive platform capable of graphically simulating SNP systems.

From a computational hardware perspective, studies have examined how matrix formulations can represent spiking neural membrane computing models [66]. Building upon the matrix model, developers engineered a high-performance simulation platform harnessing GPU parallel processing through CUDA programming. The massively parallel design of GPUs aligns well with the concurrent processing nature of SNP systems, enabling significant enhancements in simulation speed through hardware acceleration.

1.4 Time Series Analysis

Today, we have entered the digital age and data is growing explosively. As a collection of data arranged in chronological order, time series data is widely present in various fields, such as stock price trends in financial markets, temperature

changes in meteorological fields, and equipment operating parameters in industrial production. Conducting in-depth analysis of time series data and uncovering the underlying patterns and trends is of crucial importance for decision-making, risk prediction, and other related areas [67].

As a multidisciplinary field, machine learning aims to enable computers to learn features and patterns through data, thereby achieving tasks such as predicting and classifying unknown data. As a subset of machine learning, deep learning employs multi-layered neural architectures to autonomously extract intricate patterns from extensive datasets. Advancements in computational capabilities have propelled machine and deep learning to the forefront of temporal data studies, where their efficacy continues to attract significant academic attention.

Classical approaches, autoregressive moving average (ARMA) and exponential smoothing, perform well in handling simple linear time series data, but often struggle to achieve ideal results for time series data with complex nonlinear relationships and long-term dependencies. Machine learning and deep learning models, with their powerful nonlinear modeling and automatic feature extraction capabilities, can effectively capture complex dynamic features and patterns in time series data, bringing a new solution for time series analysis.

In the financial field, forecasting accurate time series can help investors grasp market trends, formulate reasonable investment strategies, reduce risks, and increase returns. For example, by analyzing and predicting the time series of stock prices, investors can decide when to buy or sell stocks. In industrial production, time series classification can be used for equipment fault diagnosis and quality control. By analyzing the time series data of equipment operation status, the normal operation status and fault status are classified, and potential fault hazards are discovered in a timely manner to ensure the continuity and stability of production. Anomaly detection can identify abnormal data points in the production process, take timely measures to adjust, and avoid production accidents. In the field of transportation, by analyzing and predicting the time series of traffic flow, it is possible to plan traffic resources reasonably, optimize traffic signal timing, and alleviate traffic congestion.

1.4.1 Prediction Problems

Forecasting future data points by analyzing historical temporal sequences is a widely utilized technique across multiple disciplines, such as finance, economics, energy, meteorology, and more [68]. Accurate time series forecasting can help companies develop reasonable production plans, investors make wise investment decisions, and governments carry out effective policy planning. With the continuous development of techniques, many methods for time series forecasting have

emerged, which can be roughly divided into machine learning-based and deep learning-based approaches [69].

Classical prediction approaches mainly include autoregressive moving average (ARIMA) and their extensions, such as seasonal autoregressive moving average (SARIMA) model, and exponential smoothing methods. However, the ARIMA model also has some limitations. It requires high stationarity of data, and for non-stationary time series data, complex differencing is required. Improper selection of different orders may affect the performance of the model. The ARIMA model assumes that time series data have a linear relationship, and for time series with complex nonlinear relationships, its prediction performance is often not ideal.

As deep neural architectures continue to evolve, their utilization for forecasting sequential data patterns has expanded significantly in practical implementations. Deep learning models have powerful nonlinear modeling and feature extraction capabilities, which can effectively process complex time series data, capturing long-term dependency, and sequential pattern.

Common deep learning models used for time series prediction include (i) gated recurrent unit (GRU) and long short-term memory network (LSTM); (ii) convolutional neural network (CNN); (iii) transformer model. Deep learning models such as LSTM, GRU, transformer, and their variants typically exhibit better performance and can more accurately obtain complex patterns and trends in data [69, 70]. CNN related models have advantages in capturing local features of time series and are suitable for processing data with obvious local patterns and trends.

1.4.2 Classification Problems

The classification task is to assign time series data to predefined categories and has wide applications in fields such as industrial manufacturing, medical diagnosis, and financial risk assessment [71]. For example, in industrial manufacturing, by classifying the time series data of equipment operation, it is possible to determine whether the equipment is in a normal operating state. In medical diagnosis, based on the patient's physiological indicators and time series data, it can assist doctors in diagnosing and classifying diseases.

Machine learning has also been extensively applied in time series classification. Traditional machine learning models include support vector machine (SVM) [72] and random forest (RF) [73].

Deep learning methods can automatically capture complex features in time series, avoiding the tedious process of manual feature extraction, and have advantages in processing large-scale data [71]. These deep learning methods include (i) RNN, LSTM, and GRU models; (ii) CNN-based models; (iii) transformer-based models.

1.4.3 Anomaly Detection Problems

The anomaly detection for time series is to identify data samples or subsequences in time series data that do not conform to normal patterns [74]. It has important application value in financial risk warning, industrial equipment fault diagnosis, network security monitoring, and other fields. For example, in the financial sector, timely detection of abnormal fluctuations in stock prices can help investors avoid significant losses. In industrial production, detecting abnormal changes in equipment operating parameters can prevent equipment failures in advance, ensuring the continuity and safety of production. With the development of machine learning and deep learning technologies, time series anomaly detection methods are constantly innovating and improving.

Statistical methods are commonly used traditional methods in time series anomaly detection, mainly based on the statistical features of data to identify outliers. Moving average is a simple and commonly used method that identifies outliers by calculating the average around a data point.

Machine learning models were widely applied in time series anomaly detection, capable of handling more complex data patterns and relationships, including isolation forests (IF) [75] and one class SVM [76].

Deep learning methods have demonstrated powerful capabilities in time series anomaly detection [77], capable of automatically learning complex features and patterns of data. These deep learning methods include AutoEncoder, LSTM, etc. They demonstrate powerful capabilities in fields such as power system fault detection and aircraft engine health monitoring by learning patterns and patterns from normal data, utilizing reconstruction errors or prediction biases to detect anomalies.

1.5 Organization of Chapters

Organized in three parts, this book comprises eight chapters in total. As a foundation, the first part establishes the core concepts, including Chapters 1 and 2. Serving as the introductory section, Chapter 1 initially explores the scholarly context and membrane computation before delving into the developmental milestones of spiking neural P systems and time series analysis. Chapter 2 describes spiking neural P systems, and then discusses three theoretical variants, including spiking neural P systems with extended rules, spiking neural P systems with autapses, and nonlinear spiking neural P systems.

The second part introduces three classes of application models of time series analysis, including three chapters. Chapter 3 discusses the recurrent-like models for time series prediction. Chapter 4 discusses the echo-like models for time series prediction. Chapter 5 discusses the reservoir computing models for time series classification.

Comprising three chapters, the final part focuses on practical applications. Chapter 6 introduces the applications in financial time series analysis. Chapter 7 introduces the applications in power load forecasting and photovoltaic power forecasting tasks. Focusing on medical signal processing problem, Chapter 8 presents corresponding method.

References

1 D.E. Shasha and C. Lazere. *Natural Computing*. W. W. Norton & Company, 2010.
2 Gh. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1): 108–143, 2000.
3 Gh. Păun, G. Rozenberg, and A. Salomaa. *The Oxford Handbook of Membrane Computing*. Oxford University Press, New York, 2010.
4 M. Ionescu, Gh. Păun, and T. Yokomori. Spiking neural P systems. *Fundamenta Informaticae*, 71: 279–308, 2006.
5 S. Ghosh-Dastidar and H. Adeli. Spiking neural networks. *International Journal of Neural Systems*, 19(4): 295–308, 2009.
6 Gh. Păun. *Membrane Computing: an Introduction*. Springer-Verlag, Berlin, Heidelberg, 2002.
7 G. Zhang, M.J. Pérez-Jiménez, and M. Gheorghe. *Real-Life Application with Membrane Computing*. Springer, 2017.
8 H. Peng and J. Wang. *Advanced Spiking Neural P Systems: Models and Applications*. Springer, Singapore, 2024.
9 A. Păun and Gh. Păun. The power of communication: P systems with symport/antiport. *New Generation Computing*, 20(3): 295–305, 2002.
10 F. Bernardini and M. Gheorghe. Languages generated by P systems with active membranes. *New Generation Computing*, 22(4): 311–329, 2004.
11 R. Freund. Asynchronous P systems and P systems working in the sequential mode. *Lecture Notes in Computer Science*, 3365: 36–62, 2004.
12 P. Sośık, A. Păun, and A. Rodríguez-Patón. P systems with proteins on membranes characterize PSPACE. *Theoretical Computer Science*, 488: 78–95, 2013.
13 B. Song, M.J. Pérez-Jiménez, and L. Pan. Efficient solutions to hard computational problems by P systems with symport/antiport rules and membrane division. *BioSystems*, 130: 51–58, 2015.
14 T.Y. Nishida. An application of P systems: a new algorithm for NP-complete optimization problems. *Proceedings of the 8th World Multi-Conference on Systems, Cybernetics and Informatics*, pages 109–112, 2004.
15 M. García-Quismondo, M. Levin, and D. Lobo. Modeling regenerative processes with membrane computing. *Information Sciences*, 381: 229–249, 2017.

16 D. Díaz-Pernil, M.A. Gutiérrez-Naranjo, and H. Peng. Membrane computing and image processing: a short survey. *Journal of Membrane Computing*, 1: 58–73, 2019.

17 C. Buiu and A.G. Florea. Membrane computing models and robot controller design, current results and challenges. *Journal of Membrane Computing*, 1(4): 262–269, 2019.

18 J. Yang, H. Peng, X. Luo, and J. Wang. Stochastic numerical P systems with application in data clustering problems. *IEEE Access*, 8(1): 31507–31518, 2020.

19 H. Peng, P. Shi, J. Wang, A. Riscos-Núñez, and M.J. Pérez-Jiménez. Multiobjective fuzzy clustering approach based on tissue-like membrane systems. *Knowledge-Based Systems*, 125: 74–82, 2017.

20 H. Peng, J. Wang, P. Shi, M.J. Pérez-Jiménez, and A. Riscos-Núñez. An extended membrane system with active membrane to solve automatic fuzzy clustering problems. *International Journal of Neural Systems*, 26(3): 1650004, 2016.

21 H. Peng, J. Wang, P. Shi, A. Riscos-Núñez, and M.J. Pérez-Jiménez. An automatic clustering algorithm inspired by membrane computing. *Pattern Recognition Letters*, 68: 34–40, 2015.

22 H. Peng, J. Wang, M.J. Pérez-Jiménez, and A. Riscos-Núñez. An unsupervised learning algorithm for membrane computing. *Information Sciences*, 304: 80–91, 2015.

23 G. Ciobanu and D. Paraschiv. P system software simulator. *Fundamenta Informaticae*, 49(1–3): 61–66, 2002.

24 O. Arsene, C. Buiu, and N. Popescu. SNUPS–a simulator for numerical membrane computing. *International Journal of Innovative Computing, Information and Control*, 7(6): 3509–3522, 2011.

25 D. Díaz-Pernil, I. Pérez-Hurtado, M.J. Pérez-Jiménez, and A. Riscos-Núñez. A P-lingua programming environment for membrane computing. *Lecture Notes in Computer Science*, 5391: 187–203, 2008.

26 J.M. Cecilia, García J.M., G.D. Guerrero, M.A. Martínez-del-Amor, P.H. Ignacio, and M.J. Pérez-Jiménez. Simulation of P systems with active membranes on CUDA. *Briefings in Bioinformatics*, 11(3): 313–322, 2009.

27 H. Chen, M. Ionescu, T.O. Ishdorj, A. Păun, Gh. Păun, and MJ. Pérez-Jiménez. Spiking neural P systems with extended rules: universality and languages. *Natural Computing*, 7: 147–166, 2008.

28 L. Pan and Gh. Păun. Spiking neural P systems with anti-spikes. *International Journal of Computers, Communications & Control*, 4(3): 273–282, 2009.

29 Gh. Păun. Spiking neural P systems with astrocyte-like control. *Journal of Universal Computer Science*, 13(11): 1707–1721, 2007.

30 T. Wu, A. Păun, Z. Zhang, and L. Pan. Spiking neural P systems with polarizations. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8): 3349–3360, 2017.

31 T. Song, L. Pan, and Gh. Păun. Spiking neural P systems with rules on synapses. *Theoretical Computer Science*, 529: 82–95, 2014.

32 R.T.A. de la Cruz, F.G.C. Cabarle, I.C.H. Macababayao, H.N. Adorna, and X. Zeng. Homogeneous spiking neural P systems with structural plasticity. *Journal of Membrane Computing*, 3(1): 10–21, 2021.

33 L. Pan, Gh. Păun, G. Zhang, and F. Neri. Spiking neural P systems with communication on request. *International Journal of Neural Systems*, 28(8): 1750042, 2017.

34 H. Peng, B. Li, J. Wang, X. Song, T. Wang, L. Valencia-Cabrera, I. Pérez-Hurtado, A. Riscos-Núñez, and M.J. Pérez-Jiménez. Spiking neural P systems with inhibitory rules. *Knowledge-Based Systems*, 30(2): 2050008, 2020.

35 X. Song, L. Valencia-Cabrera, H. Peng, and J. Wang. Spiking neural P systems with autapses. *Information Sciences*, 570: 383–402, 2021.

36 X. Song, L. Valencia-Cabrera, H. Peng, J. Wang, and M.J. Pérez-Jiménez. Spiking neural P systems with delay on synapses. *International Journal of Neural Systems*, 31(1): 2050042, 2021.

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

38 H. Peng and J. Wang. Coupled neural P systems. *IEEE Transactions on Neural Networks and Learning Systems*, 30(6): 1672–1682, 2019.

39 H. Peng, J. Wang, M.J. Pérenz-Jiménez, and A. Riscos-Núñez. Dynamic threshold neural P systems. *Knowledge-Based Systems*, 163: 875–884, 2019.

40 F.G.C. Cabarle, X. Zeng, N. Murph, T. Son, A. Rodríguez-Patón, and X. Liu. Neural-like P systems with plasmids. *Information and Computation*, 281(4): 104766, 2021.

41 Y. Huang, W. Yi, H. Peng, J. Wang, X. Luo, and Q. Yang. Computational power of dynamic threshold neural P systems for generating string languages. *Theoretical Computer Science*, 851: 77–91, 2021.

42 X. Song, J. Wang, H. Peng, G. Ning, Z. Sun, T. Wang, and F. Yang. Small universal asynchronous spiking neural P systems with multiple channels. *Neurocomputing*, 378: 1–8, 2020.

43 T. Bao, N. Zhou, H. Peng, Q. Yang, and J. Wang. Computational completeness of sequential spiking neural P systems with inhibitory rules. *Information and Computation*, 281(1): 104786, 2021.

44 L. Pan, Gh. Păun, and MJ. Pérez-Jiménez. Spiking neural P systems with neuron division and budding. *Science China Information Sciences*, 54(8): 1596–1607, 2011.

45 T. Song, P. Zheng, M.D. Wong, and X. Wang. Design of logic gates using spiking neural P systems with homogeneous neurons and astrocytes-like control. *Information Sciences*, 372: 380–391, 2016.

46 M. Zhu, Q. Yang, J. Dong, G. Zhang, X. Gou, H. Rong, P. Paul, and F. Neri. An adaptive optimization spiking neural P systems for binary problems. *International Journal of Neural Systems*, 31(1): 1440006, 2021.

47 H. Peng, J. Wang, J. Ming, P. Shi, M.J. Pérez-Jiménez, W. Yu, and C. Tao. Fault diagnosis of power systems using intuitionistic fuzzy spiking neural P systems. *IEEE Transaction on Smart Grid*, 9(5): 4777–4784, 2018.

48 J. Wang, H. Peng, W. Yu, M. Ming, M.J. Pérez-Jiménez, C. Tao, and X. Huang. Interval valued fuzzy spiking neural P systems for fault diagnosis of power transmission networks. *Engineering Applications of Artificial Intelligence*, 82: 102–109, 2019.

49 B. Li, H. Peng, J. Wang, and X. Huang. Multi-focus image fusion based on dynamic threshold neural P systems and surfacelet transform. *Knowledge-Based Systems*, 196: 105794, 2020.

50 Y. Cai, S. Mi, J. Yan, H. Peng, X. Luo, Q. Yang, and J. Wang. An unsupervised segmentation method based on dynamic threshold neural P systems for color images. *Information Sciences*, 587: 473–484, 2022.

51 R. Xian, R. Lugu, H. Peng, Q. Yang, X. Luo, and J. Wang. Edge detection method based on nonlinear spiking neural systems. *International Journal of Neural Systems*, 33(1): 2250060, 2023.

52 R. Xian, X. Xiong, H. Peng, J. Wang, A. Ramírez-de-Arellano, and Q. Yang. Feature fusion method based on spiking neural convolutional network for edge detection. *Pattern Recognition*, 147: 110112, 2024.

53 Q. Liu, L. Long, Q. Yang, H. Peng, J. Wang, and X. Luo. LSTM-SNP: A long short-term memory model inspired from spiking neural P systems. *Knowledge-Based Systems*, 235: 107656, 2022.

54 Q. Liu, H. Peng, L. Long, J. Wang, Q. Yang, M.J. Pérez-Jiménez, and D. Orellana-Martín. Nonlinear spiking neural systems with autapses for predicting chaotic time series. *IEEE Transactions on Cybernetics*, 54(3): 1841–1853, 2024.

55 Q. Liu, L. Long, H. Peng, J. Wang, Q. Yang, X. Song, A. Riscos-Núñez, and M.J. Pérez-Jiménez. Gated spiking neural P systems for time series forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, 34(9): 6227–6236, 2023.

56 S. Zhao, L. Zhang, H. Peng, Z. Liu, and J. Wang. ConvSNP: a deep learning model embedded with SNP-like neurons. *Journal of Membrane Computing*, 4: 87–95, 2022.

57 Y. Zhang, Q. Yang, Z. Liu, H. Peng, and J. Wang. A prediction model based on gated nonlinear spiking neural system. *International Journal of Neural Systems*, 33(6): 2350029, 2023.

58 L. Ye, C. Zhou, H. Peng, J. Wang, Z. Liu, and Q. Yang. Multi-level feature interaction image super-resolution network based on convolutional nonlinear spiking neural model. *Neural Networks*, 177: 106366, 2024.

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

60 Y. Huang, H. Peng, Q. Liu, Q. Yang, J. Wang, D. Orellana-Martín, and M.J. Pérez-Jiménez. Attention-enabled gated spiking neural P model for aspect-level sentiment classification. *Neural Networks*, 157: 437–443, 2023.

61 X. Bai, Y. Huang, H. Peng, J. Wang, Q. Yang, D. Orellana-Martín, A. Ramírez-de-Arellano, and M.J. Pérez-Jiménez. Sequence recommendation using multi-level self-attention network with gated spiking neural P systems. *Information Sciences*, 656: 119916, 2024.

62 X. Bai, L. Zhang, M. Jiang, H. Peng, J. Wang, Q. Yang, and A. Ramírez-de-Arellano. Gated graph spiking neural P network for session-based recommendation. *Knowledge-Based Systems*, 300: 112162, 2024.

63 L.F. Macías-Ramos, M.J. Pérez-Jiménez, T. Song, and L. Pan. Extending simulation of asynchronous spiking neural P systems in P-Lingua. *Fundamenta Informaticae*, 36(3): 253–267, 2015.

64 V.P. Metta, K. Krithivasan, and D. Garg. Extending simulation of asynchronous spiking neural P systems in P-Lingua. *Proceedings of the 12th International Conference on Membrane Computing*, pages 381–394, 2011.

65 A.G.S. Dupaya, A.C.A.P. Galano, F.G.C. Cabarle, R.T.D. La Cruz, K.J. Ballesteros, and P.P. Lazo. A web-based visual simulator for spiking neural P systems. *Journal of Membrane Computing*, 4: 21–40, 2022.

66 Z.B. Jimenez, F.G.C. Cabarle, R.T.A. de la Cru, K.C. Buño, H.N. Adorna, N.H.S. Hernandez, and X. Zeng. Matrix representation and simulation algorithm of spiking neural P systems with structural plasticity. *Journal of Membrane Computing*, 1: 145–160, 2019.

67 G.E. Box, G.M. Jenkins, G.C. Reinsel, and G.M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.

68 N. Wu and P. Smyth. Autoformer: Decomposition-based auto-regression for long-term series forecasting. *Advances in Neural Information Processing Systems*, 33: 16704–16715, 2020.

69 H. Zhou, S. Zhang, Y. Zhang, and Y. Wang. Pyraformer: pyramid-attention-based deep learning model for long-range time series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12): 10373–10381, 2021.

70 Y. Wang, Y. Wang, Y. Zhang, and Y. Wang. TiDE: A time series dense encoder for fast and accurate long-term time series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(5): 5474–5483, 2023.

71 L. Zhao and H. Lu. Convolutional neural networks for time series classification: a survey. *Knowledge-Based Systems*, 169: 1–16, 2019.

72 C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3): 273–297, 1995.

73 L. Breiman. Random forests. *Machine Learning*, 45(1): 5–32, 2001.
74 V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: a survey. *ACM computing surveys (CSUR)*, 41(3): 15, 2009.
75 F.T. Liu, K.M. Ting, and Z.H. Zhou. Isolation forest. *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.
76 J. Park and J. Kim. Anomaly detection in multivariate time series using one-class support vector machines with relative density. *Entropy*, 21(3): 248, 2019.
77 Y. Sun and X. He. A survey on deep learning for time series anomaly detection. *Neurocomputing*, 415: 100–110, 2020.