An Adaptive Optimization Spiking Neural P System for Binary Problems

MING ZHU, QIANG YANG
School of Control Engineering,
Chengdu University of information Technology, Chengdu, 610225, China
E-mail: zhuming@cuit.edu.cn, yuxia2008@126.com

JIANPING DONG, GEXIANG ZHANG
College of Information Science and Technology,
Chengdu University of Technology, Chengdu 610059, China
E-mail: master_djp@163.com, zhgxdylan@126.com

XIANTAI GOU*, HAINA RONG, PRITHWINEEL PAUL
School of Electrical Engineering,
Southwest Jiaotong University, Chengdu, 610031, China
E-mail: 491098063@qq.com, ronghaina@126.com, prithwineelpaul@gmail.com

FERRANTE NERI*
COL Laboratory, School of Computer Science,
University of Nottingham, Nottingham, UK
E-mail: Ferrante.Neri@nottingham.ac.uk

Optimization Spiking Neural P System (OSNPS) is the first membrane computing model to directly derive an approximate solution of combinatorial problems with a specific reference to the 0/1 knapsack problem. OSNPS is composed of a family of parallel Spiking Neural P Systems (SNPS) that generate candidate solutions of the binary combinatorial problem and a Guider algorithm that adjusts the spiking probabilities of the neurons of the P systems. Although OSNPS is a pioneering structure in membrane computing optimization, its performance is competitive with that of modern and sophisticated metaheuristics for the knapsack problem only in low dimensional cases.
In order to overcome the limitations of OSNPS, the present paper proposes a novel Dynamic Guider algorithm which employs an adaptive learning and a diversity based adaptation to control its moving operators. The resulting novel membrane computing model for optimization is here named Adaptive Optimization Spiking Neural P System (AOSNPS). Numerical result shows that the proposed approach is effective to solve the 0/1 knapsack problems and outperforms multiple various algorithms proposed in the literature to solve the same class of problems even for a large number of items (high dimensionality). Furthermore, case studies show that a AOSNPS is effective in fault sections estimation of power systems in different types of fault cases: including a single fault, multiple faults and multiple faults with incomplete and uncertain information in the IEEE 39 bus system and IEEE 118 bus system.

Keywords: Spiking neural system; adaptive optimization spiking neural P system; adaptive learning rate; adaptive mutation; power system fault diagnosis; combinatorial optimization; membrane computing.

1. Introduction

Artificial neural networks (ANNs), also referred to as neural networks (NNs) or connection models, are computational models consisting of interconnected neurons[10, 17] that mimic the behavioral characteristics of biological neural networks and

perform distributed parallel information processing. ANNs have been extensively investigated and widely used in various fields, such as signal and image processing[13; 15; 82], classification[4], pattern recognition[7; 37], earthquake prediction[1; 51; 52], epilepsy and seizure detection[27; 28], medical diagnostics[47], and optimization[2; 3; 5; 12; 11].

In the last decades, ANNs have passed three developmental generations with notable characteristics.

- The fundamental feature of the **first generation** is McCulloch–Pitts neurons[55] (perceptrons or threshold gates), where only Boolean functions[41] can be processed and output digital results.
- The salient feature of the **second generation** is the presence of an activation function with weighted learning ability and the fact that analog input and output can be processed[41].
- The introduction of time[18] (single action spiking of the neuron) concept when neurons encode information is a prominent feature of the **third generation** of ANNs.

Spiking Neural Networks (SNNs) present a successful ANN structure belonging to the second[40] and third generations[23; 56; 60; 26; 24]. The success of SNNs is due to the fact that they are both computationally powerful and biologically plausible models of neuronal processing[53; 16; 81; 45; 79]. Some successful implementations of SNN are reported in Ref.[42; 57; 20; 22; 80] while a successful implementation in the medical domain is presented in Ref.[25].

Over the past two decades, in parallel with the development of ANNs, other computational devices attracted attention. Specifically, Membrane Computing[54] increased significantly its popularity, gaining relevance as a computational paradigm. Membrane computing is a branch of natural computing which designs computing models, called P systems, inspired from the structure and the functioning of the biological cells. Most of P systems can be viewed as a general purpose automaton equivalent to the Turing Machine[74]. Due to their general purpose nature, P Systems can be used in a broad range of real-world applications, such as computer graphics[21], cryptography[74] and robot control[8; 64].

Membrane computing models can be divided into three categories on the basis of their structure:

- **cell-like P systems**, inspired by the intracellular communication[54];
- **tissue-like P systems**, inspired by intercellular communication[44];
- **neural-like P systems**, inspired by the communication among neurons[44], just like an ANN.

Hence, at the conceptual level, neural-like P systems can be viewed as an intersection between ANN and Membrane Computing. Although the conceptual definitions of neural P systems have been given in Ref[44], one of their greatest success stories leading to practical use is due to the Spiking Neural P Systems (SNPS)[35], which integrate the SNN logic within Membrane Computing. Unlike the other P systems[14; 21; 74] which use multiple symbols to encode the pieces of information required, SNPS[35; 50; 71] usually deal with a unique symbol, namely spike, that is repeated multiple times, see also Ref. [49; 34]. The name "spike" is due to the analogy with electrical impulses sent from a neuron to another one along the synapses. Among the various membrane computing models, SNPS is one of the most promising and studied[9; 48; 59; 58]. The SNPS consists of a set of neurons placed in the nodes of a graph and sending signals (spikes) along synapses (edges of the graph), under the control of firing and forgetting rules[68; 69; 70; 73]. One neuron is designated as the output neuron of the system and its spikes can be sent out into the environment, thus producing a different interpretations of the result, including: 1) the number of spikes sent out into the environment in total, along the computation; 2) the number of steps between the first two spikes sent to the environment; 3) a so-called spike-train, as the sequence of spikes sent out all along the computation (in every step, 0 or more spikes are sent out into the environment.).

This paper focuses on SNPS and in particular on its use for addressing optimization problems. The first study of solving combinatorial optimization problems by means of SNPS was proposed in Ref.[78] where the 0/1 Kanpsack Problem was ad-

dressed. In Ref.[78] an Extended SNPS (ESNPS) was obtained by introducing two additional neurons, the probabilistic selection of evolution rules and the output (spike train) collection from multiple neurons. In such work, multiple ESNPS are arranged in parallel, each of them generating a candidate solution. Furthermore, in Ref.[78] a Guider algorithm has been introduced to adjust probabilities of the spiking rules (by continuous learning generation by generation). The resulting computational model, namely Optimization Spiking Neural P System (OSNPS), appeared to be promising and displayed a performance competitive in comparison with that of six optimization algorithms taken as a reference.

As indicated in Ref.[78], one of the main limitations of OSNPS lies in the fact that the learning rate is constant, while the search may require a different learning rate in different moments of the search. Due to this limitation, it was observed in Ref.[78] that the probability variation operated by the Guider algorithm does not always allow an effective learning of the family of SNPS. Thus the performance of the OSNPS is not as good as that of some modern metaheuristics, especially when a large number of items is considered in the knapsack problem. For the sake of clarity, it must be remarked that unlike membrane-inspired evolutionary algorithms (MIEAs)[32; 38; 76; 77] that combine a P system framework with metaheuristic algorithms to solve optimization problems, OSNPS uses only SNPS to generate the candidate solutions of the combinatorial optimization problems.

The present paper extends the work of Ref.[78] and proposes a novel algorithm, namely Adaptive Optimization Spiking Neural P System (AOSNPS) to solve combinatorial problems with binary encoding. More specifically, a novel adaptive learning rate and adaptive mutation are proposed and integrated within OSNPS. In addition, the proposed AOSNPS includes a mutation operator governed by two dynamic parameters that estimate the accuracy of the candidate solution and diversity of probability matrices. The proposed mutation enhances upon the balance between exploration and exploitation features of OSNPS. Finally, the real application of AOSNPS to fault section estimation of power systems. This is the first time to use AOSNPS to solve real application problems. When the status information of protective relays and circuit breakers read from a supervisory control and data acquisition system is input, AOSNPS can automatically search and output fault sections. Case studies show that AOSNPS is effective in fault sections estimation of power systems in different types of fault cases including single fault, multiple faults and multiple faults with incomplete and uncertain information.

The remainder of this study is organized as follows: Section 2 briefly introduces the used notation, the basics of SNPS theory used in this article as well as the OSNPS as in its original implementation[78]. Section 3 presents the proposed AOSNPS consisting adaptive learning rate, adaptive mutation and the novel guider. The numerical results obtained by the algorithm proposed when applied to the $0/1$ knapsack problem are shown in Section 4, while an application to power systems, is given in Section 5. Finally, conclusion and future work are given in Section 6.

2. Background: Problem and Optimization by Spiking Neural P Systems

In this section, we briefly introduce the notation and analyse the basic concepts of optimization by SNPS. We address the minimisation/maximisation of a function $f(\mathbf{x})$ where $\mathbf{x} = (x_1, x_2, \dots x_m)$ is a vector of binary numbers.

The SNPS [35] is an automaton. The distinguishing feature of SNPS is that the sequence of configurations can associate a spike train[36; 43; 78]. If the output neuron spikes, then we have 1 and otherwise we have 0. Hence, the spike train can be represented by the sequence of ones and zeros. Here, we give an extended definition of SNPS.

2.1. *Extended Spiking Neural P System*

Formally, an *extended spiking neural P system* (abbreviated as ESNPS), of degree (m, q) with $m, q \geq 1$, is a tuple of the form

$$\Pi = (O, \sigma_1, \dots, \sigma_m, \sigma_{m+1}, \dots, \sigma_{m+q}, syn, I_{out})$$

where:

(1) $O = \{a\}$ is the singleton alphabet (a is called *spike*);
(2) σ_i, $1 \leq i \leq m + q$, is an ordered tern (n_i, R_i, P_i) such that:

- ⋆ n_i is a natural number.
- ⋆ R_i, for $1 \leq i \leq m$, is a finite set of rules of the types:

(i) $E/a^c \rightarrow a; d$, where E is a regular expression over O, $c \geq 1$, and $d \geq 0$, d is the *delay* associated with the rule (`spiking rule`);

(ii) $a^s \rightarrow \lambda$, for some $s \geq 1$, with the restriction that for each rule $E/a^c \rightarrow a; d$ from the previous type we have $a^s \notin L(E)$ (`forgetting rule`).

- ⋆ $R_{m+1}, \ldots, R_{m+q}$ is a finite set of spiking rules.
- ⋆ P_i, for $1 \leq i \leq m$, is a map from the set R_i into $[0, 1]$ such that $\sum_{r \in R_i} P_i(r) = 1$.
- ⋆ $P_{m+1}, \ldots, P_q$ are the identically one functions.

(3) $syn \subseteq \{(\sigma_i, \sigma_j) \mid 1 \leq i, j \leq m + q \wedge i \neq j\}$.

(4) $I_{out} \subseteq \{\sigma_1, \ldots, , \sigma_m\}$.

An ESNPS, Π, of degree (m, q), with $m, q \geq 1$ can be viewed as a set of neurons $\{\sigma_1, \ldots, \sigma_m, \sigma_{m+1}, \ldots, \sigma_{m+q}\}$ placed in the nodes of a directed graph G_Π, where the set syn specifies the arcs of the graph which are labeled by natural numbers representing the *delays* associated with synapses: $D_{syn}(\sigma_i, \sigma_j)$ will denote the label associated with the arc (σ_i, σ_j). If $(\sigma_i, \sigma_j) \in syn$ then we say that σ_i is a *pre-synaptic neuron* of σ_j or σ_j is a *postsynaptic neuron* of σ_i. In these systems there are two kind of neurons: $\sigma_1, \ldots, \sigma_m$ are called *working neurons*, some of them can also act as *output neurons*, whereas $\sigma_{m+1}, \ldots, \sigma_{m+q}$ are *supplier neurons* that can send spikes to each working neuron. For each neuron σ_i, n_i is the number of spikes initially contained in that neuron, R_i is a finite set of (spiking and/or forgetting) rules with probabilities associated by means of functions P_i.

The semantics of such an ESNPS, Π, is defined as follows. A *configuration* C_t at a moment t (a natural number, $t \geq 0$) of an ESNPS of degree (m, q) is a tuple $(p_1, \ldots, p_{m+q})$ where $p_i, 1 \leq i \leq m + q$, describes the number of spikes present in the neuron σ_i at instant t. The *initial configuration* of Π, that is, the configuration at moment $t = 0$, is $(n_1, \ldots, n_{m+q})$, being $\sigma_i = (n_i, R_i, P_i)$, for $1 \leq i \leq m + q$.

A spiking rule $E/a^c \rightarrow a; d \in R_i$ is applicable to a configuration C_t, at the moment t, if the neuron σ_i contains exactly $k \geq c$ spikes at instant t and $a^k \in L(E)$. By applying such a rule, c spikes are consumed from the neuron σ_i, it is fired and all postsynaptic neurons σ_j from the neuron σ_i would receive p spikes from neuron σ_i, at step $t + D_{syn}(i, j) + 1$, that is, if $D_{syn}(i, j) = 0$, then the spikes are received by such postsynaptic neuron σ_j immediately, otherwise the spikes are received after $D_{syn}(i, j)$ steps. A forgetting rule $a^s \rightarrow \lambda \in R_i$ is applicable to a configuration C_t at the moment t if the neuron σ_i contains exactly s spikes at instant t. By applying such a rule, s spikes are removed from the neuron σ_i.

In each time unit, if neuron σ_i is *enabled* (i.e., one of its rules can be used), then one rule from R_i (and only one) must be applied, chosen non–deterministically among all possible rules applicable to σ_i. That is, each neuron processes sequentially its spikes by using, at most, one rule in each time unit. Let us recall that a spiking rule and a forgetting rule cannot be simultaneously applied to a neuron. The rules are used in a maximally parallel way at the level of the system: at each step, all neurons which can use some rule, must evolve using one rule.

Using the rules of the system in the way described before, a configuration C_{t+1} can be reached from another configuration C_t; such a step is called a *transition step*, and we denote it by $C_t \Longrightarrow C_{t+1}$. Any (finite or infinite) sequence of transitions $C_0 \Longrightarrow C_1 \Longrightarrow C_2 \Longrightarrow \ldots \Longrightarrow C_r$, $r \in \mathbb{N} \cup \{\infty\}$, is a *computation* if the following holds: (a) the first term of the sequence, C_0, is the initial configuration of the system; (b) for each n, the $(n+1)$-th term of the sequence is obtained from the n-th term in one transition step; and (c) if the sequence is finite (called *halting computation*) then the last term C_r, $r \in \mathbb{N}$, of the sequence is called a *halting configuration*, that is, a configuration where no rule of the system is applicable to it.

In order to propose a novel algorithm, AOSNPS, to solve combinatorial optimimization problems, a specific ESNPS (where $q = 2$ and $n_i = 1$), is considered with the following syntactical restrictions:

$$\Pi = (O, \sigma_1, \ldots, \sigma_m, \sigma_{m+1}, \sigma_{m+2}, syn, I_0),$$

where

(1) $O = \{a\}$ is the singleton alphabet, a is called *spike*

(2) $\sigma_1, \ldots, \sigma_m$ are neurons of the form

$$\sigma_i = (1, R_i, P_i).$$

Further comments on the neurons are

(a) in each neuron σ_i, there is only 1 initial spike, i.e. $n_i = 1$, for $i = 1, 2, \ldots, m+2$

(b) $R_i = \{r_i^1, r_i^2\}$ is a set of rules, **spiking** $r_i^1 : a \to a$ and **forgetting** $r_i^2 : a \to \lambda$, respectively

(c) $P_i = \{p_i^1, p_i^2\}$ is a finite set of **probabilities**, where p_i^1 and p_i^2 are the selection probabilities of rules r_i^1 and r_i^2, respectively, and satisfy p_i^1+p_i^2=1.

(3) The two additional neurons, $\sigma_{m+1} = \sigma_{m+2} = (1, a \to a)$, work as a step by step supplier of spikes to neurons $\sigma_1, \sigma_2, \ldots, \sigma_m$.

(4) $syn = \{(\sigma_i, \sigma_j) | (i = m+2$ **AND** $1 \leq j \leq m+1$ **OR** $(i = m+1$ **AND** $j = m+2)\}$.

(5) $I_0 = \{1, 2, \ldots, m\}$ is a finite set of *output* neurons, i.e., the output is a spike train formed by concatenating the outputs of σ_1, σ_2, ..., σ_m.

The structure of the specific ESNPS is shown in Fig. 1.

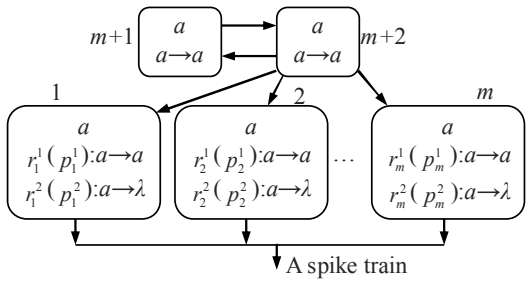

Figure 1. Logical and functioning scheme of the specific ESNPS

2.2. *Optimization Spiking Neural P System*

The OSNPS is composed of multiple parallel ESNPSs and a Guider algorithm, that supervises the family of ESNPSs, as shown in Fig. 2.

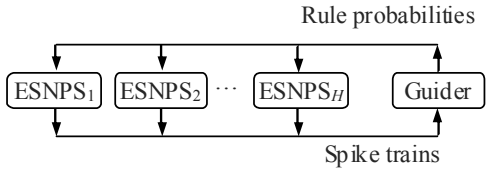

Figure 2. Structure of OSNPS (and AOSNPS)

Each ESNPS$_i$, with $1 \leq i \leq H$, has the structure shown in Fig. 1. The Guider algorithm in Fig. 2 is used to adjust the probabilities P_i of each ESNPS$_i$. The input of the Guider algorithm is a spike train T_s, that is a set of H binary strings/candidate solutions of length m (a binary matrix of $H \times m$ elements) produced by each ESNPS$_i$. The output of the Guider algorithm is the rule probability matrix $\mathbf{P_R} = [p_{ij}^1]_{H \times m}$ (the apex 1 indicates the probability of the bit to be 1), which is composed of the spiking rule probabilities of H ESNPSs, i.e.,

$$\mathbf{P_R} = \begin{pmatrix} p_{11}^1 & p_{12}^1 & \cdots & p_{1m}^1 \\ p_{21}^1 & p_{22}^1 & \cdots & p_{2m}^1 \\ \vdots & \vdots & \ddots & \vdots \\ p_{H1}^1 & p_{H2}^1 & \cdots & p_{Hm}^1 \end{pmatrix}.$$

The probabilities are fed back to the ESNPSs which are updated accordingly.

Input: Spike train T_s, probabilities p_j^a, learning rate Δ, number of ESNPS H and the current best solution $\mathbf{x} = (x_1, x_2, \ldots x_m)$ of length m
Rearrange T_s as matrix $\mathbf{P_R}$
$i = 1$
while $(i \leq H)$ **do**
  j=1
  while $(j \leq m)$ **do**
    if $\left(rand() < p_j^a\right)$ **then**
      $k_1, k_2 = ceil\,(rand() * H), k_1 \neq k_2 \neq i$ and correct $\mathbf{x_{k_1}}$ and $\mathbf{x_{k_2}}$ if violate the constraint
      if $\left(f\left(x^{k_1}\right) > f\left(x^{k_2}\right)\right)$ **then**
        $x_j = x_j^{k_1}$
      else
        $x_j = x_j^{k_2}$
      end if
      if $(x_j == 1)$ **then**
        $p_{ij}^1 = p_{ij}^1 + \Delta$
      else
        $p_{ij}^1 = p_{ij}^1 - \Delta$
      end if
    else
      if $(x_j == 1)$ **then**
        $p_{ij}^1 = p_{ij}^1 + \Delta$
      else
        $p_{ij}^1 = p_{ij}^1 - \Delta$
      end if
    end if
    if $(p_{ij}^1 > 1)$ **then**
      $p_{ij}^1 = p_{ij}^1 - \Delta$
    else
      if $(p_{ij}^1 < 0)$ **then**
        $p_{ij}^1 = p_{ij}^1 + \Delta$
      end if
    end if
    $j = j + 1$
  end while
  $i = i + 1$
end while
Output: Rule probability matrix $\mathbf{P_R}$

Figure 3. The Guider algorithm of OSNPS

Fig. 3 provides the details of the Guider algorithm used in OSNPS [78]. The basic idea is that one current best solution (binary vector)

$$\mathbf{x} = (x_1, x_2, \dots x_m)$$

is stored in memory and two candidate solutions $\mathbf{x_{k1}}$ and $\mathbf{x_{k2}}$ are randomly selected, with a certain probability p_j^a, and compared on the basis of a fitness (objective function) f. If appropriate, the constraints are handled.

If $\mathbf{x_{k1}}$ and $\mathbf{x_{k2}}$ are generated, for each design variable x_j, the current solution is perturbed ($\mathbf{x}$ inherits the design variable of the winning candidate solution). Then, the probability associated with the most successful solution is increased by a constant learning rate Δ. Conversely, the probability associated with the candidate solution displaying a lower performance is reduced. If $\mathbf{x_{k1}}$ and $\mathbf{x_{k2}}$ are not generated, the binary variable of x_j is directly used to affect the probability, see lines 19-24 of the pseudocode in Fig. 3. More details on OSNPS can be found in Ref. [78].

3. Adaptive Optimization Spiking Neural P System

The proposed AOSNPS employs the same structure of OSNPS depicted in Fig. 2 but makes use of a novel Dynamic Guider algorithm to manipulate the probability matrix $\mathbf{P_R}$. More specifically, the Guider algorithm contains two novel adaptive mechanisms: 1) an adaptive learning rate Δ; 2) an adaptive mutation. The novel mechanisms are described in Subsections 3.1 and 3.2, respectively. The new Dynamic Guider algorithm embedding these two elements is outlined in Subsection 3.3.

3.1. *Adaptive Learning Rate*

The learning rate Δ is the step size of probability adjustment for the elements of the probability matrix $\mathbf{P_R}$:

$$p_{ij}^1 = p_{ij}^1 + \Delta$$
$$p_{ij}^1 = p_{ij}^1 - \Delta$$

with $1 \leq i \leq H$ and $1 \leq j \leq m$. In Ref. [78], Δ is a random number between 0.005 and 0.02 set at the beginning of the optimization and kept constant throughout the OSNPS execution.

We propose here a variable and adaptive learning rate Δ. In order to better explain the rationale of the proposed adaptation, let us revisit the role of p_i^1 of ESNPS presented in Section **??**.

The term p_i^1 is the selection probability of rule r_i^1 while r_i^1 is the spiking rule in each neuron. If the value of p_i^1 is large, the rule r_i^1 has a high probability of execution and the rule r_i^2 has a low probability of execution since $p_i^1 + p_i^2 = 1$. If the output neuron spikes, a 1 is written as a bit of the candidate solution, otherwise a 0 is written. Thus, if we want to get 1, p_i^1 should be large (ideally $p_i^1 = 1$) and if we want to get 0, p_i^1 should be small (ideally $p_i^1 = 0$).

This paper proposes an adaptive probability adjustment step size for each neuron. At each time unit, the adaptive updating rule of probability is

$$p_{ij}^1 = p_{ij}^1 + \Delta_{ij}^a$$

where Δ_{ij}^a (a stands for "adaptive") is the step size and is defined as

$$\Delta_{ij}^a = \frac{P_b - p_{ij}^1}{2}.$$

Δ_{ij}^a is designed to take the middle point of the distance between the current probability p_{ij}^1 and the ideal probability. The ideal probability $P_b = \{0, 1\}$ is the lower or upper bound of the probability of p_{ij}^1 (the pedex b stands for "bound"). For $P_b = 1$, the update rule is

$$p_{ij}^1 = p_{ij}^1 + \frac{1 - p_{ij}^1}{2} = 0.5 + 0.5p_{ij}^1$$

while for $P_b = 0$ the update rule is

$$p_{ij}^1 = p_{ij}^1 + \frac{0 - p_{ij}^1}{2} = 0.5p_{ij}^1.$$

Compared with the learning rate Δ defined in OSNPS, the adaptive learning rate Δ_{ij}^a proposed in this paper presents the following advantages.

- Δ_{ij}^a changes for each neuron at each time unit during the algorithm execution by following the learning needs. If the distance between the current probability p_{ij}^1 and the ideal probability P_b is big, the learning rate Δ_{ij}^a is big. On the other hand, if the distance between the current probability p_{ij}^1 to the ideal probability P_b is small, the learning rate is also small.

- The adaptive learning rate allows a quick achievement of the desired probability. For example, if we want to get 1 from a neuron, from an initial probability $p_{ij}^1 = 0.1$ the proposed adaptive learning rate Δ_{ij}^a enables to reach $0.9 < p_{ij}^1 < 1$ after four steps whereas with a constant Δ over 40 steps are needed.
- The probability of p_{ij}^1 does not overflow. In OSNPS, it may result $p_{ij}^1 > 0$ or $p_{ij}^1 < 0$. In this case, an extra mechanism is required. The use of the proposed adaptive learning rate ensures that the probability overflow never occurs.

3.2. *Adaptive Mutation*

The proposed AOSNPS makes also use of an adaptive mutation strategy. Two dynamic parameters P_{m1} and P_{m2} are defined to characterise and monitor the learning (evolutionary state) of AOSNPS. The parameter P_{m1} is the first mutation probability which varies between 0 and 1 and is defined as follows:

if $G_{bf}(gen) > G_{bf}(gen-1)$ **then**
  $P_{m1} = 0$
end if
if $G_{bf}(gen) = G_{bf}(gen-1)$ **then**
  $P_{m1} = P_{m1} + \frac{1}{N_{\max}}$
end if

where $G_{bf}(gen)$ is the best fitness value ever computed at the generation time of the execution and $G_{bf}(0)$ is the best fitness at the initialization. The parameter $N_{\max}$ determines a stopping criterion: if the global best fitness does not improve for consecutive $N_{\max}$ generations the algorithm stops.

The parameter P_{m2} is the second mutation probability and is defined as follows:

$$P_{m2} = \frac{DP_a(gen)}{DP_a(0)}$$

where $DP_a\,(gen)$ is an aggregated metric representing the diversity (of probabilities) at the current generation gen, see [46]. This value is calculated as:

$$DP_a\,(gen) = \frac{2}{(H-1)(H-2)} \sum_{i=1}^{H-1} \left(\sum_{j=i+1}^{H} \frac{1}{m} \sum_{k=1}^{m} \left| p_{ik}^{gen} - p_{jk}^{gen} \right| \right).$$

The value $DP_a(0)$ is the average probability at the initialization.

The triggering rule for adaptive mutation is defined as

$$rand_1() < P_{m1} \quad \textbf{AND} \quad rand_2() > P_{m2}$$

where $rand_1()$ and $rand_2()$ are two random numbers in the range $[0, 1]$.

P_{m1} and P_{m2} are two independent dynamic parameter. P_{m1} is used to describe the trend of improvement of the optimal solution. P_{m2} is used to describe the diversity of probability sample matrix between rows in the current generation. N_{max} is the parameter of when the algorithm should stop running. The value of N_{max} affect the standard deviation between the results of multiple runs of AOSNPS for the same issue. In order to make the results of multiple runs more concentrated, we should choose a larger N_{max}. The logic of the mutation probabilities P_{m1} and P_{m2} is summarised in the following points.

- If a new current best solution has been detected at the current generation ($G_{bf}(gen) > G_{bf}(gen-1)$), then the mutation is not triggered ($P_{m1} = 0$). Conversely, if the current best solution has not been updated ($G_{bf}\,(gen) = G_{bf}\,(gen-1)$), the probability of triggering the mutation is increased ($P_{m1} = P_{m1} + \frac{1}{N_{\max}}$).
- If the diversity $DP_a(gen)$ is larger than that at initialization $DP_a(0)$), the probability for the mutation to be triggered is low.
- If the current best solution is not updated for many generations ((G_{bf}) remains the same), and the diversity $DP_a(gen)$ of the probability matrix $\mathbf{P_R}$ is low, the mutation is performed with a high probability (P_{m1} is large, P_{m2} is small).

When triggered the mutation of the matrix $\mathbf{P_R}$ is performed in the following way:

for $i = 1, 2, \cdots, H$ with $i \neq R_{bestfit}$ **do**
  for $j = 1, 2, \cdots, m$ **do**
    if $rand_3() < P_j^m$ **then**
      $p_{ij}^1 = rand()$
    end if
  end for
end for

where $rand_3()$ and $rand()$ are two random numbers

in the range $[0, 1]$. The mutation probability P_j^m is a parameter sampled in the range $[0, 0.1]$ at the beginning of the Dynamic Guider algorithm's execution. $R_{bestfit} \in [1, H]$ is the row coordinate of the best candidate solution found at the current generation. This means that the best candidate solution, in an elitist fashion, does not participate in the mutation.

Based on the adaptive learning rate and the adaptive mutation rule, the Dynamic Guider algorithm is an algorithm operating on the learning probabilities to support the generation of successful solution for the binary combinatorial problem.

One current best solution $\mathbf{x}$ is stored in memory and modified by crossover with other candidate solutions. The candidate solutions are generated by the parallel ESNPSs (one solution by each ESNPS$_j$) and processed in pairs by the evolutionary Dynamic Guider algorithm that evaluates and compare their fitness values. The evaluation of these candidate solutions are used to modify the probability matrix $\mathbf{P_R}$ and hence generate solutions with a higher performance. Like for OSNPS, the constraint is handled by means of the random chromosome repair technique, see [29; 30; 75].

The evolution of the $\mathbf{P_R}$ matrix is handled by two mechanisms: the update of the adaptive learning rate which happens every time two candidate solutions are sampled as described in Subsection 3.1, the re-initialisation of the neuron probability which is triggered only by the satisfaction of the conditions described in Subsection 3.2.

3.3. *The Dynamic Guider Algorithm of AOSNPS*

Hence, we may consider AOSNPS as a co-evolution where the (binary) candidate solutions of the optimization problem evolve with the probabilities of the neurons in each of the P systems. The pseudocode of the proposed evolutionary Dynamic Guider is shown in Fig. 4.

4. Numerical Results: 0/1 knapsack problem

Given a group of items, each item with its own weight and price, and a knapsack of limited capacity, the problem consists of selecting the items to make the total price of the knapsack as high as possible without violating its maximum capacity. If we indicate with m the total number of items available and we label each item with a number $j = 1, 2, \ldots, m$, we can represent the selection of the items as a binary vector $\mathbf{x}$.

Input: Spike train T_s, probabilities p_j^a, mutation probability P_j^m, number of ESNPS H and the current best solution $\mathbf{x} = (x_1, x_2, \ldots x_m)$ of length m
Rearrange T_s as matrix $\mathbf{P_R}$, initialise $gen = 0$ and $P_{m1} = 0$ and then calculate $G_{bf}(0)$ and $DP_a(0)$
while $(P_{m1} \leq 1)$ **do**
  $gen = gen + 1$
  $i = 1$
  while $(i \leq H)$ **do**
    j=1
    while $(j \leq m)$ **do**
      if $(rand\,() < p_j^a)$ **then**
        $k_1, k_2 = ceil(rand * H)$, $k_1 \neq k_2 \neq i$ and correct $\mathbf{x_{k_1}}$ and $\mathbf{x_{k_2}}$ if violate the constraint
        if $(f\left(x^{k_1}\right) > f\left(x^{k_2}\right))$ **then**
          $x_j = x_j^{k_1}$
        else
          $x_j = x_j^{k_2}$
        end if
        {**Adaptive Learning Rate**}
        if $(x_j == 1)$ **then**
          $p_{ij}^1 = 0.5 + 0.5 p_{ij}^1$
        else
          $p_{ij}^1 = 0.5 p_{ij}^1$
        end if
      else
        if $(x_j^{max} == 1)$ **then**
          $p_{ij}^1 = 0.5 + 0.5 p_{ij}^1$
        else
          $p_{ij}^1 = 0.5 p_{ij}^1$
        end if
      end if
      $j = j + 1$
    end while
    $i = i + 1$
  end while
  {**Adaptive Mutation**}
  calculate $G_{bf}(gen)$, $DP_a(gen)$ and $R_{bestfit}$
  if $(G_{bf}(gen) > G_{bf}(gen - 1))$ **then**
    $P_{m1} = 0$
  else
    $P_{m1} = P_{m1} + \frac{1}{N_{\max}}$
  end if
  $P_{m2} = \frac{DP_a(gen)}{DP_a(0)}$
  if $(rand_1() < P_{m1}$ **AND** $rand_2() > P_{m2})$ **then**
    $i = 1$
    while $(i \leq H)$ **do**
      if $i \neq R_{bestfit}$ **then**
        $j = 1$
        while $(j \leq m)$ **do**
          if $rand_3() < P_j^m$ **then**
            $p_{ij}^1 = rand()$
          end if
          $j = j + 1$
        end while
      end if
      $i = i + 1$
    end while
  end if
end while
Output: Rule probability matrix $\mathbf{P_R}$

Figure 4. The Dynamic Guider algorithm of AOSNPS

The 0/1 knapsack problem consists of the max-

imisation of the function

$$f(\mathbf{x}) = \sum_{j=1}^{m} p_j x_j$$

subject to

$$\sum_{j=1}^{m} \omega_j x_j \leq C$$

where p_j and ω_j are price and weight of the j^{th} item, respectively and C is the capacity of the knapsack. The parameters of the problem have been set in following way. The weights ω_j have been sampled from the interval $[1, \Omega]$, with $\Omega = 50$, $p_j = \omega_j + \frac{1}{2}\Omega$ and the average knapsack capacity C is applied:

$$C = \frac{1}{2} \sum_{j=1}^{m} \omega_j.$$

In order to validate the proposed AOSNPS, we tested it against the following algorithms designed/used in the literature

- Genetic Quantum algorithm (GQA) [39]
- Novel Quantum Evolutionary algorithm (NQEA) [33]
- Optimization Spiking Neural P System (OSNPS) [78]

For each algorithm we used the recommended parameter setting used in the original paper. The proposed AOSNPS has been run with $H = 50$ ESNPS. Regarding the evolutionary Guider algorithm, the learning probability p_j^a ($j = 1, \ldots, m$) uses the same value used in OSNPS (a random number in the range [0.05, 0.2]), the mutation probabilities $p_j^m = 0.01$ ($j = 1, \ldots, m$) and $N_{\max} = 500$.

The 0/1 knapsack problem has been run in different scenarios (problem instances), i.e. $m = 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000$. Each algorithm in this paper has been run for 30 independent runs. When the total weight of all selected items exceeds the capacity C (constraints violation), we implemented the random chromosome repair technique suggested in Ref. [29; 30; 75].

Table 1 displays the numerical results in terms of mean value $\mu \pm$ standard deviation σ of the knapsack cost $f(\mathbf{x})$. The best results for each problem instance are highlighted in bold.

Furthermore, the statistical significance of the results has been enhanced by the application of the Wilcoxon rank sum test, see [66]. A "+" indicates that AOSNPS significantly outperforms its competitor, a "-" indicates that the competitor significantly outperforms AOSNPS, and a "=" indicates that there is no significant difference in performance.

Numerical results in Table 1 show that the proposed AOSNPS is very efficient to address the 0/1 knapsack problem since it achieves the best results in nine cases out of the ten considered. In only one scenario (with the smallest number of items $m = 1000$) NQEA achieves slightly better results than AOSNSP. The effectiveness of the novel Dynamic Guider algorithm appears from the direct comparison against OSNPS: AOSNPS systematically outperforms OSNPS.

In order to further strengthen the statistical analysis of the presented results, we performed the Holm-Bonferroni [31] procedure for the eight algorithms ($N_A = 4$) and ten problem instances ($N_p = 10$). The rank R_k for $k = 1, \ldots, N_A$ by assigning, for each problem instance has been calculated. For each problem instance, a score N_A is assigned to the best algorithm, a score $N_A - 1$ to the second best, ..., 1 to the worst algorithm. The ranks R_k are the scores averaged over all the problem instances. Let us indicate with R_0 the ranking of AOSNPS. For the remaining $N_A - 1$ algorithm the score z_k is calculated as the values z_k have been calculated as:

$$z_k = \frac{R_k - R_0}{\sqrt{\frac{N_A(N_A+1)}{6N_p}}}.$$

By means of the z_k values, the corresponding cumulative normal distribution values p_k have been calculated, see [19]:

$$p_k = \frac{2}{\sqrt{\pi}} \int_{\frac{-z_k}{\sqrt{2}}}^{\infty} e^{-t^2} dt.$$

These p_k values have then been compared with the corresponding δ/k where δ is the level of confidence, set to 0.05 in this case.

These p_k values have then been compared with the corresponding δ/k where δ is the level of confidence, set to $\delta = 0.05$ in this case. Table 2 displays the ranks, z_k values, p_k values, and corresponding δ/k obtained. Moreover, it is indicated whether the null-hypothesis (that the two algorithms have indistinguishable performances) is "Rejected", i.e. the algorithms have statistically different performance, or

Table 1. Mean value ± standard deviation of the knapsack cost for the algorithms and problem instances under consideration

m	GQA			NQEA			OSNPS			AOSNPS	
	μ	σ	W	μ	σ	W	μ	σ	W	μ	σ
1000	26340.617	162.941	+	**29273.757**	131.036	=	28089.664	311.094	+	29225.319	186.584
2000	52908.434	208.761	+	58515.548	293.090	+	56150.321	535.740	+	**58561.778**	385.807
3000	78059.658	276.459	+	85886.637	502.8594	+	82745.029	692.8688	+	**86738.048**	586.712
4000	103801.413	422.955	+	113690.579	666.608	+	109458.886	724.4787	+	**114706.678**	831.661
5000	131224.181	342.227	+	142077.755	713.820	+	137927.802	835.4819	+	**143601.783**	1328.339
6000	157119.187	415.339	+	169516.775	577.702	+	164674.207	730.2888	+	**171543.765**	1515.563
7000	182669.798	440.894	+	196377.110	873.184	+	191659.447	849.2236	+	**200107.477**	1218.512
8000	208561.466	322.213	+	223674.462	953.050	+	217577.959	1128.401	+	**227193.619**	1531.739
9000	233945.639	570.136	+	249931.187	916.934	+	244397.767	1129.060	+	**254551.819**	1162.772
10000	259881.277	541.028	+	276903.178	1159.924	+	270663.831	769.233	+	**282101.492**	1774.430

"Accepted" if the distribution of values can be considered the same (there is no outperformance).

The Holm-Bonferroni procedure in Table 2 shows that AOSNPS achieves the best performance over all the algorithms taken into account. It can be observed that AOSNPS achieves the best ranking and significantly outperforms GQA and OSNPS.

Table 2. Holm-Bonferroni Procedure with AOSNPS as reference (Rank 3.9000e+00)

	R_k	z_k	p_k	$\frac{\delta}{k}$	Test
NQEA	3.1e+00	-1.3856e+00	1.6586e-01	5.00e-02	Accepted
OSNPS	2.0e+00	-3.2909e+00	9.9869e-04	2.50e-02	Rejected
GQA	1.0e+00	-5.0229e+00	5.0884e-07	1.67e-02	Rejected

4.1. *Comparative Analysis of AOSNPS and OSNPS*

In this subsection, we highlight the effectiveness of the new Dynamic Guider algorithm by comparing the behaviour of OSNPS and AOSNPS. The analysis involves the following five indicators[75].

(1) G_{bf}: **performance trend**. This is the objective function value of the current best solution at each generation gen.

(2) D_{hbw}: **Hamming distance between the best and worst binary individuals in a population**.

$$D_{hbw} = \frac{1}{m}\sum_{i=1}^{m}(x_{bi} \oplus x_{wi})$$

where, x_{bi} and x_{wi} are the ith bits in the best and worst binary solutions, respectively; m is the number of bits in a binary solution; the symbol $\oplus$ represents the OR operator.

(3) D_{hm}: **average Hamming distance** of all binary individuals in a population.

$$D_{hm} = \frac{2}{n(n-1)}\sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{1}{m}\sum_{k=1}^{m}(x_{ik} \oplus x_{jk})$$

where, x_{ik} and x_{jk} are the kth bits in the ith and jth binary solutions, respectively; m is the number of bits in a binary solution; n is the number of individuals in a population; the symbol $\oplus$ represents OR operator.

(4) DP_{bw}: **distance between the best and worst probability column vectors in** P_R corresponding to the best and worst fitness values in a population, respectively.

$$DP_{bw} = \frac{1}{m}\sum_{k=1}^{m}\left|p_{bk}^{1} - p_{wk}^{1}\right|$$

where p_{bk}^{1} and p_{wk}^{1} are the probabilities associated with the best and worst.

(5) DP_a: **average probability distance** of all probability individuals as shown in Subsection 3.2.

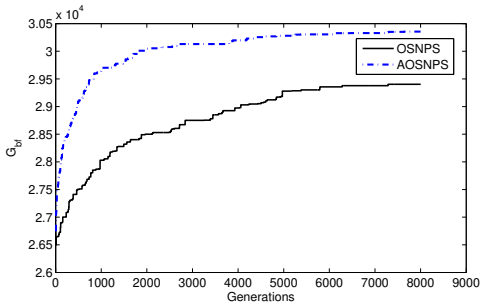

Figure 5. Performance trend G_{bf} of OSNPS and AOSNPS

Figures 5 - 9 depicts the trends of these five metrics for $m = 1000$ items for OSNPS and AOSNPS, respectively. The algorithms have been run with the same starting random initial probability

matrices $P_{R50\times1000}$. Both OSNPS and AOSNPS have been run 8000 generations respectively. Figure 5 shows that AOSNPS outperforms OSNPS throughout the entire run. Figures 6 and 7 show that AOSNPS exhibits better transient convergence:the diversity metrics D_{hbw} and D_{hm} of AOSNPS drop more rapidly than that of OSNPS in the early stages of the run. However, subsequently AOSNPS achieves and maintains a better diversity than OSNPS. This is the result of the joint contributions of the adaptive learning rate (the rapid decline of diversity in early stage is mainly due to the improved learning efficiency using $\Delta_{ij}^{adaptive}$) and the adaptive mutation (better diversity obtained in the middle and late stages is due to the frequent triggering of mutation rule).

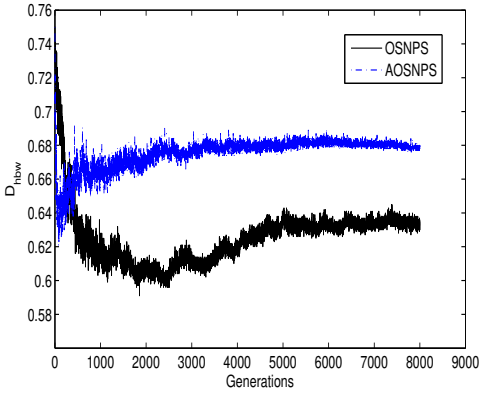

Figure 6. Plot of Hamming distance between the best and worst binary individuals D_{hbw}: OSNPS vs AOSNPS

Numerical results of DP_{bw} and DP_a in Fig.8 - 9 highlight the better ability of AOSNPS to maintain the diversity of probability individuals comparing with OSNPS. This is due to the contribution of the adaptive mutation. During the entire run of AOSNPS (8000 generations), the adaptive mutation was triggered 1971 times.

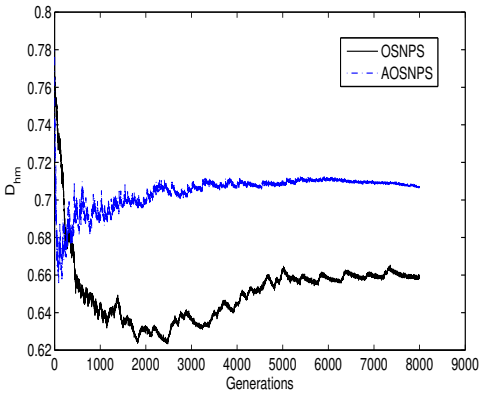

Figure 7. Plot of average Hamming distance D_{hm}: OSNPS vs AOSNPS

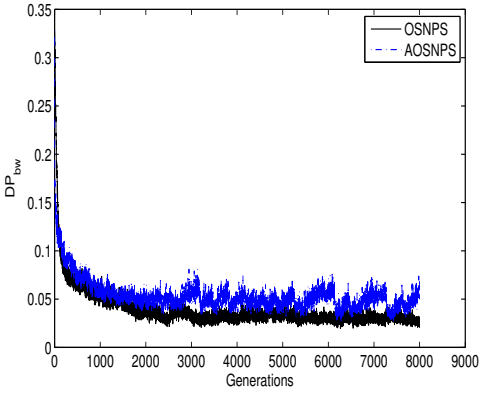

Figure 8. Plot of the distance between the best and worst probability DP_{bw}: OSNPS vs AOSNPS

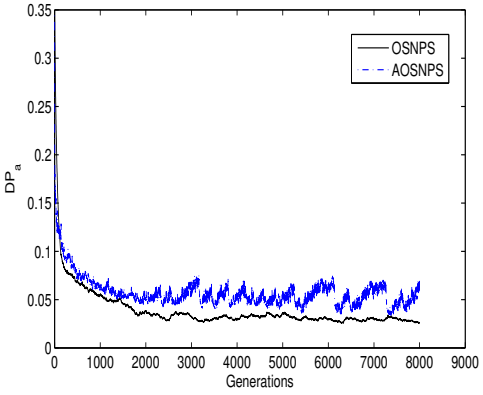

Figure 9. Plot of the average probability distance DP_a: OSNPS vs AOSNPS

Fig.10 shows the histogram of the mutation

triggering during the run. At the beginning of the algorithm execution, the mutation was not enabled because the optimal solution improved quickly. In the middle and at the late stages of the run, the mutation is often enabled to enhance upon a current best solution with a high performance.

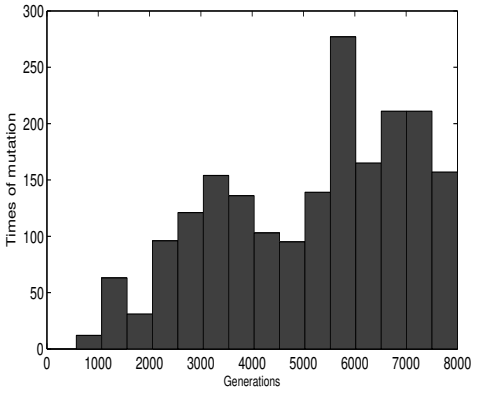

Figure 10. The histogram of the mutation triggering

In order to better display the behavior of adaptive mutation, Fig.11 displays the interval generations, i.e. the number of generations gen between two consecutive triggered mutations. When the interval generation is one, then the mutation has occurred for two consecutive generations. At this time, the global optimal solution may have not been improved for many generations (P_{m1} is large), and the difference between the populations is small (P_{m2} is small), which could correspond to a convergence to a suboptimal solution. In this situation, mutations are needed to explore possible new promising search directions. If the interval generation of mutation is large, either AOSNPS is likely to have recently detected a new global optimal solution (P_{m1} is reset to zero) or the difference between the populations is large (P_{m2} is large).

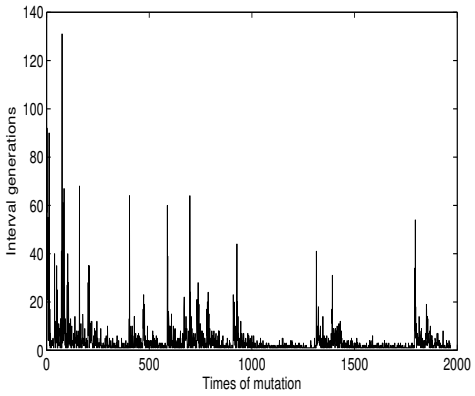

Figure 11. The interval generations between two adjacent mutations triggered

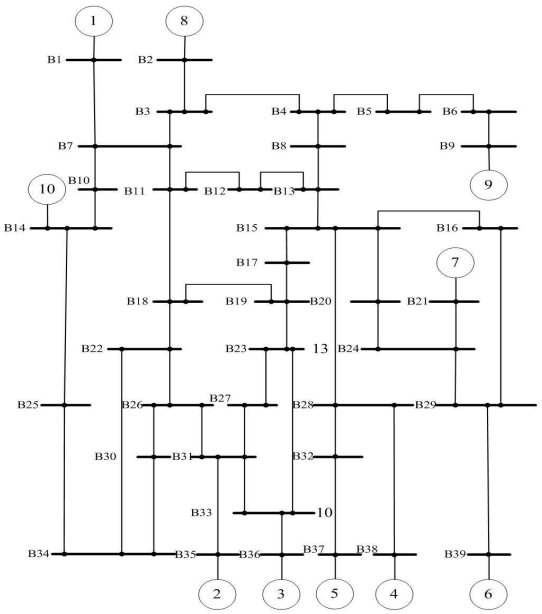

Figure 12. IEEE 39 bus power system

5. AOSNPS for Power System Fault Diagnosis

Power system fault diagnosis is the identification of location and causes of a fault on a power system. In this subsection, AOSNPS is applied to solve this fault diagnosis problem. We considered the power system represented in Fig. 12, see Ref. [63]. The power system contains 39 buses, 45 lines and 99 Circuit breakers. 39 buses and 45 lines are $B_1, ..., B_{39}$, and, $L_{1-7}, ..., L_{29-39}$, 99 Circuit breakers are $CB_{(1)-7}, ..., L_{(39)-29}$, where, $CB_{(1)-7}$ represents the breaker such that L_{1-7} is near the B_1

side. For instance, line L_{3-4} has three types of protective relays including main protective relay $L_{(3)-4m}$ and $L_{3-(4)m}$, first backup protective relay $L_{(3)-4p}$ and $L_{3-(4)p}$,and second backup protective relay $L_{(3)-4s}, L_{3-(4)s}, L_{4-(5)s}$ and $L_{4-(8)s}$.

The operational rules of the protective relays of buses and lines are described in Ref.[63] as follows.

1) Protective relays of buses. If the main protective relays of a bus operate, all breakers connected to the bus are tripped.

2) Protective relays of lines. If the main protective relays of a line operate, all breakers connected to the line are tripped. Likewise, when the main protective relays of a bus fail to operate, the first backup protective relays operate to trip all breaker connected to the line. When adjacent regions of a line fail and their main protective relay and first backup relay fail to operate, the second backup relay of a line operates.

Six fault cases in the IEEE 39 bus system and IEEE 118 bus system have been considered: case 1 is a single fault, case 2 is multiple faults, case 3 and case 4 are multiple faults with incompleteness and uncertainty in the IEEE 39 node electric power system. Case 5 is multiple faults and case 6 is multiple faults with incompleteness and uncertainty in the IEEE 118 node electric power system[6] .The network data for the four cases are listed in Table 3.

In order to identify a fault, the engineer is usually informed about the real status of protective relays and circuit breakers. We represent this piece as information as two binary vectors $\mathbf{r}$ and $\mathbf{c}$ for relays and circuit breakers, respectively

$$\mathbf{r} = (r_1, r_2, \dots r_{Nr})$$
$$\mathbf{c} = (c_1, c_2, \dots c_{Nc})$$

where for every value of i, 1) $r_i = 1$ and $r_i = 0$ represent the operation and non-operation of the protective relay i; 2) $c_i = 1$ and $c_i = 0$ represent the operation (the circuit breakers trips) and non-operation of the circuit breaker. Let us now define the binary vector $\mathbf{s}$,

$$\mathbf{s} = (s_1, s_2, \dots, s_n)$$

where s_i represents the status of section: s_i =1 and s_i =0 represent the fault status and normal status of section i, respectively.

On the basis of these pieces of information, the expected values of the protective relays and circuit breakers

$$\mathbf{re} = (re_1, re_2, \dots, re_{Nr})$$
$$\mathbf{ce} = (ce_1, ce_2, \dots, ce_{Nc})$$

are represented as functions of $\mathbf{r}$ and $\mathbf{s}$ and computed as shown in Ref. [65]. We used AOSNPS to optimise the following objective function[62]

$$e(\mathbf{s}) = \sum_{i=1}^{Nr} |r_i - re_i(\mathbf{s})| + \sum_{j=1}^{Nc} |c_j - ce_j(\mathbf{s}, \mathbf{r})|$$

Table 4 displays the results of the fault diagnosis estimation obtained by AOSNPS and compare them with those obtained by the ad-hoc expert based methods presented in Ref. [61] and [72]. It can be observed that for case 1 the three meathods reach the same results. Regarding case 2, the estimation result of AOSNPS is the same as [72], but is different to [72]. In case 3, case 4, case5 and case 6, we may observe that the results of AOSNPS are different from those in [61] and [72]. However, our results agree with those achieved by a more recent study that considers only complex scenarios, see Ref.[67].

6. Conclusions

This paper proposes a novel Adaptive Spiking Neural P System, indicated as AOSNPS, for solving the 0/1 knapsack problem. The proposed method uses multiple neural P systems, each of them generating a candidate solution and a Dynamic Guider algorithm that supervises the search. The proposed scheme includes two novel adaptive mechanisms within the Dynamic Guider algorithm: an adaptive learning rate and an adaptive mutation based on the estimation of the diversity of the sampling rules.

Numerical results demonstrate that thanks to their adaptive learning rate and adaptive mutation, the proposed AOSNPS overcomes the diversity loss limitation of its predecessor OSNPS. The novel adaptive mechanism enhances upon the behaviour of the Dynamic Guider algorithm: the search operators, through the probabilities of the spiking neural P system, react to the temporary diversity loss of the population by suggesting fresh search directions. The proposed AOSNPS outperforms two popular metaheuristics and OSNPS on the 0/1 knapsack problem in all the scenarios considered. Furthermore, the application of AOSNPS to a power distribution problem shows that AOSNPS behaves like

Table 3. Status information about four cases

case	protective relays	breakers
1	$L_{(18)-19m}$operated,$L_{18-(19)p}$ operated $L_{18-(19)m}$ refused to operated	$CB_{(18)-19}$ operated,$CB_{18-(19)}$ operated
2	$L_{(18)-19m}$ operated, $L_{18-(19)m}$ operated $L_{(17)-19m}$ operated, $L_{19-(23)s}$ operated $L_{(17)-19m}$ operated, $L_{19-(23)s}$ operated	$CB_{(18)-19}$ operated,$CB_{17-(19)}$ operated $CB_{19-(23)}$ operated $CB_{18-(19)}$ refused to operate
3	$L_{(18)-19m}$ operated, $L_{18-(19)m}$ operated B_{19m} operated, $L_{(17)-19s}$ operated $L_{19-(23)m}$ misoperation	$CB_{(18)-19}$operated, $CB_{18-(19)}$ operated $CB_{(17)-19}$ operated, $CB_{19-(23)}$ operated $CB_{17-(19)}$ refused to operated
4	$L_{(11)-12m}$ operated, $L_{11-(12)m}$ operated $L_{12-(13)m}$ operated, $L_{(18)-19p}$ operated $L_{18-(19)p}$ operated,$L_{(12)-13m}$ misoperation $L_{(18)-19m}$ refused to operated $L_{18-(19)m}$ refused to operated	$CB_{(11)-12}$ operated, $CB_{12-(13)}$ operated $CB_{(17)-19}$ operated, $CB_{17-(19)}$ operated $CB_{11-(12)}$ refused to operated
5	B_{83m} operated,$L_{(19)-20m}$ operated $L_{19-(20)p}$ operated,$L_{(21)-22p}$ operated $L_{21-(22)p}$ operated,$L_{(9)-10p}$ misoperation	$CB_{(9)-10}$ operated,$CB_{7-(8)}$ operated $CB_{(11)-12}$ operated,$CB_{(19)-20}$ operated $CB_{19-(20)}$ operated,$CB_{(21)-22}$ operated $CB_{21-(22)}$ operated $CB_{9-(10)}$ refused to operated
6	B_{85m} operated,$L_{(3)-4m}$ operated B_{86m} misoperation,$L_{3-(4)p}$ operated $L_{7-(16)m}$ misoperation $L_{3-(4)m}$ refused to operated	$CB_{(3)-4}$ operated,$CB_{3-(4)}$ operated, $CB_{(5)-6}$ operated,$CB_{(7)-10}$ operated, $CB_{(13)-14}$ operated,$CB_{15-(16)}$ operated, $CB_{(15)-16}$ refused to operated $CB_{(1)-2}$ refused to operated

Table 4. Comparisons between AOSNPS and two other methods for power grid fault detection

case	candidate sections	The results in Ref.[61]	The results in Ref.[72]	AOSNPS
1	L_{18-19}	L_{18-19}	L_{18-19}	L_{18-19}
2	B_{19}, L_{17-19}, L_{18-19}, L_{19-23}	L_{17-19},L_{18-19}	L_{18-19}	L_{18-19}
3	B_{19}, L_{17-19}, L_{18-19}, L_{19-23}	B_{19}, L_{17-19}, L_{18-19}	B_{19}, L_{17-19}, L_{18-19}	B_{19}, L_{18-19}
4	B_{12}, L_{11-12}, L_{12-13}, L_{17-19}	B_{12}, L_{12-13}, L_{17-19}	B_{12}, L_{11-12}, L_{17-19}	L_{11-12}, L_{17-19}
5	B_{83}, L_{9-10}, L_{19-20} ,L_{21-22}	B_{83},L_{9-10},L_{19-20},L_{21-22}	B_{83},L_{19-20},L_{21-22}	B_{83},L_{19-20},L_{21-22}
6	B_{85}, B_{86}, L_{3-4}, L_{1-2},L_{7-16}	B_{85}, B_{86}, L_{3-4},L_{7-16}	B_{85}, B_{86}, L_{3-4},	B_{85}, L_{3-4}

an artificial intelligence element able to correctly locate electrical faults.

Acknowledgments

This work was supported the National Natural

Science Foundation of China (61972324, 61672437, 61702428), the Sichuan Science and Technology Program (2017FZ0010, 2018GZ0185, 2018GZ0086, 20YYJC2596), New Generation Artificial Intelligence Science and Technology Major Project of Sichuan Province (2018GZDZX0043) and Artificial Intelligence Key Laboratory of Sichuan Province (2019RYJ06).

Bibliography

1. H. Adeli and A. Panakkat, A probabilistic neural network for earthquake magnitude prediction, *Neural Networks* **22**(7) (2009) 1018–1024.
2. H. Adeli and H. S. Park, Optimization of space structures by neural dynamics, *Neural Networks* **8**(5) (1995) 769–781.
3. F. Ahmadkhanlou and H. Adeli, Optimum cost design of reinforced concrete slabs using neural dynamics model, *Engineering Applications of Artificial Intelligence* **1**(1) (2005) 65–72.
4. M. Ahmadlou and H. Adeli, Enhanced probabilistic neural network with local decision circles: A robust classifier, *Integrated Computer-Aided Engineering* **17**(3) (2010) 197–210.
5. M. Atencia, G. Joya and F. Sandoval, Dynamical analysis of continuous higher-order hopfield networks for combinatorial optimization, *Neural Computation* **17**(8) (2005) 1802–1819.
6. T. Bi, Z. Yan, F. Wen, Y. Ni, F. Wu and Q. Yang, P systems-based computing polynomials with integer coefficients: Design and formal verification, *Power System Technology* **25**(11) (2001) 27–37.
7. C. M. Bishop, *Neural Networks for Pattern Recognition* (Oxford, U.K.:Oxford University, 1995).
8. C. Buiu, C. Vasile and O. Arsene, Development of membrane controllers for mobile robots, *Information Sciences* **187**(1) (2012) 33–51.
9. F. G. C. Cabarle, H. N. Adorna, M. Jiang and X. Zeng, Spiking neural P systems with scheduled synapses, *IEEE Transactions on Nanobioscience* **16**(8) (2017) 792–801.
10. Z. Cen, J. Wei and R. Jiang, A gray-box neural network-based model identification and fault estimation scheme for nonlinear dynamic systems, *International Journal of Neural Systems* **23**(6) (2013) 497–383.
11. T. K. Chau, S. Yu, T. Fernando, H. H. Iu and M. Small, A load-forecasting-based adaptive parameter optimization strategy of statcom using anns for enhancement of lfod in power systems, *IEEE Transactions on Industrial Informatics* **14**(6) (2018) 2463–2472.
12. L. Cheng, Z. Hou, Y. Lin, M. Tan, W. Zhang and F. Wu, Recurrent neural network for non-smooth convex optimization problems with application to the identification of genetic regulatory networks, *IEEE Transactions on Neural Networks* **25**(2) (2011) 714–726.
13. A. Cichocki and R. Unbehauen, Neural networks for optimization and signal processing, *John Wiley and Sons* **74**(1) (1992) 245–250.
14. G. Ciobanu, M. J. Pérez-Jiménez and G. Păun (eds.), *Applications of Membrane Computing*Natural Computing Series, Natural Computing Series (Springer, 2006).
15. D. Ciresan, U. Meier and J. Schmidhuber, Multi-column deep neural networks for image classification, *Computer Vision and Pattern Recognition* **175**(10) (2012) 3642–3649.
16. T. S. Clawson, S. Ferrari, S. B. Fuller and R. J. Wood, Spiking neural network (SNN) control of a flapping insect-scale robot, *55th IEEE Conference on Decision and Control, CDC 2016, Las Vegas, NV, USA, December 12-14, 2016*, 2016, pp. 3381–3388.
17. S. Ding, H. Li, C. Su, J. Yu and F. Jin, Evolutionary artificial neural networks: a review, *Artificial Intelligence Review* **39**(3) (2013) 251–260.
18. I. Fiete, W. Senn, C. Wang and R. Hahnloser, Spike-time-dependent plasticity and heterosynaptic competition organize networks to produce long scale-free sequences of neural activity, *Neuron* **65**(4) (2010) 563–576.
19. R. A. Fisher, *The Design of Experiments*, ninth edn. 1971 (1935).
20. J. Friedrich, R. Urbanczik and W. Senn, Code-specific learning rules improve action selection by populations of spiking neurons, *Int. J. Neural Syst.* **24**(5) (2014).
21. P. Frisco, M. Gheorghe and M. J. Pérez-Jiménez (eds.), *Applications of Membrane Computing in Systems and Synthetic Biology*Emergence, Complexity and Computation, Emergence, Complexity and Computation (Springer, 2014).
22. F. Galán-Prado, A. Morán, J. Font, M. Roca and J. L. Rosselló, Compact hardware synthesis of stochastic spiking neural networks, *Int. J. Neural Syst.* **29**(8) (2019) 1950004:1–1950004:13.
23. S. Ghosh-Dastidar and H. Adeli, A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection, *Neural Networks* **22**(10) (2009) 1419–1431.
24. S. Ghosh-Dastidar and H. Adeli, Third generation neural networks: Spiking neural networks, *Advances in Computational Intelligence*, ed. S. E. Yu W., *Advances in Intelligent and Soft Computing* **116** (Springer, 2009).
25. S. Ghosh-Dastidar and H. Adeli, Improved spiking neural networks for EEG classification and epilepsy and seizure detection, *Integr. Comput. Aided Eng.* **14**(3) (2007) 187–212.
26. S. Ghosh-Dastidar and H. Adeli, Spiking neural networks, *Int. J. Neural Syst.* **19**(4) (2009) 295–308.
27. S. Ghosh-Dastidar, H. Adeli and N. Dadmehr, Mixed-band wavelet-chaos-neural network methodology for epilepsy and epileptic seizure detection, *IEEE Transactions on Biomedical Engineering* **54**(9) (2007) 1545–1551.
28. S. Ghosh-Dastidar, H. Adeli and N. Dadmehr, Prin-

cipal component analysis-enhanced cosine radial basis function neural network for robust epilepsy and seizure detection, *IEEE Transactions on Biomedical Engineering* **55**(2) (2008) 512–518.
29. K. H. Han and J. H. Kim, Quantum-inspired evolutionary algorithm for a class of combinatorial optimization, *IEEE Transactions on Evolutionary Computation* **6**(6) (2002) 580–593.
30. K. H. Han and J. H. Kim, Quantum-inspired evolutionary algorithms with a new termination criterion, hepsilon gate, and two-phase scheme, *IEEE Transactions on Evolutionary Computation* **8**(2) (2004) 156–169.
31. S. Holm, A simple sequentially rejective multiple test procedure, *Scandinavian Journal of Statistics* **6**(2) (1979) 65–70.
32. L. Huang, X. He, N. Wang and Y. Xie, P systems based multi-objective optimization algorithm, *Progress in Natural Science* **17**(4) (2007) 458–465.
33. Hui Gao, Guanghui Xu and Zheren Wang, A novel quantum evolutionary algorithm and its application, *2006 6th World Congress on Intelligent Control and Automation*, **1**June 2006, pp. 3638–3642.
34. M. Ionescu, G. Puaun, M. J. Pérez-Jiménez and A. Rodríguez-Patón, Spiking neural P systems with several types of spikes, *Int. J. Comput. Commun. Control* **6**(4) (2011) 647–655.
35. M. Ionescu, G. Păun and T. Yokomori, Spiking neural P systems, *Fundamenta Informaticae* **71**(2-3) (2006) 279–308.
36. Y. Jiang, Y. Su and F. Luo, An improved universal spiking neural P system with generalized use of rules, *Journal of Membrane Computing* **1**(4) (2019) 270–278.
37. N. Kasabov, K. Dhoble, N. Nuntalid and G. Indiveri, Dynamic evolving spiking neural networks for online spatio-and spectro-temporal pattern recognition, *Neural Networks* **41**(5) (2013) 188–201.
38. M. Kociecki and H. Adeli, Two-phase genetic algorithm for size optimization of free form steel spaceframe roof structures, *Journal of Constructional Steel Research* **90**(9) (2013) 283–296.
39. Kuk-Hyun Han and Jong-Hwan Kim, Genetic quantum algorithm and its application to combinatorial optimization problem, *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, **2**July 2000, pp. 1354–1360 vol.2.
40. W. Maass, Lower bounds for the computational power of networks of spiking neurons, *Neural Computation* **8**(1) (1996) 1–40.
41. W. Maass, Networks of spiking neurons: the third generation of neural network models, *Neural Networks* **10**(9) (1997) 1659–1671.
42. L. Maguire, T. McGinnity, B. Glackin, A. Ghani, A. Belatreche and J. Harkin, Challenges for large-scale implementations of spiking neural networks on fpgas, *Neurocomputing* **71**(1) (2007) 13 – 29, Dedicated Hardware Architectures for Intelligent Systems Advances on Neural Networks for Speech and Audio Processing.
43. V. Manca, Metabolic computing, *Journal of Membrane Computing* **1**(3) (2019) 223–232.
44. C. Martín-Vide, G. Păun, J. Pazos and A. Rodríguez-Patón, Tissue p systems, *Theoretical Computer Science* **296**(2) (2003) 295 – 326, Machines, Computations and Universality.
45. P. Mazumder, D. Hu, I. E. Ebong, X. Zhang, Z. Xu and S. Ferrari, Digital implementation of a virtual insect trained by spike-timing dependent plasticity, *Integration* **54** (2016) 109–117.
46. F. Neri, V. Tirronen, T. Karkkainen and T. Rossi, Fitness diversity based adaptation in multimeme algorithms:a comparative study, *2007 IEEE Congress on Evolutionary Computation*, 2007, pp. 2374–2381.
47. A. Ortiz, J. Munilla, J. M. Górriz and J. Ramírez, Ensembles of deep learning architectures for the early diagnosis of the alzheimer's disease, *Int. J. Neural Syst.* **26**(7) (2016) 1650025:1–1650025:23.
48. L. Pan, G. Păun, G. Zhang and F. Neri, Spiking neural p systems with communication on request, *International Journal of neural systems* **27**(08) (2017).
49. L. Pan and G. Puaun, Spiking neural P systems with anti-spikes, *Int. J. Comput. Commun. Control* **4**(3) (2009) 273–282.
50. L. Pan, T. Wu, Y. Su and A. V. Vasilakos, Cell-like spiking neural p systems with request rules, *IEEE Transactions on Nanobioscience* **16**(6) (2017) 513–522.
51. A. Panakkat and H. Adeli, Neural network models for earthquake magnitude prediction using multiple seismicity indicators, *International Journal of Neural Systems* **17**(1) (2007) 13–33.
52. A. Panakkat and H. Adeli, Recurrent neural network for approximate earthquake time and location prediction using multiple seismicity indicators, *Computer-Aided Civil and Infrastructure Engineering* **24**(4) (2009) 280–292.
53. S. Park, N. G. Laxpati, C. Gutekunst, M. J. Connolly, J. Tung, K. Berglund, B. Mahmoudi and R. E. Gross, A machine learning approach to characterize the modulation of the hippocampal rhythms via optogenetic stimulation of the medial septum, *Int. J. Neural Syst.* **29**(10) (2019) 1950020:1–1950020:21.
54. G. Păun, Computing with membranes, *Journal of Computer and System Sciences* **61**(1) (2000) 108–143.
55. W. Pitts and W. McCulloch, A logical calculus of the ideas immanent in nervous activity, *Bull. Math. Biophys* **5** (1943) 115–133.
56. F. Ponulak and A. Kasinski, Introduction to spiking neural networks: information processing, learning and applications, *Acta Neurobiologiae Experimentalis* **71**(4) (2011) 409–433.
57. J. L. Rosselló, V. Canals, A. Morro and A. Oliver, Hardware implementation of stochastic spiking neural networks, *Int. J. Neural Syst.* **22**(4) (2012).
58. T. Song, A. Rodríguez-Patón, P. Zheng and X. Zeng, Spiking neural P systems with colored spikes, *IEEE Transactions on Cognitive and Development Systems*

10(4) (2018) 1106–1115.
59. T. Song, X. Zeng, P. Zheng, M. Jiang and A. Rodríguez-Patón, A parallel workflow pattern modelling using spiking neural p systems with colored spikes, *IEEE Transactions on Nanobioscience* **17**(4) (2018) 474–484.
60. S. Stefan, K. Nikola and D.-P. Michael, On the probabilistic optimization of spiking neural networks, *International Journal of Neural Systems* **20**(6) (2010) 481–500.
61. J. Sun, S. Qin and Y. Song, Fault diagnosis of electric power systems based on fuzzy petri nets, *IEEE Transactions on Power Systems* **19**(4) (2004) 2053–2059.
62. T. Wang, S. Zeng, G. Zhang, M. J. Pérez-Jiménez and J. Wang, Fault section estimation of power systems with optimization spiking neural P systems, *Romanian Journal of Information Science and Technology* **9**(6) (2014) 786–799.
63. T. Wang, G. Zhang, J. Zhao, Z. He, J. Wang and M. J. Pérez-Jiménez, Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural P systems, *IEEE Transactions on Power Systems* **30**(3) (2015) 1182–1194.
64. X. Wang, G. Zhang, F. Neri, T. Jiang, J. Zhao, M. Gheorghe, F. Ipate and R. Lefticaru, Design and implementation of membrane controllers for trajectory tracking of nonholonomic wheeled mobile robots, *Integrated Computer Aided Engineering* **23**(1) (2015) 15–30.
65. F. Wen, Y. Qian, L. T. Z. Han, J. Shi and H. Zhang, A tabu search based approach to fault section estimation and state identification of unobserved protective relays in power systems using information from protective relays and circuit breakers, *Proceedings of the CSEE* **13**(5) (1998) 1000–6753.
66. F. Wilcoxon, Individual comparisons by ranking methods, *Biometrics Bulletin* **1**(6) (1945) 80–83.
67. K. Wu, F. Wen, Y. Xue, H. Zhou and X. Li, Fault diagnosis model of time-delay constrained weighted fuzzy petri nets based on multi-source information, *Automation of electric power system* **37**(24) (2013) 43–53.
68. T. Wu, Y. Wang, S. Jiang, Y. Su and X. Shi, Spiking neural P systems with rules on synapses and anti-spikes, *Theoretical Computer Science* **16**(8) (2017) 888–895.
69. T. Wu, F.-D. Bilbie, A. Păun, L. Pan and F. Neri, Simplified and yet turing universal spiking neural p systems with communication on request, *International Journal of Neural Systems* **28**(8) (2018).
70. T. Wu, A. Păun, Z. Zhang and L. Pan, Spiking neural P systems with polarizations, *IEEE Transactions on Neural Networks and Learning Systems* **29**(8) (2018) 3349–3360.
71. T. Wu, Z. Zhang, G. Păun and L. Pan, Cell-like spiking neural p systems, *Theoretical Computer Science* **623** (2016) 180–189.
72. J. Yang and Z. He, Power system fault diagnosis approach based on time sequence fuzzy petri net, *Automation of Electric Power Systems* **35**(15) (2011) 46–51.
73. X. Zeng, L. Pan and M. J. Pérez-Jiménez, Small universal simple spiking neural p systems with weights, *Science China Information Sciences* **57**(9) (2014) 1–11.
74. G. Zhang, M. J. Pérez-Jiménez and M. Gheorghe, *Real-life Applications with Membrane Computing*Emergence, Complexity and Computation, Emergence, Complexity and Computation (Springer, 2017).
75. G. Zhang, J. Cheng and M. Gheorghe, Dynamic behavior analysis of membrane-inspired evolutionary algorithms, *International Journal of Computers, Communications and Control* **9**(2) (2014) 227–242.
76. G. Zhang, J. Cheng, M. Gheorghe and Q. Meng, A hybrid approach based on differential evolution and tissue membrane systems for solving constrained manufacturing parameter optimization problems, *Applied Soft Computing* **13**(3) (2013) 1528–1542.
77. G. Zhang, M. Gheorghe, L. Pan and M. J. Pérez-Jiménez, Evolutionary membrane computing: A comprehensive survey and new results, *Information Sciences* **279** (2014) 528–551.
78. G. Zhang, H. Rong, F. Neri and M. J. Pérez-Jiménez, An optimization spiking neural P system for approximately solving combinatorial optimization problems, *International Journal of Neural Systems* **24**(5) (2014) 1440006:01–16.
79. X. Zhang, G. Foderaro, C. Henriquez and S. Ferrari, A scalable weight-free learning algorithm for regulatory control of cell activity in spiking neuronal networks, *Int. J. Neural Syst.* **28**(2) (2018) 1750015:1–1750015:20.
80. X. Zhang, G. Foderaro, C. S. Henriquez, A. M. J. VanDongen and S. Ferrari, A radial basis function spike model for indirect learning via integrate-and-fire sampling and reconstruction techniques, *Adv. Artificial Neural Systems* **2012** (2012) 713581:1–713581:16.
81. X. Zhang, Z. Xu, C. Henriquez and S. Ferrari, Spike-based indirect training of a spiking neural network-controlled virtual insect, *Proceedings of the 52nd IEEE Conference on Decision and Control, CDC 2013, December 10-13, 2013, Firenze, Italy*, 2013, pp. 6798–6805.
82. Y. Zhang, G. Zhou, J. Jin, X. Wang and A. Cichocki, Frequency recognition in ssvep-based BCI using multiset canonical correlation analysis, *Int. J. Neural Syst.* **24**(4) (2014).