*applied sciences*

*Review*

# On Applications of Spiking Neural P Systems

**Songhai Fan [1,†], Prithwineel Paul [2,3,†], Tianbao Wu [1], Haina Rong [3] and Gexiang Zhang [2,3,*,†]**

1 State Grid Sichuan Electric Power Company, Chengdu 610094, China; fansonghai@126.com (S.F.); wu_tianbao@163.com (T.W.)
2 Research Center for Artificial Intelligence, Chengdu University of Technology, Chengdu 610059, China; prithwineelpaul@gmail.com
3 School of Electrical Engineering, Southwest Jiaotong University, Chengdu 611756, China; hainarong@swjtu.edu.cn
* Correspondence: zhanggexiang19@cdut.edu.cn
† These authors contributed equally to this work.

Received: 8 August 2020; Accepted: 28 September 2020; Published: 9 October 2020

**Abstract:** Over the years, spiking neural P systems (SNPS) have grown into a popular model in membrane computing because of their diverse range of applications. In this paper, we give a comprehensive summary of applications of SNPS and its variants, especially highlighting power systems fault diagnoses with fuzzy reasoning SNPS. We also study the structure and workings of these models, their comparisons along with their advantages and disadvantages. We also study the implementation of these models in hardware. Finally, we discuss some new ideas which can further expand the scope of applications of SNPS models as well as their implementations.

**Keywords:** spiking neural P systems; fault diagnosis; hardware implementation; NP-complete; pattern recognition; membrane computing

## 1. Introduction

In 1965, Gordon Moore predicted that the components of integrated circuits will reach a physical limit [1]—i.e., miniaturization of silicon based circuits will eventually slow down. With this prediction, many computer scientists shifted their focus on constructing computing devices which can be an alternative to the silicon-based computer. Natural motivations to construct these models came from different biological phenomena and the area which investigates these computing mechanisms became well-known as unconventional computing. One of the biggest motivations of constructing bio-computers came from the experiment performed by Leonard Adleman in his laboratory where, with the use of DNA molecules, restriction enzymes and other chemicals, a well-known NP-complete problem—i.e., HPP (Hamilton path problem)—was solved [2]. It initiated a popular area of research known as DNA computing. Furthermore, following the experiment of Adleman, many researchers used DNA molecules as computing units to solve many computationally hard problems. Along with the experimental branch of DNA computing, a theoretical branch of DNA computing was introduced by Tom Head where the splicing of DNA molecules was mathematically modelled using the concepts of formal language theory [3]. This theoretical branch of DNA computing became very popular among formal language theorists [4]. Following the success of DNA computing, Gh. Păun introduced another area inspired from the structure and functioning of biological cells/membranes and it became popularly known as Membrane computing.

Since the introduction of Membrane computing in 1998, it has become a popular direction of research. The seminal paper by Gh. Păun [5] was mentioned by the Institute for Scientific Information(ISI) in 2003 as a fast-breaking in computer science research. Membrane computing models are known as P systems and work as parallel and distributive computing models. Moreover, in P systems the objects present in the membranes are represented by a multiset of objects, strings, etc., and these objects can also evolve using the rules present in the membranes. One target is also attached to the rules so that the newly generated objects can be sent to the membranes mentioned by the target. The newly generated objects can be sent to upper or lower membranes or can be sent to any other connected cells/membranes. Depending on this criteria, membrane computing models are divided into three categories: cell-like, tissue-like and neural-like.

In cell-like P systems, the objects can be sent only to upper and lower membranes of the membrane where the rule is applied. However, the inter-cellular communications are possible in tissue-like and neural-like systems. Neural-like P systems [6] were introduced in 2006, by Ionescu, Păun and Yokomori. These models are known as spiking neural P systems (SNPS). These models were inspired by the structure and functioning of the biological neurons. More specifically, they were inspired from the structure and workings of third generation neural networks which are well-known as SNNs (spiking neural networks). One of the fundamental features of SNNs is that they use time to encode the information and also uses the concept of individual spikes. These properties of SNNs make it a model which is much closer to the biological neurons. Moreover SNNs are hardware friendly and energy efficient [7]. The idea of encoding time as information and individual spikes has been incorporated in SNPS in the framework of formal language theory. In SNPS, the information is encoded in the form of time differences between the spiking of a particular neuron. Moreover, the time difference between the spikes by a specified neuron is collected and considered as the numbers generated by the SNPS. SNPS also work as accepting devices where some neurons are designated as input neurons an,d depending on the positive/negative output by the output neurons, the acceptance/rejection of the strings received in the input neurons are decided to be accepted or rejected. SNPS models are Turing complete. Furthermore, SNPS have established themselves as a very popular direction of research in the last few decades, where the computing power of different variants of SNPS and the use of SNPS models in solving many problems in computing as well as in real-life applications have been investigated extensively. Many variants of SNPS have been introduced by incorporating features from the biological neurons such as asynchronous systems [8], astrocytes [9], rule on synapses [10], communication on request [11,12], synapses with schedules [13], structural plasticity [14], weighted synapses [15], inhibitory synapses [16], anti-spikes [17], etc. These models have also been used in solving problems related to real-life applications, such as fault diagnosis of power systems [18–31], pattern recognition [32–34], computational biology [35], performing arithmetic and logical operations and hardware implementation [36–47], solving computational hard problems [18,48–64], computing morphisms [65,66], biochip design [67], programming for logic controllers [68], etc.

In this paper, we study the SNPS models and their applications. We also give comparisons of the models, their structures and workings. Furthermore, we study their advantages and disadvantages and the implementation of these models in hardware. Finally, we discuss some methodologies to extend the works present in the literature. The main motivation to prepare this survey is as follows:

(1) To give an updated and comprehensive survey of the SNPS models, their structures and workings and their applications;
(2) Study comparison of these models while solving these problems along with their advantages and disadvantages;
(3) Study implementations of these models in hardware;
(4) Introduce some new ideas to expand the scope of applications of SNPS models.

The main contributions of this work are as follows:

(1) Listing a majority of the SNPS models used for solving problems in fault diagnosis, pattern recognition, computational biology, intrusion detection, computing morphism, performing arithmetic and logical operations. Additionally, their use in solving computationally

hard problems, the construction of $\mu$-fluidic biochip design and programming for PLC (programmable logic controller);

(2) Studying a comparison of these models;

(3) Studying their advantages and disadvantages in solving problems;

(4) Discussing a possible extension of these models in applications.

The paper is organized in the following manner: Section 2, discusses the structure of the SNPS model; in Section 3, we study the applications of SNPS models; Section 4 is conclusive in nature.

## 2. Spiking Neural P Systems

**Definition 1** ([6,69])**.** *A spiking neural P system of degree $m \geq 1$ is a $(m+4)$-tuple of the form $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in, out)$, where*

- $O = \{a\}$ *(Singleton alphabet; it is called spike);*
- $\sigma_1, \ldots, \sigma_m$ *are neurons where $\sigma_i = (n_i, R_i), i \geq 1$*

  *where*

  *(a) $n_i \geq 0$ represents the initial number of spikes present in the neuron $\sigma_i$;*

  *(b) $R_i$ represents the finite set of rules present in the neuron $\sigma_i$ and it contains two-types of rules:*

  *(1) Spiking rule: $E/a^c \to a; d$ , $E$ is a regular language over $\{a\}$ and $c \geq 1, d \geq 0$;*

  *(2) Forgetting rule: $a^s \to \lambda, s \geq 1$ and $a^s \notin L(E)$ for all rules of type (1) in $R_i$;*
- *$syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$ represents the synapses and for any $(i, j) \in syn, i \neq j, 1 \leq i, j \leq m$;*
- *$in, out \in \{1, 2, \ldots, m\}$*

The spiking rules of the SNPS are applicable depending on the number of spikes and the regular language over alphabet $a$—i.e., $E$. If at any time $t$, the neuron $\sigma_i$ has $k$ number of spikes and $a^k \in L(E), k \geq c$, then the spiking rule $E/a^c \to a$ rule is applicable. Moreover, after application of this rule, $c$ spikes are consumed and remaining $k - c$ spikes stay in the neuron and one spike is sent to the neurons which have synaptic connection with $\sigma_i$. Note that, in this case, delay $d = 0$. If the delay $d \neq 0$, and the spiking rule are applied at any moment $t$ in any neuron $\sigma_i$, then the neuron becomes inactive—i.e., it will not send any spike out of the neuron and it will not receive any spike from other neurons from time $t, t+1, \ldots, t+d-1$. At $t+d$, the neuron $\sigma_i$ will be open again and it can receive spikes from other neurons. At step $t+d+1$, the spiking rule will be applicable again.

The spiking rules of SNPS can be of the form $E/a^c \to a^p; d$ where $c \geq p \geq 1, d \geq 0$. These rules are known as extended spiking rules and the SNPS model with this type of rules is called as extended spiking neural P systems.

If $E = \{a^c\}$, then the rule $E/a^c \to a; d$ is simply written as $a^c \to a; d$.

The forgetting rules in $\sigma_i$ are of the form $a^s \to \lambda$ and it is applicable only when a neuron contains exactly $s$ number of spikes and $a^s \notin L(E)$ for any spiking rule of the form $E/a^c \to a; d, c \geq 1, d \geq 0$. Furthermore, after application of this rule $s$ spikes are consumed.

Now we discuss the non-deterministic application of the spiking rules—i.e., if at any time $t$, two rules of the form $E_1/a^{c_1} \to a; d$ and $E_2/a^{c_2} \to a; d$ where $L(E_1) \cap L(E_2) \neq \emptyset$ are applicable, then one of the rules is applied (i.e., one of the rule is chosen non-deterministically to be applied). Similarly, if at any time more than two rules are applicable, one of them is selected non-deterministically.

In the next section we discuss the application of the SNPS and their variants. Throughout the paper, in order to improve the readability, instead of writing the full names of the models, we use the acronyms in Table 1.

**Table 1.** List of acronym of SNPS models.

| **ACRONYM** | |
|---|---|
| SNPS | Spiking Neural P Systems |
| FRSNPS | Fuzzy Reasoning Spiking Neural P Systems |
| tFRSNPS | Trapezoidal Fuzzy Reasoning Spiking Neural P Systems |
| AFSNPS | Adaptive Fuzzy Reasoning Spiking Neural P Systems |
| rFRSNPS | Real Fuzzy Reasoning Spiking Neural P Systems |
| WFRSNPS | Weighted Fuzzy Reasoning Spiking Neural P Systems |
| MFRSNPS | Modified Fuzzy Reasoning Spiking Neural P Systems |
| TFSNPS | Time Free Reasoning Spiking Neural P Systems |
| IFSNPS | Intuitionistic Fuzzy Reasoning Spiking Neural P Systems |
| ESTSNPS | Electrical Synaptic Transmission based Spiking Neural P Systems |
| OSNPS | Optimization Spiking Neural P Systems |
| IVFSNPS | Interval-Valued Fuzzy Reasoning Spiking Neural P Systems |
| srSNPS | Spiking Neural P Systems with Self-Updating Rules |
| SNPSPCR | Spiking Neural P Systems with Pre-Computed Resources |
| NLPPS | Neural-like Probabilistic P Systems |
| IMSNPS | Improved Spiking Neural P Systems |
| SNPBR | Spiking Neural P Systems with Budding Rules |
| SNPSNDB | Spiking Neural P Systems with Neuron Division and Budding rules |
| SNPSP | Spiking Neural P Systems with Structural Plasticity |
| ASYSNPS | Asynchronous Spiking Neural P Systems |
| SNPC | Spiking Neural P Systems with Chain structures |
| SNPSAS | Spiking Neural P Systems with Anti-Spikes |
| SNPSDD | Spiking Neural P Systems with Dendritic Delay |
| SNPSACL | Spiking Neural P Systems Astrocyte-Like Control |
| ESNPSEIA | Extended Spiking Neural P Systems with Excitatory and Inhibitory Astrocytes |
| HSNPS | Spiking Neural P Systems with Hebbian Learning |
| TFRSNPS | Triangular Fuzzy Reasoning Spiking Neural P Systems |
| rTFRSNPS | Temporal Fuzzy Reasoning Spiking Neural P Systems with real numbers |

## 3. Applications of Sn P Systems

In this section, we discuss the application of spiking neural P systems in solving problems related to the fault diagnosis of power systems, computationally hard problems, pattern recognition, performing arithmetic and logical operations and their implementation in hardware as well as writing programming languages based on these models, computing morphisms, computational biology, fingerprint recognition, etc.

We give a summary of the SNPS models and their corresponding applications in Figure 1. We also divided the applications of the SNPS models into five subsections—i.e., (1) Power Systems Fault diagnosis; (2) Solving computationally hard problems; (3) Performing Arithmetic and logical operations and hardware implementation; (4) Pattern recognition; (5) Other applications.

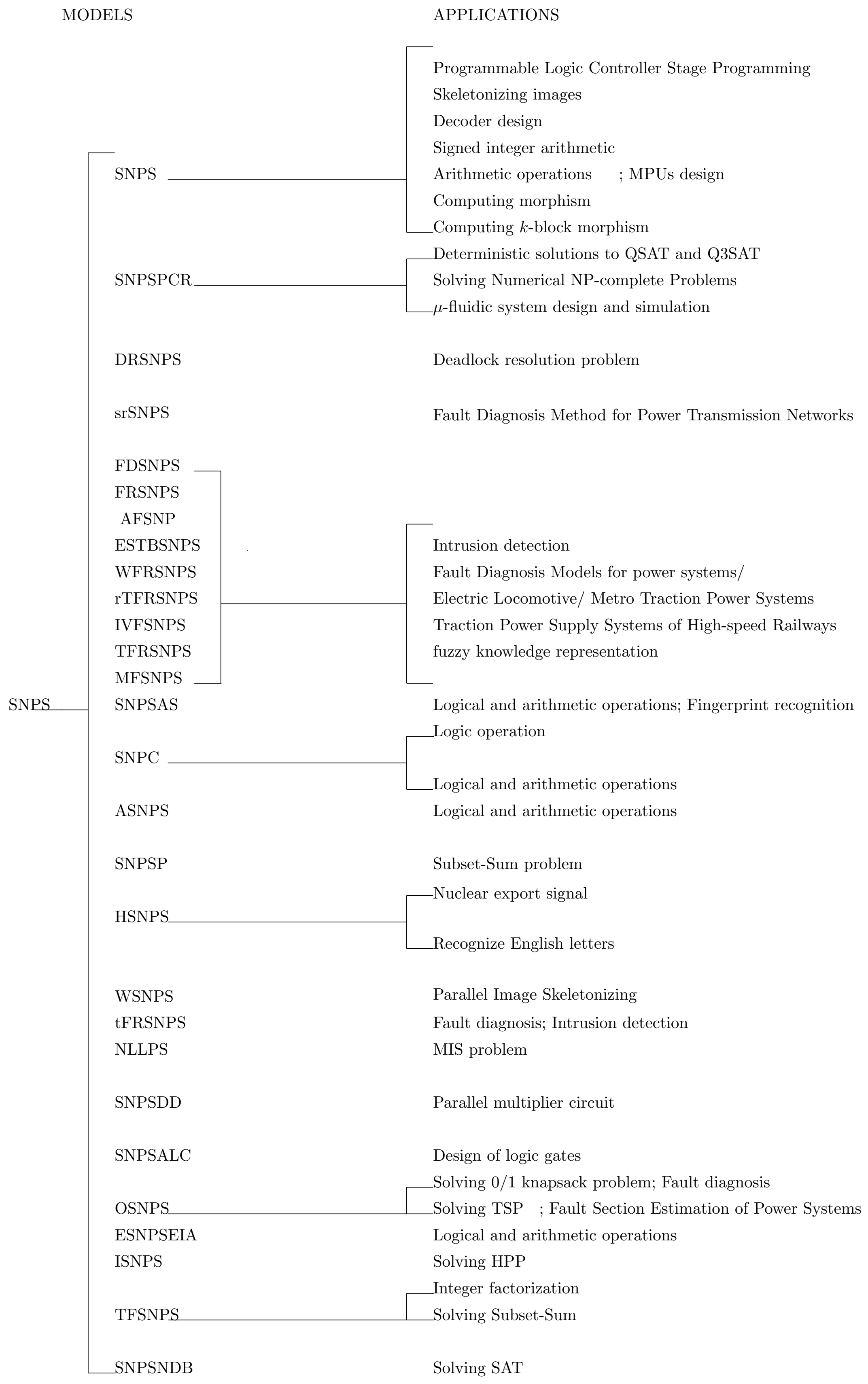

MODELS
APPLICATIONS
SNPS
SNPS
Programmable Logic Controller Stage Programming
Skeletonizing images
Decoder design
Signed integer arithmetic
Arithmetic operations ; MPUs design
Computing morphism
Computing $k$-block morphism
SNPSPCR
Deterministic solutions to QSAT and Q3SAT
Solving Numerical NP-complete Problems
$\mu$-fluidic system design and simulation
DRSNPS
Deadlock resolution problem
srSNPS
Fault Diagnosis Method for Power Transmission Networks
FDSNPS
FRSNPS
AFSNP
ESTBSNPS
WFRSNPS
rTFRSNPS
IVFSNPS
TFRSNPS
MFSNPS
Intrusion detection
Fault Diagnosis Models for power systems/
Electric Locomotive/ Metro Traction Power Systems
Traction Power Supply Systems of High-speed Railways
fuzzy knowledge representation
SNPSAS
Logical and arithmetic operations; Fingerprint recognition
SNPC
Logic operation
Logical and arithmetic operations
ASNPS
Logical and arithmetic operations
SNPSP
Subset-Sum problem
HSNPS
Nuclear export signal
Recognize English letters
WSNPS
Parallel Image Skeletonizing
tFRSNPS
Fault diagnosis; Intrusion detection
NLLPS
MIS problem
SNPSDD
Parallel multiplier circuit
SNPSALC
Design of logic gates
OSNPS
Solving 0/1 knapsack problem; Fault diagnosis
Solving TSP ; Fault Section Estimation of Power Systems
ESNPSEIA
Logical and arithmetic operations
ISNPS
Solving HPP
TFSNPS
Integer factorization
Solving Subset-Sum
SNPSNDB
Solving SAT

**Figure 1.** List of SNPS models and applications.

## *3.1. Power Systems Fault Diagnosis*

The complexity of power systems has increased significantly with the increasing complexity and size of generators, transmission lines, busbars and transformers. PRs (protective relays) and CBs (circuit breakers) protect these devices. One of the major components associated with the power system is called SCADA (supervisory control and data acquisition) and whenever a fault occurs in a power systems, the SCADA system sends a large number of alarm messages. Additionally, at the same time, the protective devices are capable of quick identification of the faults by activation of the PRs and tripping of the CBs to isolate the fault section. It is also a natural phenomenon that the messages received from the SCADA are incomplete and the uncertainty associated with the tripping of PRs and CBs increases the complexity of the fault diagnosis. The parallel and distributed architecture of bio-computing models provided a theoretical framework which can solve these problems effectively. One of the main advantages of these models is that multiple operations can be performed in a step of computation and it helps to solve many real life problems efficiently as well as quickly. The membrane computing model—i.e., fuzzy reasoning spiking neural P system (FRSNPS)—was introduced by Peng, et al. in 2013 [20] with the purpose to solve the problem of fault diagnosis. This model along with the parallel and distributed architecture has capabilities, such as fuzzy knowledge representation, fuzzy reasoning, non-determinism, non-linearity, dynamic reasoning, high understandability and synchronization. Moreover FRSNPS models can simulate model fuzzy production rules graphically which makes the model more easily understandable.

At first we discuss the mathematical structure of the FRSNPS.

**Definition 2** ([20])**.** *A FRSN P system of degree $m \geq 1$, is a $(m+4)$-tuple of the form $\Pi = (A, \sigma_1, \ldots, \sigma_m, syn, I, O)$ where*

*(1)* $A = \{a\}$ *is the singleton alphabet (the object $a$ is called spike).*

*(2)* $r_1, \ldots, r_m$ *are neurons with the form $r_i = (\alpha_i, \tau_i, r_i)$ with $i \in \{1, \ldots, m\}$ where*

*(i) $\alpha_i \in [0,1]$ represents the (potential) value of spike contained in neuron $\sigma_i$ (also called pulse value);*

*(ii) $\tau_i \in [0,1]$ represents the truth value associated with neuron $\sigma_i$;*

*(iii) $r_i$ (firing/spiking rules) contained in neuron $\sigma_i$ are of the form $E/a^{\alpha} \rightarrow a^{\beta}$, where $\alpha, \beta \in [0,1]$.*

*(3)* $syn \subseteq \{1,2,\ldots,m\} \times \{1,2,\ldots,m\}$ *with $i \neq j$ for all $(i,j) \in syn, 1 \leq i,j \leq m$ (synapses between neurons).*

*(4)* $I, O \subseteq \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ *represent the input neuron set and output neuron set, respectively.*

The rules of this model are different from the rules in the SNPS model in [6] and also instead of the number of spikes, the value of the spikes is represented by number $\alpha_i \in [0,1]$. The neurons in fuzzy reasoning spiking neural P systems (FRSNPS) are divided into two categories—i.e., proposition neurons and rule neurons. Moreover, rule neurons are divided into two categories—i.e., AND and OR neurons. Based on the firing mechanism and matrix operations, a reasoning algorithm was introduced in [20]. This matrix reasoning algorithm in the framework of FRSNPS is capable of diagnosing single and multiple faults irrespective of proper functioning of the PRs. Moreover the fault diagnosis method based on FRSNPS has good fault tolerant capacity. This method is also suitable for online applications, since the construction and storing of these models in a file can be done in advance and it only takes five reasoning steps to obtain the results of the diagnosis. Inspired by this model, a new variant of FRSNPS, called MFRSNPS (modified fuzzy reasoning spiking neural P systems) was introduced in [25] by He et al. In this model, rule neurons are divided into three categories—i.e, general, AND and OR rule neurons. MFRSNPS have also been used in fault diagnoses of metro traction power supply systems [25]. The energy systems of rail transportation systems are called traction power supply systems and play an important role in safe and reliable operations of trains. It is important to note that the identification of the faults in traction power supply systems is important for uninterrupted supply of the power. Additionally, the identification of the faults is impacted by the uncertain and incomplete operation information received from the SCADA. The MFRSNPS models are very effective in the diagnosis of the fault section while certain/uncertain as well as complete/incomplete informations is received from the SCADA systems.

Since the introduction of the first FRSNPS model in [20], different variants of FRSNPS models have been introduced where the value of the spikes inside the neurons is represented by a triangular fuzzy number (TFRSNPS) [27], intuitionistic fuzzy numbers (IFSNPS) [70], interval-valued fuzzy numbers (IVFSNPS) [71,72], real numbers (rFRSNPS) [21] and trapezoidal fuzzy numbers (tFRSNPS) [73]. Additionally, a temporal fuzzy reasoning spiking neural P systems with real numbers (rTFRSNPS) was proposed [74] by Huang, et al. in 2016. It is also important to note that, in tFRSNPS, both the value of the spike and truth value are represented by a trapezoidal fuzzy number in $[0,1]$; in rFRSNPS, both potential and truth value are real numbers in $[0,1]$. However, in TFSNPS, the potential is a triangular fuzzy number and truth value is a real number in $[0,1]$; in IFSNPS, the potential value is an intuitionistic fuzzy number and the CF is a real number in $[0,1]$, and in IVFSNPS, both potential and CF are interval-valued fuzzy numbers. Moreover, the rule neurons of these models have different structures. The tFRSNPS and rFRSNPS models contain proposition neurons and, in general, AND and OR neurons. However, the TFRSNPS, IFSNPS and IVFSNPS models have proposition neurons and AND and OR rule neurons. IVFRSNPS models are flexible and capable of handling incomplete and uncertain messages. The IFSNPS models are also very effective in the identification of faults in power systems where the messages received from the SCADA systems are incomplete and uncertain. The interval-valued fuzzy reasoning spiking neural P systems (IVFRSNPS) [71] introduced in 2019 can also process the incomplete and uncertain messages and can identify the faulty sections efficiently.

The efficiency of the reasoning algorithm plays an important role in the efficiency of system and identification of faults quickly and effectively. In tFRSNPS, the inference ability of these models was developed by introducing a matrix-based fuzzy reasoning algorithm which is based on dynamic firing mechanism. The rFRSNPS model proposed in [21,75], has been used for the fault diagnosis of electric locomotive systems where the relationship between the breakdown signals and faulty sections in the locomotive unit have been represented by the fuzzy production rules. These rules are further simulated by the rules in rFRSNPS. More specifically, in [21,75] fault diagnosis model for Shaoshan4 (SS4) electric locomotive systems has been investigated. In [76], the rFRSNPS model was used for classification of ten types of faults occurring in the lines of power systems. The proposed method in [76] is a combination of wavelet transform, singular value decomposition and a rFRSNPS model. Moreover, the proposed method is more feasible and effective and in comparison with benchmark methods has superiority in robustness of noise. In 2018, another fault line detection method based on the rFRSNPS model was introduced [77] by Rong et al. More specifically, this model was introduced for fault diagnosis in small current grounding systems. Additionally, steady and transient component features present in the current or voltage signals were considered in this paper. Information gain degree is an important aspect of feature information fusion and the weight of the importance of features is measured by it. In [77], these features were reduced using the rough set theory. Another fault diagnosis model based on the trapezoidal fuzzy reasoning spiking neural P systems was introduced in [19] by Wang et al. in 2015. This model was named FDSNPS and a matrix-based fuzzy reasoning algorithm was introduced. The strictly mathematical expression of this model helps this model to be intuitively illustrative, and also, since it can effectively handle the incomplete and uncertain messages, this model has good fault tolerant capacity. Moreover, the trapezoidal fuzzy reasoning spiking neural P systems framework is capable of representing the relationship between the PRs and the faults efficiently and hence the model becomes easily understandable. Additionally, the fault diagnosis model introduced in [19] is capable of performing the task of identifying single/multiple faults when incomplete and uncertain messages are received from SCADA systems. Triangular fuzzy reasoning spiking neural P systems (TFRSNPS) were introduced in [27], by the integration of the idea of triangular fuzzy numbers in SNPS. The effectiveness and ability of the fuzzy reasoning method based on this model have been investigated over the ring network with 220kV. More specifically, the reasoning method has been used to identify the fault in the bus, line and transformer in the ring network. This process is performed after the PRs and CBs receive incomplete and uncertain messages from SCADA. The reasoning algorithm based on this model is fast and highly accurate, while longitudinal differential protection has been considered.

Another new variant of FRSNPS was introduced in 2015 by Wang, et al. [24] which is conceptually different from the previous models. In this model. weights are associated with the synapses connecting the neurons and it is known as weighted fuzzy reasoning spiking neural P systems (WFRSNPS). Moreover this model was used for fault diagnosis in traction power supply systems, which generally occur in high-speed railways. The neurons of this system are of four types—i.e., proposition neuron, general, AND and OR rule neurons. The relationship between the PRs and CBs and the faults can be easily represented in the framework of WFSNPS and also a weighted matrix-based reasoning algorithm was introduced in [24]. Additionally, the proposed method has satisfying results for fault diagnosis in normal supply and over zone feeding where the information received from the SCADA system is complete/incomplete. Moreover, with simple reasoning this method can obtain good results. This idea of associating weights with the synapses was further extended in [22] by Tu et al. It is well-known that the machine learning algorithms deal with updating the weights in the neural networks and these algorithms have been used to solve many problems in neural networks. This idea was extended for spiking neural P systems in [22] where the weights associated with the synapses were updated using the learning algorithms in the framework SNPS. Additionally, it was used very effectively in identifying the faults in power systems. Adaptive fuzzy spiking neural P systems (AFSNPS) are one of the variants of WFSNPS. The problem of fault diagnosis of power systems was solved in the framework of AFSNPS in [22] by Tu et al., where the rules were constructed in such a manner that they could simulate weighted fuzzy production rules. Moreover, the adjustment of the weights on the synapses connecting the proposition neurons and rule neurons helped the system to adjust automatically. Additionally, AFSNPS models can perform dynamic fuzzy reasoning. Along with simple reasoning processes, this method is faster compared to other algorithms and has parallel processing capabilities. In AFSNPS, the weights are updated using the Widrow–Hoff learning rules. AFSNPS have some intrinsic properties, such as simple learning process, fast speed and parallel processing. These properties are useful in solving the problem of fault diagnosis and hence AFSNPS is a good model for fault diagnosis. The AFSNPS has been further improved by integration of PSO (particle swarm optimization) in the learning algorithm of AFSNPS [23] in 2016 by Wang et al. In this method, the PSO algorithm optimizes the learning algorithm in the framework of AFSNPS. This method also has a matrix reasoning algorithm. The PSO algorithm also improves the efficiency and accuracy of the method. Additionally, compared with other methods AFSNPS with PSO algorithms have better convergence speed of diagnosis, higher accuracy and are also conducive to the changes in the topology of the grid. One of the major advantages of the AFSNPS model is the learning ability to adjust automatically. This feature helps to obtain the diagnosis results which are closer to the actual situations. However since a large amount of samples are required to train these networks, modelling of large scale networks is difficult using this method.

In 2014, the OSNPS model [18] was introduced by Zhang et al., which provides a framework to solve optimization problems using SNPS. The fault section estimation problem can be formulated as an optimization problem (i.e., FSE, fault section estimation). The effective solution to this optimization problem is possible by formulating it into the 0–1 integer programming problem. In [26], OSNPS was used to solve this problem by Wang et al. The structure of OSNPS is very different from the traditional SNPS models. In this model, an ESNPS (extended spiking neural P system) was introduced where a guider is associated with each rule inside the neurons in order to adjust the selection probabilities. This model is capable of automatically finding the faults in the power systems efficiently after receiving the inputs from SCADA systems. Moreover, it is capable of finding the faults when the received messages are uncertain and incomplete. Additionally, this model is efficient in finding single/multiple faults. Recently, in [78], a new variant of the OSNPS model is introduced to solve the problem of fault section estimation. This model is called AOSNPS (Adaptive Optimization Spiking Neural P System) and it is efficient in the identification of single and multiple faults as well as multiple faults while receiving with incomplete and uncertain messages from SCADA. One of the distinctive feature of this model is dynamic guider algorithm which has capabilities such as adaptive learning and diversity based adaption ability.

Based on the dissolved and gas analysis (DGA) in the framework of FRSNPS [31], a new method for fault diagnosis was introduced in 2016. DGA is a very informative method. In this method, a sample of oil is taken and the concentration of the dissolved gases in this sample is tested. The IEC ratio of gases is given as input signature in the FRSNPS diagnosis model and the reasoning process can also be graphically represented by FRSNPS. The uncertainty in processing—the reasoning based on the rules of the system, symbolic representation and parallel computing ability—makes this diagnosis method more accurate, faster and adaptive to any kind of change happening in the system. Furthermore, the matrix-based reasoning process can be graphically represented, which makes it more easily understandable.

Very recently, some new types of SNPS have been introduced by integrating some new biological phenomena into the SNP model. Moreover, these models have been used in the fault diagnosis of power systems. A new method of fault diagnosis method for power transmission networks was introduced in 2020 by Liu et al., where spiking neural P systems with self-updating rules and biological apoptosis mechanisms (srSNPS) [29] were used for this task. This model was constructed by combining the idea of attribute reduction ability ( concept from rough set) and apoptosis mechanism (from biological neurons) in the framework of membrane computing models—i.e., spiking neural P systems. One of important features of apoptosis algorithms is that these algorithms can devise the conditional neurons in such a manner that they can remove the unnecessary information received from SCADA systems. This further helps the model to perform better while uncertain and incomplete messages are received. The fault diagnosis method in the framework of srSNPS can be divided into four sections—i.e., (1) transmission network partition; (2) construction of SNPS model; (3) the correction of the pulse value; (4) computing along with evaluation of protective device behaviour. The topological adaptive ability is improved by the transmission network partition. Moreover, this method has good ability to interpret the diagnosis result. However, during this process, it maintains high fault tolerance along with fast diagnosis speed, even though the messages received from the SCADA system are uncertain and incomplete. One of the major advantages of this model is that historical statistics and expertise are not required. Other new variant of the SNPS model, which have been used for the task of fault diagnosis, are called electrical synaptic transmission-based spiking neural P systems(ESTSNPS). This model was introduced in [30] by adding some new synapses and neurons, removing the delay and adding bidirectional characteristics of electric synaptic transmission with the original definition of SNPS. Additionally, the rules of this model are on the electric synapse. The fault diagnosis method based on ESTSNPS is very effective and are capable of identifying single/multiple faults, misinformation faults in distribution networks with DGs (distributive generators) with high accuracy quickly. It also has high traceability.

All the above models mentioned above are efficient. However the implementations of these models are manual which are also very time consuming as well as inefficient when dealing with large scale networks. In order to solve this problem, an automatic implementation method [28] was introduced in 2019. The method introduced in [28] is called MCFD (membrane computing fault diagnosis) and it can solve the problem of fault diagnosis automatically.

Comparisons of the SNPS models solving the fault diagnosis problem are presented in Figures 2–4.

**Remark 1.** *(1) Different variants of SNPS have been used to solve the problem of fault diagnosis of power systems and these models can successfully solve these problems even after receiving incomplete and uncertain messages from SCADA systems. Additionally, these models are capable of finding single/multiple faults. However, these models are not effective when dealing with large scale power systems. It can be a future direction* *of research whether the above discussed models can be further extended using the properties of fuzzy logic to efficiently solve fault diagnosis problems for large scale networks.*

*(2) The basic system of the automatic implementation method introduced in [28] is an FRSNPS model. It can be a future direction of research to construct similar methods for other variants of FRSNPS and it will also be interesting to study the comparison of the time and space complexity of these methods with the method introduced in [28].*

*(3) Until now the models used in the fault diagnosis method have static network structures. It can be further investigated whether it is possible to construct fault diagnosis methods using the SNPS models with dynamic network structure, such as SNPS with structural plasticity [14] and SNPS with neuron budding rules [55] or SNPS with neuron division and budding rule [56,57], etc., which can increase the size of the synapse graph. Furthermore, we can compare the performance of these models with the existing models with static network.*

| MODEL | PROPERTIES | ADVANTAGES AND DISADVANTAGES |
|---|---|---|
| 1. FRSNPS(FSNPS) | 1. $\Pi = (A, \sigma_1, \sigma_2, \ldots, \sigma_m, syn, I, O)$<br>2. Neuron: $\sigma_i = (\alpha_i, \tau_i, r_i)$; $\alpha_i \in [0,1]$ (value of the spike) $\tau_i$(truth value); Spiking rule: $r_i : E/a^{\alpha} \to a^{\beta}$<br>3. No delay<br>4. Proposition neuron: $\sigma = (\alpha, \tau, r)$, $r : E/a^{\alpha} \to a^{\alpha}$ AND-type rule neuron: $E/a^{\alpha} \to a^{\beta}, \alpha = min(\alpha_1, \alpha_2, \ldots, \alpha_n)$, $\beta = \alpha * \tau$; OR-type rule neuron: $E/a^{\alpha} \to a^{\beta}$, $\alpha = max(\alpha_1, \alpha_2, \ldots, \alpha_n), \beta = \alpha * \tau$<br>5. Composite fuzzy production rule:<br>Type 1 $R_i$: IF $p_1$ and $p_2$ and ...$p_{k-1}$ Then $p_k(CF = \tau_i)$<br>Type 2 $R_i$: IF $p_1$ THEN $p_2$ and $p_3$ and ... and $p_k$ $(CF = \tau_i)$<br>Type 3 $R_i$: IF $p_1$ or $p_2$ or ... or $p_{k-1}$ THEN $p_k(CF = \tau_i)$ | 1. Diagonize single and multiple faults;<br>2. Good fault tolerant capacity; |
| 2. tFRSNPS | 1. Neurons: $\sigma_i = (\theta_i, c_i, r_i)$, $\theta_i$(trapezoidal fuzzy number in $[0,1]$ $c_i$ (trapezoidal fuzzy number in $[0,1]$)<br>2. Proposition neuron; general, AND and OR -rule neuron<br>3. Composite fuzzy production rule: Type 1, Type 2, Type 3, Type 4 and Type 5<br>Type 4: $p_j(\theta_j) \to p_k(\theta_k), \theta_j$ (trapezoidal fuzzy number in $[0,1]$)<br>Type 5: $p_1(\theta_1) \to p_2(\theta_2)$ or ... or $p_k(\theta_k)$ | 1. Matrix-based fuzzy reasoning algorithm based on dynamic firing mechanism;<br>2. Quick and effective identification of faults. |
| 3. AFSNPS | 1. $\Pi = (A, N_p, N_r, syn, I, O)$; $N_p$(proposition neuron set) $N_r$(rule neuron set)<br>2. $\vec{\omega_i} = (\omega_{i1}, \omega_{i2}, \ldots, \omega_{is_i})$ (weights on synapses connecting proposition and rule neurons)<br>3. Spiking rule in proposition neuron: $E/a^{\alpha} \to a^{\beta}$ Spiking rule in rule neuron: $E/a^{\alpha} \to a^{\beta}$<br>4. Weighted fuzzy production rules of Type 1, Type 2 and Type 3 | 1. System can adjust automatically;<br>2. Dynamic fuzzy reasoning Faster, simple and have parallel processing capability; better convergence speed of diagnosis, higher accuracy;<br>3. Not good for large scale networks; |
| 4. rFRSNPS | 1. Proposition neuron: $\sigma_i = (\theta_i, r_i)$ ($\theta_i$ is a real number in $[0,1]$)<br>2. Rule neuron: $\sigma_i = (\delta_i, c_i, r_i)$, $\delta_i$ (potential value, real number in $[0,1]$) $c_i$(truth value, real number in $[0,1]$)<br>3. Spiking rule: $E/a^{\theta} \to a^{\beta}$, $\theta$ and $\beta$ real numbers<br>4. $R_1, R_2, \ldots, R_{12}$ fuzzy production rules | 1. More feasible and effective ;<br>2. Superiority in robustness of noise; |

**Figure 2.** List of SNPS models solving fault diagnosis problem.

| MODEL | PROPERTIES | ADVANTAGES AND DISADVANTAGES |
|---|---|---|
| 5. MFRSNPS | 1. It contains three types of rule neurons: general, AND and OR. | 1. Effective in diagnosis of the fault section with certain/uncertain and complete/incomplete informations; |
| 6. TFRSNPS | 1. Neurons: $\sigma = (\alpha_i, \beta_i, r_i)$, $\alpha_i$(triangular fuzzy number, potential value), $\beta_i \in [0,1]$ is a real number; 2. Spiking rule: $a^{\alpha} \to a^{\alpha}$ or $a^{\alpha} \to a^{\alpha'}$, $\alpha, \alpha'$(triangular fuzzy numbers). 3. Proposition neuron, AND and OR type rule neuron 4. Fuzzy production rule: Type 1: IF $p_1$ AND $p_2$ AND …AND $p_n$ THEN $p_k$ (CF $= \beta$) Type 2: IF $p_1$ OR $p_2$ OR …OR $p_n$ THEN $p_k$ (CF $= \beta$). | 1. Used in identification of faults in the bus, line and transformer in the ring network; 2. Effective in identification of faults in power systems with incomplete and uncertain informations; 3. Reasoning algorithm based on this model is fast and highly accurate; |
| 7. IFSNPS | 1. Neuron: $\sigma_i = (\alpha_i, \tau_i, r_i)$, $\alpha_i$ (intuitionistic fuzzy number, initial value of spikes), $\tau_i \in [0,1]$ (real number, confidence level) 2. Spiking rule: $E/a^{\alpha} \to a^{\alpha}$ or $E/a^{\alpha} \to a^{\beta}$ 3. Proposition neurons, AND and OR rule neurons 4. Models fuzzy production rules of Type 1 and Type 2. | 1. Effective in identification of faults in power systems with incomplete and uncertain informations; |
| 8. ESTSNPS | 1. $\Pi = (A, \sigma_1, \sigma_2, \ldots, \sigma_m, I/O, O/I)$ 2.$E/a^{\alpha} \to a^{\beta}; E/a^{\beta} \to a^{\alpha}$ $E/a^{\alpha} \to \lambda; E/a^{\beta} \to \lambda, \alpha \in (-1,0,1), \beta \in (-1,0,1)$ 3. $syn \subseteq \{1,2,\ldots,K,m\} \times \{1,2,\ldots,K,m\}, i \neq j, (i,j) \in syn$ | 1. Effective and capable of identifying single/ multiple faults, misinformation faults in distribution networks with DGs (distributive generators) with high accuracy and quickly; High traceability. |
| 9. OSNPS | 1. $\Pi = (O, \sigma_1, \ldots, \sigma_{m+2}, syn, I_0)$ (ESNPS) 2. Neuron: $\sigma_i = (1, R_i, P_i), R_i = \{r_i^1, r_i^2\}, r_i^1 = \{a \to a\}$, $r_i^2 = \{a \to \lambda\}$, $P_i = \{p_i^1, p_i^2\}$ (probabilities associated with rule $r_i^j$) $p_i^1 + p_i^2 = 1$. 3. $\sigma_{m+1} = \sigma_{m+2} = (1, \{a \to a\})$ 4. $syn = \{(i,j) \vert (i = m+2 \wedge 1 \leq j \leq m+1) \vee$ $(i = m+1 \wedge j = m+2)\}$ 5. $I_0 = \{\sigma_1, \ldots, \sigma_m\}$ 6. Combination of $H$ ESNPS | 1. Capable of automatically finding the faults in the power systems efficiently; 2. Capable of finding the faults with uncertain and incomplete messages; efficient in finding single/multiple faults; 3. Adaptive learning and diversity based adaption ability; |
| 10. IVFSNPS | 1. Neuron: $\sigma_i = (\theta_i, c_i, r_i)$, $\theta_i$(value of spikes, interval-valued fuzzy numbers) $c_i$(CF of a fuzzy production, interval-valued fuzzy number) 2. Spiking rule: $a^{\theta} \to a^{\theta}$ or $a^{\theta} \to a^{\beta}$, $\alpha, \beta$ (interval-valued fuzzy numbers) 3. Proposition neuron, $\oplus$ and $\otimes$ rule neurons 4. Simulates fuzzy production rules of Type 1: if $p_1$ and $p_2$ and …and $p_{k-1}$ then $p_k$ (CF $= c$) Type 2: if $p_1$ or $p_2$ or …or $p_{k-1}$ then $p_k$ (CF $= c$). | 1. Effective in identification of faults in power systems with uncertain and incomplete messages; |

**Figure 3.** List of SNPS models solving fault diagnosis problem.

| MODEL | PROPERTIES | ADVANTAGES AND DISADVANTAGES |
|---|---|---|
| 11. rTFRSNPS | 1. Proposition neurons: $\sigma_i = (\theta_i, t_i, r_i)$;<br>Rule neurons: $\sigma_i = (\delta_i, c_i, r_i)$<br>$\theta_i, \delta_i \in [0,1]$ (real number, potential value of spikes),<br>$t_i \in \{0,1\}$ (state of proposition)<br>$c_i \in [0,1]$(real number, truth value of spikes)<br>2. Spiking rule: $E/a^{\theta} \rightarrow a^{\beta}, \alpha, \beta \in [0,1]$<br>3. Proposition neuron, AND and OR neuron; | 1. Use temporal order information of alarm messages to model candidate fault sections;<br>2. Good ability to handle incomplete and uncertain alarm messages;<br>3. Effective in single and multiple fault situations with/without incomplete and uncertain alarm messages; |
| 12. srSNPS | 1. $\Pi = (O, M_e, \sigma_1, \ldots, \sigma_m, syn, in, out)$<br>2. Microenvironment: $M_e = (D_i, C_j)$, $D_i = (\theta_{di/T}, T, f_i)$ ($i$-th decision making neuron),<br>$\theta_{di/T} \in \{0,1\}$ (pulse value of $i$-th DN at time $T$),<br>$T$ (sequence time of spikes),<br>$f_i : E/\{a^{\theta di/T} \rightarrow \lambda; g = 0\}$ (forgetting rule of $i$-th DN)<br>$C_j = (\theta_{cj/T}, T, f_j, A_{rj})$ ($j$-th condition neuron(CN) in $M_e$),<br>$\theta_{cj/T} \in \{0,1\}$ (pulse value of $j$-th CN at time $T$)<br>$f_j : E/\{a^{\theta_{cj/T}} \rightarrow \lambda; g = 0\}$<br>$A_{rj} : E/\{C_j\} \rightarrow \{Algorithm1^L\}$ (Apoptosis rule in $M_e$)<br>3. Proposition neuron: $\sigma_i = (\theta_i, r_i, e_i)$, $r_i : E/a^{\theta} \rightarrow a^{\theta}$,<br>$\theta_i \in \{0,1\}$<br>$e_i : E/a^{\theta} \rightarrow a^{\overline{\theta}}$, $\theta \in \{0,1\}$(pulse value), $\overline{\theta} \in \{0,1\}$<br>(anti-spike of $\theta$) | 1. Self-updating rules and biological apoptosis mechanism;<br>2. Perform better when uncertain and incomplete messages are received;<br>3. Topological adaptive ability;<br>good ability to interpret the diagnosis result;<br>4. High fault tolerance; fast diagnosis speed;<br>5. Historical statistics and expertise is not required; |

**Figure 4.** List of SNPS models solving fault diagnosis problem.

In the next section, we discuss some models of SNPS which are capable of solving computationally hard problems in polynomial and linear time. It is also well-known that sometimes it is not possible to give analytic solutions to computationally hard problems. So, it is always useful to have a method which can approximate the solution. In the next section we also discuss such methods based on spiking neural P systems.

### *3.2. Solving Computationally Hard Problems*

SNPS and their variants are very powerful and these models can characterize recursively enumerable languages [79]. Until now, many variants of SNPS have been introduced, and investigating computational power of these models has been a popular direction of research. However, the use of these models in solving computationally hard problems has been much less investigated. The study of the use of SNPS models in solving computationally hard problems was initiated in 2006 by Chen, Ionescu and Ishdorj. In [49], along with SNPS, a new idea inspired by the biological phenomena was incorporated into the SNPS model. It is well known that, in biological neurons at any time, there exist millions of inactive neurons. In [49], this idea was used and it was assumed that an arbitrary large number of inactive neurons are present in the initial configuration—i.e., neurons are present in exponential numbers which can be activated in polynomial time. Furthermore, this idea was used to deterministically solve the $SAT$ problem which is a popular NP-complete problem in constant time—i.e., the computation starts with an exponentially large precomputed workspace because of the exponentially large number of inactive neurons present in the initial configuration where the neurons can be activated in constant time. This idea of activating the inactive neurons in polynomial time was further extended to solve some computationally hard problems in [48,50,51]. This new SNPS model is also called SNPS with pre-computed resources. The fundamental concept of these models has been that, instead of the production of exponential workspace in linear/polynomial time, it is always useful to start with a precomputed workspace which is exponentially large. The neurons in these models are inactive until a spike enters the neuron. In [51], the idea of SNPS with pre-computed resources was used to solve the NP-complete problem $SUBSET - SUM$ by Leporati and Gutiérrez-Naranjo. Initially a semi-uniform family of SNPS was constructed and then specified instances of $SUBSET - SUM$ were solved by using this system. The systems introduced in [51] are deterministic. However depending on the size of the instances, the size of the system also grows exponentially. Another solution of the $SUBSET - SUM$ problem using SNPS with pre-computed resources is discussed in [53] by Leporati et al. Along with the concept of pre-computed resources, the concept of maximal parallelism has been used to solve the problem in [53]. Using maximal parallelism the conversion of any integer from binary notation to unary notation is possible in polynomial time and during this process the idea of pre-computed resources is used—i.e., the initialization of the P systems with exponential number of spikes. It has also been shown in [53] that this conversion is not possible when maximal parallelism is forbidden. Moreover, a construction of a uniform family of SNPS has been provided where different sub-systems, whether deterministic or non-deterministic, sequential or maximal parallel, work together in order to solve the $SUBSET - SUM$ problem. This result was further improved in [52]. In [52], a uniform construction of SNPS was introduced where the extended rules or application of the rules in parallel was avoided. Moreover, the SUBSET-SUM and SAT problems were solved using a constant number of steps where the system works in a non-deterministic manner. In 2010, deterministic solutions of two well-known PSPACE-complete problems—i.e., $QSAT$ and $Q3SAT$ using the concept of SNPS with pre-computed resources were investigated in [48] by Ishdorj et al. Moreover, a deterministic solution of these two problems was provided. Additionally, using this method, any instance of the $QSAT$ problem can be solved using a time which is linear with respect to $n$ and $m$, where $n$ represents the number of Boolean variable and $m$ represents the number of clauses in the instance. Similarly, using this method, any instance of $Q3SAT$ can be solved in, at most, $n^3$ time, where $n$ represents the number of Boolean variables.

Along with the assumption of presence of arbitrary number of spikes, the spiking rules also play an important role in solving computationally hard problems. Generally the SNPS models contain two types of rules—i.e., spiking rule and forgetting rule. Many researchers also introduced many new types of rules which improve the computational power of the SNPS and also these rules can be used in many areas of applications [63]. These rules of the SNPS systems also play an important role in solving computationally hard problem. The SNPS systems with budding rules [55], SNPS with neuron division and budding rules [56,57] and SNPS with structural plasticity [58] are examples of such models. By applying the budding rules in a maximal parallel manner, in polynomial time an SNPS is capable of increasing the size of its synapse graph. Moreover this feature can be further used to solve the well-known NP-complete problem $SAT$ in deterministic polynomial time with respect to the $n$ number of Boolean variables and $m$ number of clauses in an instance. In [56], SNPS with neuron division and budding rule were introduced by Pan, Păun and Pérez-Jiménez in 2009. These rules are capable of generating exponential workspace in linear time. Furthermore, the SNPS with neuron division and budding was used to obtain a uniform solution to the $SAT$ problem. Similarly, as with SNPS with budding rule and neuron division and budding rules, a new variant of SNPS was introduced in [58] by Cabarle, Hernandez, and Martínez-del-Amor in 2015. This model is inspired by a popular phenomenon in biological neuron networks—i.e., during the learning process biological neurons are capable of creating new synaptic connections as well as removing old synaptic connections. This feature is known as structural plasticity. In [58], a new model SNPS with structural plasticity was introduced by combining the features of both SNPS and structural plasticity (SNPSSP). In SNPSSP, the plasticity rule was introduced which can create new connections and remove old connections. Furthermore, this model has been used in solving the $SUBSET - SUM$ problem. More specifically, a uniform solution and non-uniform solution of $SUBSET - SUM$ were provided in [58] using SNPSSP.

In the case of SNPS with budding rule, neuron with division and budding rule and structural plasticity, new rules were introduced. However, only using some restrictions of the traditional spiking and forgetting rule, is it possible to solve computationally hard problems [59]. Improved SNPS (IMSNPS) is one such model and it was introduced in 2013 by Xue and Liu. The distinctive features of this model are that the priorities are associated with the rules present in the system. More specifically, the rules are divided into three categories—i.e., (1) $E/a^c \rightarrow a^p, c \geq 1, c \geq p, p \geq 1$; (2) $a^c \rightarrow a^p, c \geq 1, c \geq p, p \geq 1$; (3) $a^s \rightarrow \lambda, s \geq 1$. According to the priority relation added with the systems, $(1) > (2) > (3)$—i.e., the application of the rules will depend on this relation. Another distinctive feature of this model is that it can have multiple output neurons. Furthermore, this model has been used to solve the DHP (directed Hamilton path) problem and the algorithm based on this model is capable of automatically scanning all possible paths and then filtering out the $DHP$ in linear time. Time-free spiking neural P systems are another variant of SNPS with traditional spiking and forgetting rules. These models were inspired from the biological phenomenon that different biochemical processes in living organisms take different time due to external factors. It was introduced in [62] by Pan, Zheng and Zhang and has been used to solve the $SAT$ problem and give uniform solution to integer factorization and $SUBSET - SUM$ problem in [60] and [54], respectively. In this model, an execution time is associated with each rule using the *time mapping*. Some of the timed SNPS are independent of the time mapping and can generate the same computational results. These models are known as time-free SNPS. A family of uniform time-free SNPS was used to solve integer factorization problem [60] by Liu, et al. in 2015. In [64], a neural-like probabilistic P system was proposed by Xue and Jevanos where a priority level and probability were associated with each rule. This model was used to solve the MIS (maximal independent set) problem. Some of the distinctive features of this model is that (1) the overall size is not required by the individual processors present in the paper (2) the processors communicate with each other using one-bit message. Additionally, the experiments performed in [64] show that NPP systems can solve distributed computational problems very efficiently.

It is well-known that it is difficult to obtain an analytic solution for NP-complete and NP-hard problems. So many researchers tried to explore the avenues when it is very difficult to obtain an analytic solution, some algorithms can approximate the solutions of these problems. In [18], a new variant of SNPS was introduced by Zhang et al. in 2014 in order to solve a well-known NP-complete problem—i.e., Knapsack problem. These models comprise optimization spiking neural P systems (OSNPS). In this model, a novel way to obtain approximate solutions of computationally hard combinatorial optimization problems instead of using the concepts of MIEAs (membrane inspired evolutionary algorithms) was proposed. Furthermore, an extended spiking neural P system was constructed by incorporating the concept of probabilistic selection of evolution rule and multi-neurons output. Note that the ENPS constructed in this paper is fundamentally different from the extended spiking neural P system discussed earlier. The family of ENPS is also called OSNPS where the concept of a guider was introduced in order to adjust the probabilities associated with the rules adaptively. Moreover, this model was used to solve combinatorial optimization problems such as knapsack problems. Additionally, the experimental results infer that, with respect to other optimization algorithms, the optimization performance of the OSNPS is better. It is important to note that one of the main advantages of this model is that it is performed by neural systems—i.e., a computing device instead of humans. The study of OSNPS was further extended in 2018 by Qi and Liu where this model was used to solve a well-known NP-hard problem—i.e., TSP (travelling salesman problem) [61]. The major difference of this model with the model introduced in [18] is that, in this model, the task of adjusting the probability associated with the rules is performed by GA algorithm—i.e., it works as a guider algorithm. This model is also effective for problems of smaller scale. Whenever the scale of the problem becomes larger in comparison with standard GA, the OSNPS model needs more time to solve the problem. Recently, a new variant of OSNPS is introduced by Zhu et al. and this model is called as AOSNPS (Adaptive Optimization Spiking Neural P System) [78]. This model is different from the previous OSNPS models. In this model, in order to control the moving operators, a dynamic guider algorithm is introduced. This guider algorithm has adaptive learning and delivery based adaption capability. Furthermore, the numerical results presented in [78] show that the proposed method can effectively solve $0/1$ knapsack problem effectively and has better performance while solving the same class of problems having large number of elements.

We give a summary of the SNPS models and the corresponding computationally hard problems which can be solved by these models in Figure 5.

| MODELS | PROPERTIES | PROBLEM SOLVED |
|---|---|---|
| 1. TFSNPS | 1. $\Pi_e = (O, \sigma_1, \ldots, \sigma_m, syn, i_{in}, i_{out}, e)$ (TSNPS)<br>2. Rules: $E/a^c \to a^p, c \geq 1, c \geq p \geq 0$<br>3. Time mapping: $e : R \to N, R = R_1 \cup \ldots \cup R_m$<br>(i.e., execution time associated with each rule)<br>4. TFSNPS (TSNPS generating same result<br>without time mapping) | Integer factorization<br>SUBSET SUM |
| 2. SNPSPCR | 1. $\Pi = (O, \sigma_1, \ldots, \sigma_n, syn, in, out)$<br>2. $E/a^c \to a^p; d, c \geq p, p \geq 0, c \geq 1$<br>3. Arbitrary large inactive neurons is activated<br>(in exponential number) in polynomial time | QSAT, Q3SAT, SAT, SUBSET SUM |
| 3. NLPPS | 1. $\Pi = (O, \sigma_1, \ldots, \sigma_n, syn, i_{out})$<br>2. $\sigma_i = (Q_i, s_{i,0}, w_{i,0}, R_i)$,<br>$Q_i$ (finite set of possible states of $\sigma_i$)<br>$s_{i,0}$ ( initial state of $\sigma_i$)<br>$w_{i,0} \in O^*$ (initial multiset object in $\sigma_i$)<br>$R_i : sw \to^p s^{'} xy_{go}z_{out}, 0 \leq p \leq 1$,<br>($p$ probability of the applied rule) | Maximal Independent Set Selection |
| 4. IMSNPS | 1. $\Pi = (O, \sigma_1, \ldots, \sigma_n, syn, in, out)$<br>2. $E/a^c \to a^p, E \neq a^c, c \geq 1, d \geq 0, p \geq 1$<br>3. $a^c \to a^p; d(c \geq 1, d \geq 0, p \geq 1)$<br>4. $a^s \to \lambda$<br>5. Priority: (2) > (3) > (4)<br>6. Multiple output neurons | Directed HPP |
| 5. SNPSBR | 1. $\Pi = (O, \Sigma, H, syn, R, in, out)$<br>2. $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ (initial neurons)<br>3. $H$ (labels of neurons)<br>4. Budding rule: $x[]_i \to y[]_i, x \in \{(k,i), (i,k), \lambda\}$,<br>$y \in \{(i,j), (j,i), \lambda\}, i, j, k \in H, i \neq k, i \neq j$.<br>5. Spiking rule: $[E/a^c \to a^p; d]_i, i \in H$ | SAT |
| 6. SNPSNDB | 1. $\Pi = (O, H, syn, n_1, \ldots, n_m, R, in, out)$<br>2. $H$(labels), $n_i$ ( initial number of spikes in $\sigma_i$)<br>3. Spiking: $[E/a^c \to a^p; d]_i, i \in H$<br>$c \geq 1, p \geq 0, d \geq 0, c \geq p$<br>4. Neuron division: $[E]_i \to []_j \| []_k; i, j, k \in H$<br>5. Budding rule: $[E]_i \to []_i/[]_j, i, j \in H$ | SAT |
| 7. SNPSSP | 1. $\Pi = (O, \sigma_i, \ldots, \sigma_m, syn, out)$<br>2. Spiking rule: $E/a^c \to a, c \geq 1$<br>3. Plasticity rule: $E/a^c \to \alpha k(i, N), c \geq 1, \alpha \in \{+, -, \pm, \mp\}$<br>$k \geq 1, N \subseteq \{1, 2, \ldots, m\} \setminus \{i\}$ | SUBSET SUM |
| 8. OSNPS<br>(combination of<br>$H$ ESNPS) | 1. $\Pi = (O, \sigma_1, \ldots, \sigma_{m+2}, syn, I_0)$ (ESNPS)<br>2. $\sigma_i = (1, R_i, P_i), R_i = \{r_i^1, r_i^2\}, r_i^1 = \{a \to a\}, r_i^2 = \{a \to \lambda\}$<br>3. $\sigma_{m+1} = \sigma_{m+2} = (1, \{a \to a\})$<br>4. $syn = \{(i,j) \| i = m+2 \wedge 1 \leq j \leq m+1) \vee (i = m+1 \wedge j = m+2)\}$ | Knapsack problem<br>TSP |

**Figure 5.** List of SNPS models solving computationally hard problems.

**Remark 2.** *(1) The application of SNPS and their variants in solving computationally hard problems has been very limited. So, it will be more interesting to observe how other variants of SNPS models, where the properties of neuroscience have been incorporated, can be useful in solving these problems;*

*(2) Construction of other approximation algorithms based on different variants of SNPS can be a future direction of research;*

*(3) It is important to note that, while solving these problems, only the time and space resources are taken into account. However, no investigation has been done towards the number of neurons used to solve these*

*problems—i.e., investigation into the description of complexity measures. Constructing new methods to solve computationally hard problems with optimal number of neurons is worth investigating in future.*

In the next section, we discuss the use of SNPS models and their variants in performing arithmetic operations and logical operations. We also explore the use of these models in hardware implementations.

### *3.3. Performing Arithmetic and Logical Operations and Hardware Implementation*

One of the major motivations for studying arithmetic and logical operations using SNPS has been the designing of the arithmetic logic unit of CPU under the framework of SNPS which can be useful in constructing novel digital circuits/chips. Along with the advancement of technologies, it has become imperative to construct efficient hardware which can perform complex tasks. SNPS models have some very useful features, such as parallel and distributive architecture, non-determinism, etc., and these features help the models to perform millions of computations very efficiently with minimum time and space resources. These models also can perform basic arithmetic and logic operations which make SNPS an important candidate for designing of CPU.

SNPS models are powerful and are Turing complete and can perform many tasks efficiently. So, these models are suitable in performing arithmetic and logic operations. In 2009, Gutiérrez-Naranjo and A. Leporati introduced a method which can perform basic arithmetic operations, such as addition, subtraction, comparison and multiplication by a fixed facto,r and it was performed by SNPS where the inputs to the model are integers represented in the form of binary strings [45]. Finally, these strings are encoded in the form of appropriate sequence of spikes. The output neurons also can give output in binary form—i.e., in 0 and 1 depending on the non-spiking and spiking of the output neuron. Subsequently, Zhang et al. introduced a method based on SNPS to perform the multiplication of two arbitrary natural numbers in [80]. Furthermore, they introduced a method of addition of $n$ natural numbers using only one input neuron and given the length of binary bits, a family of spiking neural P systems was constructed such that the multiplication of two arbitrary natural numbers can be performed by each system. In [36], signed integer arithmetic operations were performed using SNPS and similarly as before the input and output are encoded in the form of sequence of spikes. Many variants of SNPS have also been used for performing logical and arithmetic operations. In [37], a method based on the SNPS with anti-spikes were introduced by Peng et al. to perform balanced ternary logic (AND, OR and NOT) and arithmetic operations (addition, subtraction) where balanced ternary digits are encoded by the anti-spikes. In [81], SNPS with asynchronous parallelism were used to perform logical operations, such as NOT, OR, AND and EX-OR. In SNPS models, the rules are applied in a maximal parallel manner. However, asynchronous parallelism is different from maximal parallelism and in this case all the applicable firing rules are applied in one computation step. Additionally, the steps of computations can be divided into sequential and parallel steps. The logical operations can be performed using constant number of sequential and parallel steps and using a constant number of neurons. Furthermore, the addition of $k$ binary numbers of $m$ bits can be performed using either $O(km)$ sequential steps or $O(m)$ parallel steps where the number of neurons required is $O(m)$. Similarly, the multiplication of two $m$ bit binary numbers can be performed using $O(m^2)$ number of neurons in $O(m^2)$ sequential steps or $O(m)$ parallel steps. Another variant of SNPS was introduced in 2014 by Luan et al., which can perform the logical operations and it is called SNPC (spiking neural P systems with chain structures) [38]. In this model, the concepts of discrete Morse theory are integrated into the SNPS model where a chain structure is introduced in SNPS by discrete gradient vector path and this model is simple and has stronger parallelism than the SNPS. It also avoids the random selection of membranes present in the computation process of SNPS models, which further improves the operational efficiency of these models. SNPC were also used to perform the logical operations AND, OR and NOT. One of the main features of SNPS has been the encoding of the information in the form of interval of two spikes. In [47], another well-known variant of SNPS—i.e., time-free

SNPS—were introduced by Liu et al. in 2015 to perform arithmetic operations, such as addition, subtraction, multiplication and division. In time-free SNPS, using a time mapping an execution time is associated with each rule of the system and if the computational results of these systems are independent of the execution times in the rules, then these systems are called time-free SNPS. One of the major advantages of using time-free SNPS is that the arithmetic operations can be performed with a lower number of neurons. Another variant of SNPS—i.e., extended SNPS—have excitatory and inhibitory astrocytes (ESNPSEIA) where the concepts of astrocytes and excitatory/inhibitory properties in between the neurons are incorporated into SNPS model [39]. Moreover, NAND-gates and discrete amplifiers were constructed using a restricted variant of this model.

In the previous models, the encoding to binary digits is done in a similar manner as in the digital circuits. However, in [46], Zeng et al. introduced a method in 2012 where the input to the systems is actually the time difference between the two spikes received by the input neuron and similarly the output of the system is considered as the time difference between the spiking of the output neuron. Note that the spiking rules present in this model are rules without delays. Moreover, this model was used to perform basic arithmetic operations, such as addition, subtraction, multiplication and division.

Recently, an arithmetic calculator based on the SNPS model was proposed by Zhang et al. in [82]. Furthermore, the implementation of the information fusion was studied where a framework combining the SNPS models and Dempster-Shafer evidence theory was constructed in order to obtain BPA (basic probability of assignment) of an event.

Following the developments in constructing novel and efficient SNPS models for performing arithmetic and logical operations, investigating the hardware implementation of these models became an interesting branch of research. In 2016, Song et al. introduced a framework using the SNPS with astrocyte-like control to simulate the AND, OR, NOT, NOR, XOR and NAND gate operations in Boolean logic gates [43]. The most distinctive feature of this framework is that only one type of neuron with a spiking rule of the form $a^*/a \rightarrow a$ was used. This study is important because more complex boolean circuits can also be constructed from these logic gates. Moreover these gates can be used in constructing finite transducers. In 2016, hardware implementation of an unary SNN multiplier with dendritic delays was investigated by Díaz et al. in [40]. More specifically, an SNP multiplier with dendritic growth, delay and feedback was introduced and also implemented in a customized neuromorphic hardware—i.e., FPGA solutions based neuromorphic circuits. Additionally, one of the advantages of this multiplier is that it eliminates complex rules and the processing speed is higher compared to the multipliers based on SNNs. Additionally, in 2016, a decoder design in the framework of SNPS was proposed by Li, Huang and Xu [44]. Decoders are composed of logical circuits and logical and arithmetical operations are an integral part of logical circuits. In [44], a single input-output $n - 2^n, n \geq 2$ decoder was constructed in the framework of SNPS . Moreover, the MeCosim platform was used in order to obtain the result of decoding of $n$-bit binary sequence. Additionally, simulations of 3–8 decoders on the MeCosim platform were observed and the results infer that the decoders based on the SNPS are efficient. In 2017, a parallel multiplier was constructed using the SNPS with dendritic delays by Díaz, et al. [41]. It is important to note that the models discussed above have one common drawback—i.e., the processing of the inputs is sequential, which increases the processing time. The parallel multiplier discussed in [41] can multiply any two natural numbers with long digits in parallel. Moreover, the parallel multiplier has optimized sequential processors which can process multiple units in parallel and then the implementation of the parallel multiplier circuit in FPGA devices, such as spiking neural-network architecture for versatile applications (SNAVA) and field programmable gate array prototypes, also were investigated.

Until now, the main discussion was regarding the construction of CPUs in the framework of SNPS, which mainly deal with constructing theoretical arithmetic and logical circuits and can be integrated into an ALU (arithmetic logic unit). Recently, many researchers also tried to use this framework to construct MPUs (memory processing units) because of their ability to process the information faster and the storage capability. In 2017, Duchen et al. initiated a study regarding the designing of

MPUs (memory processing units) which was inspired from some features in neurons, such as neural processing of soma, dendritic behaviours, such as delays, feedback connections, etc., and astrocyte-like controls [42]. One of the main advantages of this model is that the addition and subtraction operations required to construct MPUs can be performed in the same unit while the basic neural memory cells remain unchanged.

In Figure 6, we presented a comparison of all SNPS models performing arithmetic and logical operations and also their hardware implementation.

| MODELS | PROPERTIES | ARITHMETIC, LOGICAL OPERATIONS & HARDWARE IMPLEMENTATION |
|---|---|---|
| 1.ASYNSNPS | 1. $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in, out)$<br>2. Spiking rule: $E/a^c \to a$ (no delay)<br>3. Forgetting rule: $a^s \to \lambda$<br>4. Rules are applicable sequentially and maximal parallel manner | 1. NOT, OR, AND and EX-OR<br>2. Addition of $k$ binary numbers of $m$ bits<br>Multiplication of two binary numbers of $m$ bits |
| 2. SNPC | 1. $\Pi = (O, \sigma_0, \sigma_1, \ldots, \sigma_m, syn, in, out)$<br>2. $\sigma_0 \prec \sigma_1 \prec \ldots \prec \sigma_m; \sigma_0$ (skin membrane)<br>3. Spiking rule: $E/a^c \to a^p; d$<br>4. Forgetting rule: $E^{'}/a^s \to \lambda, s \geq 1$<br>$E \cap E^{'} = \emptyset$ | 1. NOT, AND, OR |
| 3. SNPS | 1. $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in, out)$<br>2. Spiking rule: $E/a^c \to a; d$<br>3. Forgetting rule: $a^s \to \lambda, s \geq 1$ | 1. Addition, Subtraction, Checking equality, Multiplication, Two's complement<br>Parallel Multiplication (SNAVA FPGA prototype) |
| 4. SNPSAS | 1. $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in, out)$<br>2. $O = \{a, \overline{a}\}$<br>3. $E/b^c \to b^{'}; b, b^{'} \in \{a, \overline{a}\}, c \geq 1$<br>4. $b^s \to \lambda; b \in \{a, \overline{a}\}$ | 1. AND, OR, NOT<br>2. Addition, Subtraction |
| 5. SNPSDD | 1. SNPS + Dendrite feedback, delay and growth | 1. Serial multiplier ( SNAVA FPGA prototype) |
| 6. SNPSACL | 1. $\Pi = (O, \sigma_1, \ldots, \sigma_n, syn, in, out)$<br>2. Spiking rule: $E/a^c \to a; d$<br>3. Forgetting rule: $a^s \to \lambda$<br>4. Astrocyte: $\{ast_1, \ldots, ast_l\}$,<br>$ast_i = (syn_{ast_i}, t_i), syn_{ast_i} \subseteq syn$<br>(set of synapses controlled by $ast_i$<br>$t_i \geq 0$ (threshold of $ast_i$) | 1. AND, OR, NOT, NOR, XOR, NAND<br>(Boolean circuits with cascade connections of logic gates) |
| 7. ESNPSEIA | 1. $\Pi = (m, n, S, R, U)$<br>$m$ (number of neurons)<br>$n$ (number of astrocytes)<br>$S$ (initial configuration)<br>2. Rules: $(i, E/a^k \to P)$<br>$i \in [1 \ldots m]$, $P$ (set of productions of the form $(l, w), l \in [1 \ldots m+n], w \in N$)<br>3. $U$ (finite set of rules of the form $(r, p, q, h, h^{'}, f, f^{'}, f^{''})$) | 1. NAND-gates, discrete amplifiers |
| 8. TFSNPS | 1. $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in, out)$<br>2. Spiking rules: $E/a^c \to a^p$<br>$c \geq 1, c \geq p \geq 0$<br>3. Time-mapping: $e : R \to N$<br>$R = R_1 \cup R_2 \cup \ldots \cup R_m$ | 1. Addition, Subtraction, Multiplication and Division |

**Figure 6.** List of SNPS models performing arithmetic and logical operations and their hardware implementation.

**Remark 3.** *(1) Constructing new SNPS models by incorporating features of neuroscience and performing arithmetic and logical operations using these models can be interesting. Moreover, comparing their performances and investigating the implementations of these models can be a research direction for the future .*

*(2) It was observed in [46] that the encoding of time is done by difference between the spiking of one neuron. Finding new encoding techniques and their performance comparison, studying their advantages and disadvantages can also be interesting.*

In the next section, we discuss the use of SNPS in solving problems in pattern recognition.

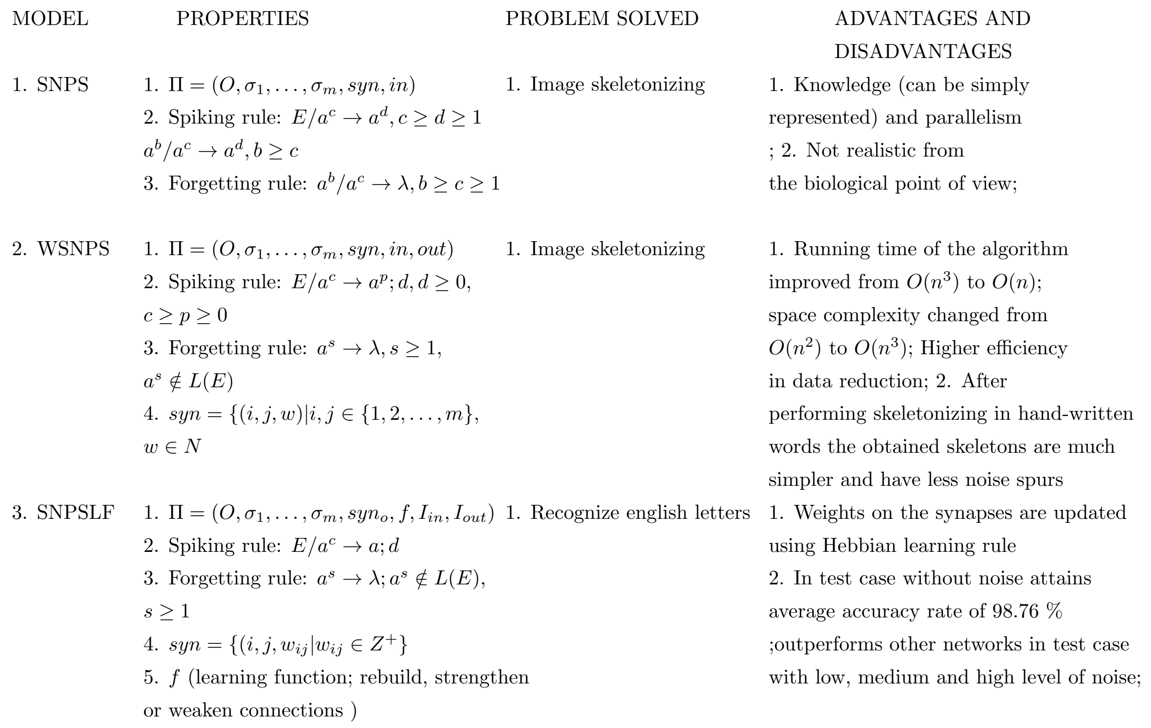
### 3.4. Pattern Recognition

The process of extracting skeletons from a digital binary picture is known as Skeletonizing and the pre-process in pattern recognition algorithms is known as Skeletonization. In 2013, Díaz-Pernil et al. introduced a method to solve the skeletonization problem in the framework of SNPS [83]. More specifically, it is possible to implement classical parallel algorithms, such as Guo and Hall in the framework of SNPS. Moreover, a parallel software was created based on the SNPS device and was implemented in GPU [83]. Some of the advantages of the proposed method are that knowledge can be simply represented and parallelism, which are helpful while dealing with the digital images. This model also has some drawbacks, such as the fact that it is not realistic from the biological point of view because the spike flows between two neurons always take same amount of time because no delay is associated with the rules and also the cost of synchronization in this model is very high. Another parallel image skeletonizing method [32] was introduced by Song et al. in 2019. It is based on a different variant of SNPS than the previous models—i.e., SNPS with weights. In this model of SNPS, a weight is associated with the synapses. SNPS with weights were used to construct the Zhang–Suen algorithm where the skeletonizing of the images is done in a parallel manner. Some of the advantages of this method are: (1) The running time of the Zhang–Suen algorithm, improved from $O(n^3)$ to $O(n)$, while the disadvantage of this method is that the space complexity is changed to $O(n^3)$ from $O(n^2)$ where $n = max\{p, q\}$, and the skeletonizing algorithm is applied to a binary image of size $p \times q$ pixels; (2) Higher efficiency in data reduction; (3) After performing skeletonizing in hand-written words the obtained skeletons are much simpler and have less noise spurs.

Fingerprint recognition is a challenging problem in pattern recognition. In 2017, Ma et al. introduced a framework based on SNPS to solve this problem [34]. More specifically, a double-layer self-organized SNPS with anti-spikes model was introduced to perform this task and the working of this model different from the working of other models mentioned in this section. It is capable of adaptively creating and deleting neurons present in the different layers. Moreover, the fingerprint recognition task is performed by the spike trains emitting from the output neurons and the results of the experiments infer that this model can have comparable performance.

In [33], a new variant of SNPS was introduced to solve another well-known pattern recognition problem—i.e., recognition of English letters by Song et al. The SNPS model used to solve this problem is called SNPS with Hebbian learning function. It is well-known that, in neural networks, weights are associated with the synapses and the weights are updated using machine learning algorithms. Moreover, this idea has been used to solve many problems. In this case, the SNPS models construct a network where the weights on the synapses are updated using Hebbian learning rulse and this model was used to recognize English letters. Furthermore, the performance of this algorithm was compared with other well-known neural networks, such as back propagation neural networks and probabilistic neural networks and, from the experiments it is clear that the proposed method in the test case without noise attains an average accuracy rate of 98.76 % while it outperforms other networks in test case with low, medium and high level of noise.

In Figure 7, we list all the SNPS models solving pattern recognition problems and also study the comparison of these models.



| MODEL | PROPERTIES | PROBLEM SOLVED | ADVANTAGES AND DISADVANTAGES |
|---|---|---|---|
| 1. SNPS | 1. $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in)$ 2. Spiking rule: $E/a^c \to a^d, c \geq d \geq 1$ $a^b/a^c \to a^d, b \geq c$ 3. Forgetting rule: $a^b/a^c \to \lambda, b \geq c \geq 1$ | 1. Image skeletonizing | 1. Knowledge (can be simply represented) and parallelism ; 2. Not realistic from the biological point of view; |
| 2. WSNPS | 1. $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in, out)$ 2. Spiking rule: $E/a^c \to a^p; d, d \geq 0,$ $c \geq p \geq 0$ 3. Forgetting rule: $a^s \to \lambda, s \geq 1,$ $a^s \notin L(E)$ 4. $syn = \{(i, j, w) \vert i, j \in \{1, 2, \ldots, m\},$ $w \in N$ | 1. Image skeletonizing | 1. Running time of the algorithm improved from $O(n^3)$ to $O(n)$; space complexity changed from $O(n^2)$ to $O(n^3)$; Higher efficiency in data reduction; 2. After performing skeletonizing in hand-written words the obtained skeletons are much simpler and have less noise spurs |
| 3. SNPSLF | 1. $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn_o, f, I_{in}, I_{out})$ 2. Spiking rule: $E/a^c \to a; d$ 3. Forgetting rule: $a^s \to \lambda; a^s \notin L(E),$ $s \geq 1$ 4. $syn = \{(i, j, w_{ij} \vert w_{ij} \in Z^+\}$ 5. $f$ (learning function; rebuild, strengthen or weaken connections ) | 1. Recognize english letters | 1. Weights on the synapses are updated using Hebbian learning rule 2. In test case without noise attains average accuracy rate of 98.76 % ;outperforms other networks in test case with low, medium and high level of noise; |

**Figure 7.** List of SNPS solving problems in pattern recognition.

**Remark 4.** *(1) The application of SNPS models and its variants in solving pattern recognition problems has been very limited. So, construction of new SNPS models inspired from the properties of neuroscience and their use in solving these problems can be a research topic for the future.*

*(2) The model introduced in [33] was trained using the Hebbian learning algorithm. It remains to be investigated whether Hebbian learning strategies can improve the performance of this algorithm in the framework of SNPS. Moreover the investigations regarding the use of the SNPS models along with the Hebbian learning strategy in solving other problems in pattern recognition can be interesting.*

Different variants of SNPS have also been used to solve problems in many other areas of application than the areas mentioned above. In the next section we give a summary of such applications.

### *3.5. Other Applications*

The SNPS models also have been used to solve many problems, such as the identification of nuclear export signal [35], computing morphisms [65,66], intrusion detection [84], solving deadlock resolution problems [85], programmable stage controller stage programming [68] and $\mu$-fluidic biochip design [67]. In this section, we briefly discuss the use of SNPS in solving these problems.

(1) Identifying NES (nuclear export signal) is a very challenging problem in computational biology. In [35], SNPS with Hebbian learning strategy were introduced by Chen et al. to solve this problem and the experimental results infer that this model is very promising while identifying nuclear export signals, since it has very good precision rate. The system can be divided into two modules—i.e., input module and predict module. The topology of the input module remains unchanged but the topology of the predict module changes over time, where for each passing unit of time the weights on the synapses change.

(2) SNPS models also can be used to compute morphisms [65] and $k$-block morphisms [66]. It is well-known that SNPS models can be considered as transducers and this idea was used to compute 2-block morphisms in [65] by Păun, Pérez-Jímenez and Rozenberg. However, non-erasing and non-length preserving morphisms cannot be computed using these

models. This work was further extended in [66] by Nishida in 2011 where SNPS were used to compute $k$-block morphisms where $k \geq 2$.

(3) We have already discussed the use of tFRSNPS (trapezoidal fuzzy reasoning spiking neural P systems) models in fault diagnoses of power systems. These models can also be used very effectively in intrusion detection [84]. In 2014 , Idowu, Chandren and Othman introduced a network intrusion prediction model using tFRSNPS [84]. The experiments performed with this model on the KDD Cup benchmark dataset infer that this model has very high detection rates and, at the same time, has very low false alarm rate while performing BFA (Brute force attack).

(4) SNPS models have also been used to solve problems in operating and database systems. In [85], Pang et al. proposed a method based on SNPS for solving the Deadlock resolution problem in 2019. In [85], SNPS were used for the revocation of the deadlock processes. Moreover, in comparison with exhaustive method and time cost method, (i.e., traditional deadlock methods) the method proposed in [85] has significantly improved the time complexity while performing the deadlock revocation process. Additionally, this method improves the deadlock resolution efficiency. It is also important to note that the variant of SNPS used to solve this problem is called SNPS for deadlock release (DRSNPS). The structure of this model is different from the tradition SNPS models.

(5) Along with the above models, the scope of the application of the membrane computing models has been expanded by many researchers in recent years with applications in areas such as $\mu$-fluidic biochip design [67] and programming for PLC (programmable logic controller) [68]. The first one is an attempt in hardware implementation of the SNPS models while the second one is an attempt to write programming based on the SNPS models.

## 4. Conclusions and Future Research Lines

In this paper, we provided an updated and comprehensive survey on applications of spiking neural P systems. We also provided a comparative study of these models and studied their advantages and disadvantages. Moreover, we studied some implementation of these models in hardware and software. Finally, we discussed some possible extensions of these models which can be investigated further. Next, we discuss some more ideas which can expand the scope of applications of these models.

1. In this work we discussed different variants of FRSNPS models and their applications in fault diagnosis. It is important to note that these models work efficiently for small scale networks. Along with the implementation of the FRSNPS models in fault diagnoses in large scale power networks, it is imperative to investigate whether these models are effective, accurate and reliable while solving the fault diagnosis problem in micro and smart grids. Additionally, the resources—i.e., time and space required by these model—also need to be investigated. We also have discussed the use of learning mechanisms in solving the problems in fault diagnosis. Until now, only few models have been introduced with the features of machine learning. This area can be further extended by constructing new SNPS models and incorporating learning mechanisms in these models and then using these models for the identification of faults in power systems. Moreover, one could then study their speed and accuracy while performing this task. Additionally, a natural extension of these works will be the construction of fault diagnosis models combining SNPS and metaheuristic algorithms.

   Another important aspect which is not properly investigated in the literature is the descriptional complexity measures of the models used for fault diagnosis. The optimized use of the neurons is a topic of future research directions.
2. We have already discussed the use of unsupervised machine learning algorithms in the framework of SNPS to solve pattern recognition problems. Construction of new machine learning algorithms is a very new and interesting research area. However, until now no known progress has been made towards the construction of supervised learning algorithms in the framework of SNPS for

solving pattern recognition problems. This can be an important direction to expand the scope of machine learning algorithms in SNPS and also applications of membrane computing models.

3. We have discussed the OSNPS model and its variants in solving knapsack problems, TSP and their use in fault diagnosis. The use of this model in solving other computationally hard combinatorial optimization problems, as well as their use in solving problems in real-life applications, can be a direction of future research.

**Author Contributions:** Conceptualization of this research was done by G.Z. and P.P.; Investigations were done by P.P., S.F., T.W. and H.R.; Writing and preparation of the draft version were done by P.P. and G.Z.; Writing and editing and providing suggestions for improving the contents were done by G.Z., P.P., S.F., T.W. and H.R. Final submission version was prepared with the involvement of all the authors. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Acknowledgments:** This work was supported by the project of State Grid Corporation of China (No. 521997180016), by the National Natural Science Foundation of China (61972324, 61672437, 61702428), by New Generation Artificial Intelligence Science and Technology Major Project of Sichuan Province (2018GZDZX0043), Beijing Advanced Innovation Center for Intelligent Robots and Systems (2019IRS14) and Artificial Intelligence Key Laboratory of Sichuan Province (2019RYJ06).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Moore, G.E. Cramming more components onto integrated circuits. *Electronics* **1965**, *38*, 114. [CrossRef]
2. Adleman, L.M. Molecular Computation of Solutions to Combinatorial Problems. *Science* **1994**, *266*, 1021–1024. [CrossRef] [PubMed]
3. Head, T. Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviours. *Bull. Math. Biol.* **1987**, *49*, 737–759. [CrossRef]
4. Păun, G.; Rozenberg, G.; Salomaa, A. *DNA Computing: New Computing Paradigms (Texts in Theoretical Computer Science. An EATCS Series)*; Springer: Berlin/Heidelberg, Germany, 2006.
5. Păun, G. Computing with Membranes. *J. Comput. Syst. Sci.* **2000**, *61*, 108–143. [CrossRef]
6. Ionescu, M.; Păun, G.; Yokomori, T. Spiking neural P systems. *Fundam. Informaticae* **2006**, *71*, 279–308.
7. Mozafari, M.; Ganjtabesh, M.; Nowzari-Dalini, A.; Masquelier, T. SpykeTorch: Efficient Simulation of Convolutional Spiking Neural Networks With at Most One Spike per Neuron. *Front. Neurosci.* **2019**, *13*. [CrossRef]
8. Cavaliere, M.; Egecioglu, O.; Ibarra, O.H.; Woodworth, S.; Ionescu, M.; Păun, G. *Asynchronous Spiking Neural P Systems: Decidability and Undecidability*; Garzon, M.H., Yan, H., Eds.; Springer: Berlin, Germany, 2008, pp. 246–255.
9. Pan, L.; Wang, J.; Hoogeboom, H. Spiking neural P systems with astrocytes. *Neural Comput.* **2012**, *24*, 805-825. [CrossRef]
10. Su, Y.; Wu, T.; Xu, F.; Păun, A. Spiking Neural P Systems with Rules on Synapses Working in Sum Spikes Consumption Strategy. *Fundam. Informaticae* **2017**, *156*, 187–208. [CrossRef]
11. Pan, L.; Păun, G.; Zhang, G.; Neri, F. Spiking neural P systems with communication on request. *Int. J. Neural Syst.* **2017**, *27*, 1750042. [CrossRef]
12. Song, T.; Pan, L. Spiking neural P systems with request rules. *Neurocomputing* **2016**, *193*, 193–200. [CrossRef]
13. Cabarle, F.G.C.; Adorna, H.N.; Jiang, M.; Zeng, X. Spiking Neural P Systems With Scheduled Synapses. *IEEE Trans. Nanobiosci.* **2017**, *16*, 792–801. [CrossRef] [PubMed]
14. Cabarle, F.G.C.; Adorna, H.N.; Pérez-Jiménez, M.J.; Song, T. Spiking neural P systems with structural plasticity. *Neural Comput. Appl.* **2015**, *26*, 1905–1917. [CrossRef]
15. Pan, L.; Zeng, X.; Zhang, X.; Jiang, Y. Spiking Neural P Systems with Weighted Synapses. *Neural Process. Lett.* **2012**, *35*, 13–27. [CrossRef]
16. Song, T.; Wang, X. Homogenous Spiking Neural P Systems with Inhibitory Synapses. *Neural Process. Lett.* **2015**, *42*, 199–214. [CrossRef]
17. Pan, L.; Păun, G. Spiking neural P systems with anti-spikes. *Int. J. Comput. Commun. Control.* **2009**, *4*, 273–282. [CrossRef]

18. Zhang, G.; Rong, H.; Neri, F.; Pérez-Jiménez, M.J. An optimization spiking neural P system for approximately solving combinatorial optimization problems. *Int. J. Neural Syst.* **2014**, *24*, 1440006. [CrossRef]
19. Wang, T.; Zhang, G.; Zhao, J.; He, Z.; Wang, J.; Pérez- Jiménez, M.J. Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural P systems. *IEEE Trans. Power Syst.* **2015**, *30*, 1182–1194. [CrossRef]
20. Peng, H.; Wang, J.; Peŕez-Jimeńez, M.J.; Wang, H.; Shao, J.; Wang, T. Fuzzy reasoning spiking neural P system for fault diagnosis. *Inf. Sci.* **2013**, *235*, 106–116. [CrossRef]
21. Xiong, G.; Shi, D.; Zhu, L.; Duan, X. A new approach to fault diagnosis of power systems using fuzzy reasoning spiking neural P systems. *Math. Probl. Eng.* **2013**, *2013*, 13. [CrossRef]
22. Tu, M.; Wang, J.; Peng, H.; Shi, P. Application of adaptive fuzzy spiking neural P systems in fault diagnosis of power systems, *Chin. J. Electron.* **2014**, *23*, 87–92.
23. Wang, J.; Pérez-Jiménez, J.M.; Peng, H.; Shi, P.; Tu, M. A fault diagnosis method of power systems based on an improved adaptive fuzzy spiking neural P systems and PSO algorithms. *Chin. J. Electron.* **2016**, *25*, 320–327. [CrossRef]
24. Wang, T.; Zhang, G.; Pérez-Jiménez, J.M.; Cheng, J. Weighted fuzzy reasoning spiking neural P systems: Application to fault diagnosis in traction power supply systems of high-speed railways. *J. Comput. Theor. Nanosci.* **2015**, *12*, 1103–1114. [CrossRef]
25. He, Y.; Wang, T.; Huang, K.; Zhang, G.; Pérez-Jiménez, M.J. Fault diagnosis of metro traction power systems using a modified fuzzy reasoning spiking neural P system. *Rom. J. Inf. Sci. Technol.* **2015**, *18*, 256–272.
26. Wang, T.; Zeng, S.; Zhang, G.; Pérez-Jiménez, M.J.; Wang, J. Fault Section Estimation of Power Systems with Optimization Spiking Neural P Systems. *Rom. J. Inf. Sci. Technol.* **2015**, *18*, 240–255.
27. Tao, C.; Yu, W.; Wang, J.; Peng, H.; Chen, K.; Ming, J. Fault diagnosis of power systems based on triangular fuzzy spiking neural P systems. *Bio-Inspired Comput. Theor. Appl.* **2017**, *618*, 385–398.
28. Rong, H.; Yi, K.; Zhang, G.; Dong, J.; Paul, P.; Huang, Z. Automatic implementation of fuzzy reasoning spiking neural P systems for diagnosing faults in complex power systems. *Complexity* **2019**, *2019*, 1–16. [CrossRef]
29. Liu, W.; Wang, T.; Zang, T.; Huang, Z.; Wang, J.; Huang, T.; Wei, X.; Li, C. A Fault Diagnosis Method for Power Transmission Networks Based on Spiking Neural P Systems with Self-Updating Rules considering Biological Apoptosis Mechanism. *Complexity* **2020**, *2020*. [CrossRef]
30. Sun, Z.; Wang, Q.; Wei, Z. Fault location of distribution network with distributed generations using electrical synaptic transmission-based spiking neural P systems. *Int. J. Parallel Emergent Distrib. Syst.* **2019**. [CrossRef]
31. Yahya, Y.; Qian, A.; Yahya, A. Power Transformer Fault Diagnosis Using Fuzzy Reasoning Spiking Neural P Systems. *J. Intell. Learn. Syst. Appl.* **2016**, *8*, 77–91. [CrossRef]
32. Song, T.; Pang, S.; Hao, S.; Rodríguez-Paton, A.; Zheng, P. A Parallel Image Skeletonizing Method Using Spiking Neural P Systems with Weights. *Neural Process. Lett.* **2019**, *50*, 1485–1502. [CrossRef]
33. Song, T.; Pan, L.; Wu, T.; Zheng, P.; Wong, M.L.D.; Rodríguez-Patón, A. Spiking Neural P Systems With Learning Functions. *IEEE Trans. Nanobiosci.* **2019**, *18*, 176–190. [CrossRef] [PubMed]
34. Ma, T.; Hao, S.; Wang, X.; Rodríguez-Patón, A.; Wang, S.; Song, T. Double Layers Self-Organized Spiking Neural P Systems with Anti-spikes for Fingerprint Recognition. *IEEE Access* **2017**, *7*, 177562–177570. [CrossRef]
35. Chen, Z.; Zhang, P.; Wang, X.; Shi, X.; Wu, T.; Zheng, P. A computational approach for nuclear export signals identification using spiking neural P systems. *Neural Comput. Appl.* **2018**, *29*, 695–705. [CrossRef]
36. Gao, X.; Chen, H.Z. Signed Integer Arithmetic on Spiking Neural P System. *Appl. Mech. Mater.* **2010**, *20–23*, 779–784. [CrossRef]
37. Peng, X.W.; Fan, X.P.; Liu, J.X. Performing Balanced Ternary Logic and Arithmetic Operations with Spiking Neural P Systems with Anti- Spikes. *Adv. Mater. Res.* **2012**, *505*, 378–385. [CrossRef]
38. Luan, J.; Liu, X.Y. Logic Operation in Spiking Neural P System with Chain Structure, Frontier and Future Development of Information Technology. In *Medicine and Education, Lecture Notes in Electrical Engineering*; Springer: New York, NY, USA, 2014; p. 269.
39. Binder, A.; Freund, R.; Oswald, M.; Vock, L. Extended Spiking Neural P systems with Excitatory and Inhibitory Astrocytes. In Proceedings of the Fifth Brainstorming Week on Membrane Computing, Sevilla, Spain, 29 January–2 February 2007.

40. Díaz, C.; Sanchez, G.; Duchen, G.; Nakano, M.; Perez, H. An efficient hardware implementation of a novel unary Spiking Neural Network multiplier with variable dendritic delays. *Neurocomputing* **2016**, *189*, 130–134. [CrossRef]
41. Díaz, C.; Frias, T.; Sanchez, G.; Perez, H.; Toscano, K.; Duchen, G. A novel parallel multiplier using spiking neural P systems with dendritic delays. *Neurocomputing* **2017**, *239*, 113–121. [CrossRef]
42. Duchen, G.; Diaz, C.; Sanchez, G.; Perez, H. First steps toward memory processor unit architecture based on SN P systems. *Electron. Lett.* **2017**, *53*, 384–385. [CrossRef]
43. Song, T.; Zheng, P.; Wong, M.L.D.; Wang, X. Design of logic gates using spiking neural P systems with homogeneous neurons and astrocytes-like control. *Inf. Sci.* **2016**, *372*, 380–391. [CrossRef]
44. Li, J.; Huang, Y.; Xu, J. Decoder Design Based on SpikingNeural P Systems. *IEEE Trans. Nanobiosci.* **2016**, *15*, 639–644. [CrossRef]
45. Gutíerrez-Naranjo, M.A.; Leporati, A. Performing arithmetic operations with spiking neural P systems. In Proceedings of the 7th Brainstorming Week Membrane Computing, Sevilla, Spain, 27 February–2 April 2009; pp. 181–198.
46. Zeng, X.; Song, T.; Zhang, X.; Pan, L. Performing four basic arithmetic operations with spiking neural P systems. *IEEE Trans. Nanobiosci.* **2012**, *11*, 366–374. [CrossRef] [PubMed]
47. Liu, X.; Li, Z.; Liu, J.; Liu, L.; Zeng, X. Implementation of arithmetic operations with time-free spiking neural P systems. *IEEE Trans. Nanobiosci.* **2015**, *14*, 617–624. [CrossRef] [PubMed]
48. Ishdorj, T.O.; Leporati, A.; Pan, L.; Zeng, X.; Zhang, X. Deterministic solutions to QSAT and Q3SAT by spiking neural P systems with pre-computed resources. *Theor. Comput. Sci.* **2010**, *411*, 2345–2358. [CrossRef]
49. Chen, H.; Ionescu, M.; Ishdorj, T.-O. On the efficiency of spiking neural P systems. In *Fourth Brainstorming Week on Membrane Computing*; Gutiérrez-Naranjo, M.A., Păun, G., Riscos-Núnez, A., Romero-Campero, F.J., Eds.; Sevilla University: Sevilla, Spain, 2006; pp. 195–206.
50. Ishdorj, T.-O.; Leporati, A. Uniform solutions to SAT and 3-SAT by spiking neural P systems with pre-computed resources. *Nat. Comput.* **2008**, *7*, 519–534. [CrossRef]
51. Leporati, A.; Gutiérrez-Naranjo, M.A. Solving Subset Sum by spiking neural P systems with pre-computed resources. *Fundam. Informaticae* **2008**, *87*, 61–77.
52. Leporati, A.; Mauri, G.; Zandron, C.; Păun, G.; Pérez-Jiménez, M.J. Uniform solutions to SAT and Subset Sum by spiking neural P systems. *Nat. Comput.* **2009**, *8*, 681–702. [CrossRef]
53. Leporati, A.; Zandron, C.; Ferretti, C.; Mauri, G. Solving numerical NP-complete problem with spiking neural P systems. In *Membrane Computing, International Workshop, WMC8, Selected and Invited Papers*; Eleftherakis, G., Kefalas, P., Păun, G., Rozenberg, G., Salomaa, A., Eds.; Springer: New York, NY, USA, 2007; pp. 336–352.
54. Song, T.; Luo, L.; He, J.; Chen, Z.; Zhang, K. Solving Subset Sum Problems by Time-free Spiking Neural P Systems. *Appl. Math. Inf. Sci.* **2014**, *8*, 327–332. [CrossRef]
55. Ishdorj, T.-O.; Leporati, A.; Pan, L.; Wang, J. *Solving NP-Complete Problems by Spiking Neural P Systems with Budding Rules*; Păun, G., Ed.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 335–353.
56. Pan, L.; Păun, G.; Pérez-Jiménez, M.J. Spiking neural P systems with neuron division and budding. In *Seventh Brainstorming Week on Membrane Computing*; RGNC Report 01/2009; Research Group on Natural Computing; Gutiérrez-Escudero, R., Ed.; Sevilla University: Sevilla, Spain, 2009; pp. 151–167.
57. Pan, L.; Păun, G.; Pérez-Jiménez, M. Spiking neural P systems with neuron division and budding. *Sci. China Inf. Sci.* **2011**, *54*, 1596–1607. [CrossRef]
58. Cabarle, F.G.C.; Hernandez, N.H.S.; Martínez-del-Amor, M.A. Spiking Neural P Systems with Structural Plasticity: Attacking the Subset Sum Problem. In *Membrane Computing*; CMC 2015, LNCS 9504; Rozenberg, G., Salomaa, A., Sempere, J., Zandron, C., Eds.; Springer: New York, NY, USA, 2015.
59. Xue, J.; Liu, X. Solving Directed Hamilton Path Problem in Parallel by Improved SN P System. In *ICPCA-SWS 2012, LNCS 7719*; Zu, Q., Hu, B., Elçi, A., Eds.; Springer: New York, NY, USA, 2012; pp. 689–696.
60. Liu, X.; Li, Z.; Suo, J.; Liu, J.; Min, X. A uniform solution to integer factorization using time-free spiking neural P system. *Neural Comput. Appl.* **2015**, *26*, 1241–1247. [CrossRef]
61. Qi, F.; Liu, M. Optimization Spiking Neural P System for Solving TSP, ICST Institute for Computer Sciences. In *Social Informatics and Telecommunications Engineering 2018 X*; MLICOM 2017, Part II, LNICST 2018; Springer: New York, NY, USA, 2018; Volume 227, pp. 668–676.
62. Pan, L.; Zeng, X.; Zhang, X. Time-free Spiking Neural P Systems. *Neural Comput.* **2011**, *23*, 1320–1342. [CrossRef]

63. Zhang, G.; Pérez-Jiménez, M.J.; Gheorghe, M. *Real-life Applications with Membrane Computing, Emergence, Complexity and Computation Book Series (ECC)*; Springer: New York, NY, USA, 2017; Volume 25.
64. Xu, L.; Jeavons, P. Simple Neural-Like P Systems for Maximal Independent Set Selection. *Neural Comput.* **2013**, *25*, 1642–1659. [CrossRef] [PubMed]
65. Păun, G.; Pérez-Jiménez, M.J.; Rozenberg, G. Computing morphisms by spiking neural P systems. *Int. J. Found. Comput. Sci.* **2007**, *18*, 1371–1382. [CrossRef]
66. Nishida, T.Y. Computing *k*-block morphisms by spiking neural P systems. *Fundam. Informaticae* **2011**, *111*, 453–464. [CrossRef]
67. Ishdorj, T.-O.; Ochirbat, O.; Naimannaran, C. A $\mu$-fluidic Biochip Design for Spiking Neural P Systems. *Int. J. Unconv. Comput.* **2020**, *15*, 59–82.
68. Chen, K.; Wang, J.; Sun, Z.; Luo, J.; Liu, T. Programmable Logic Controller Stage Programming Using Spiking Neural P Systems. *J. Comput. Oretical Nanosci.* **2015**, *12*, 1292–1299. [CrossRef]
69. Chen, H.; Ionescu, M.; Ishdorj, T.-O.; Păun, A.; Păun, G.; Pérez-Jiménez, M.J. Spiking neural P systems with extended rules: Universality and languages. *Nat. Comput.* **2008**, *7*, 147–166. [CrossRef]
70. Peng, H.; Wang, J.; Ming, J.; Shi, P.; Pérez-Jiménez, M.J.; Yu, W.; Tao, C. Fault diagnosis of power systems using intuitionistic fuzzy spiking neural P systems. *IEEE Trans. Smart Grid* **2018**, *9*, 4777–4784. [CrossRef]
71. Wang, J.; Peng, H.; Yu, W.; Ming, J.; Pérez-Jiménez, M.J.; Tao, C.; Huang, X. Interval-valued fuzzy spiking neural P systems for fault diagnosis of power transmission networks. *Eng. Appl. Artif. Intell.* **2019**, *82*, 102–109. [CrossRef]
72. Yu, W.; Wang, J.; Peng, H. Fault diagnosis of power systems using fuzzy reasoning spiking neural P systems with interval-valued fuzzy numbers. *Rom. J. Inf. Sci. Technol.* **2017**, *1*, 5–17.
73. Wang, T.; Zhang, G.; Rong, H.; Pérez-Jiménez, M.J. Application of Fuzzy Reasoning Spiking Neural P Systems to Fault Diagnosis. *Int. J. Comput. Commun. Control.* **2014**, *9*, 786–799. [CrossRef]
74. Huang, K.; Wang, T.; He, Y.; Zhang, G.; Peŕez-Jiménez, M.J. Temporal fuzzy reasoning spiking neural P systems with real numbers for power system fault diagnosis. *J. Comput. Theor. Nanosci.* **2016**, *13*, 3804–3814. [CrossRef]
75. Wang, T.; Zhang, G.; Pérez-Jiménez, M.J. Fault Diagnosis Models for Electric Locomotive Systems Based on Fuzzy Reasoning Spiking Neural P Systems. In *Membrane Computing*; CMC 2014. LNCS 8961; Gheorghe, M., Rozenberg, G., Salomaa, A., Sosík, P., Zandron, C., Eds.; Springer: New York, NY, USA, 2014.
76. Rong, H.; Zhu, M.; Feng, Z.; Zhang, G.; Huang, K. A Novel Approach to Fault Classification of Power Transmission Lines Using Singular Value Decomposition and Fuzzy Reasoning Spiking Neural P Systems. *Rom. J. Inf. Sci. Technol.* **2017**, *20*, 18–31.
77. Rong, H.; Ge, M.; Zhang, G.; Zhu, M. An Approach for Detecting Fault Lines in a Small Current Grounding System using Fuzzy Reasoning Spiking Neural P Systems, *Int. J. Comput. Commun. Control.* **2018**, *13*, 521–536. [CrossRef]
78. Zhu, M.; Yang, Q.; Dong, J.; Zhang, G.; Gou, X.; Rong, H.; Paul, P.; Neri, F. An adaptive optimization spiking neural P system for binary problems. *Int. J. Neural Syst.* **2020**. [CrossRef]
79. Păun, G.; Rozenberg, G.; Salomaa, A. *The Oxford Handbook of Membrane Computing*; Oxford University Press: London, UK, 2009.
80. Zhang, X.; Zeng, X.; Pan, L.; Luo, B. A spiking neural P system for performing multiplication of two arbitrary natural numbers. *Chin. J. Comput.* **2009**, *32*, 2362–2372.
81. Hamabe, R.; Fujiwara, A. Asynchronous SN P systems for logical and arithmetic operations. In Proceedings of the International Conference on Foundations of Computer Science, Las Vegas, NV, USA, 21–23 October 2012; pp. 58–64.
82. Zhang, G.; Rong, H.; Paul, P.; He, Y.; Neri, F.; Pérez-Jiménez, M.J. A Complete Arithmetic Calculator Constructed from Spiking Neural P Systems and its Application to Information Fusion. *Int. J. Neural Syst.* **2020**. [CrossRef]
83. Díaz-Pernil, D.; Peña-Cantillana, F.; Gutiérrez-Naranjo, M.A. A parallel algorithm for skeletonizing images by using spiking neural P systems. *Neurocomputing* **2013**, *115*, 81–91. [CrossRef]

84. Idowu, R.K.; Chandren, R.; Othman, Z.A. Advocating the use of fuzzy reasoning spiking neural P system in intrusion detection. In Proceedings of the Asian Conference on Membrane Computing ACMC 2014, Coimbatore, India, 18–19 September 2014.
85. Pang, S.; Chen, H.; Liu, H.; Yao, J.; Wang, M. A deadlock resolution strategy based on spiking neural P systems. *J. Ambient. Intell. Humaniz. Comput.* **2019**. [CrossRef]