An Adaptive Memetic P System to Solve the 0/1 Knapsack Problem

Jianping Dong, Haina Rong
School of Electrical Engineering
Southwest Jiaotong University
Chengdu, China
1969140573@qq.com; ronghaina@126.com

Ferrante Neri
COL Lab, School of Computer Science
University of Nottingham
Nottingham, United Kingdom
ferrante.neri@nottingham.ac.uk

Qiang Yang, Ming Zhu
School of Control Engineering
Chengdu University of information Technology
Chengdu, China
yuxia2008@126.com, zhuming@cuit.edu.cn

Gexiang Zhang
College of Information Science and Technology
Chengdu University of Technology
Chengdu, China
zhgxdylan@126.com

***Abstract*—Memetic Algorithms are traditionally composed of an evolutionary framework and one or more local search elements. However, modern generation Memetic Algorithms do not necessarily follow a pre-established scheme and are hybrid structures of various types. By following these modern trends, the present paper proposes an original and unconventional adaptive memetic structure generated by the hybridisation of a set of theoretical computational models, namely P Systems, and an evolutionary algorithm employing adaptation rules and moving operators inspired by Evolution Strategies. The resulting memetic algorithm, namely Adaptive Optimisation Spiking Neural P System (AOSNPS), is a tailored algorithm to solve optimisation problems with binary encoding.**

More specifically AOSNPS is composed of a family of parallel spiking neural P systems, each of them generating a binary vector representing a candidate solution on the basis of internal probability parameters and an adaptive Evolutionary Guider Algorithm that evolves the probabilities encoded in each P system. Numerical result shows that the proposed approach is effective to solve the 0/1 knapsack problem and outperforms various algorithms proposed in the literature to solve the same class of problems.

***Index Terms*—Memetic Algorithms, P Systems, Membrane Computing, Evolutionary Algorithms, Knapsack Problem**

I. Introduction

In their early implementations, Memetic Algorithms (MAs) were identified with an evolutionary algorithm employing a local search [1], [2]. Subsequently, MAs no longer referred to a specific algorithmic structure but progressively became a strategy about the design of algorithms by using multiple and diverse search operators, see [3].

This fact has been emphasised in the review paper [4] which categorises MAs into three major groups

- Simple Hybrids: this category includes hybrid algorithms generated by two or more algorithms (usually one global and at least one local) combined together in a synergistic manner, see [1]. Some hybridisations belonging to this category can be very efficient to address specific domains, see e.g. [5]–[8].
- Adaptive Hybrids: this category includes hybrid algorithms where multiple local search algorithms are coordinated by a supervising adaptive mechanism e.g. performance-based like hyperheuristics [9] and meta-Lamarckian learning [10], diversity-based [11], [12] or self-adaptation, see [13].
- (Future) Memetic Automation: this category is a visionary idea that sees memetic algorithms fully self-generated by machines as a combination of "agents", see e.g. [14], [15]. Although this design approach is still under investigation, some interesting domain-specific frameworks [16], [17] and prototypes [18] have been proposed.

Thus, simple and adaptive hybrids tend to have a prearranged structure composed of an evolutionary framework and one or more local searches (possibly adaptively activated) that enhance upon the performance of one solution [19]. Conversely, modern MAs are thought as algorithms resulting from any hybridisation as long as they are capable to efficiently address a class of problems.

By following this consideration, this paper proposes an MA based on an unconventional hybridisation to solve binary problems and focuses on the 0/1 knapsack problem. More specifically, the proposed algorithm makes use of an adaptive evolutionary framework which includes an adaptive variation operator inspired by Evolutionary Strategy (ES) and another mutation operator that re-samples the points on the basis of the diversity and historical data. This evolutionary framework, namely Evolutionary Guider Algorithm, does not vary the solutions of the knapsack problem but varies the probabilities of the neurons of a family of Spiking Neural P Systems (SNPSs) [20], [21]. A SNPS is a general purpose computational structure belonging to the field of Membrane Computing, see [22]–[24]. In the context of the proposed MA, the SNPSs arranged in a family are used as number generators

to generate strings of binary numbers (the candidate solutions of the 0/1 knapsack problem). Hence, the Evolutionary Guider Algorithm modifies and detects the parameters that control the generation of the candidate solutions.

The resulting algorithm namely Adaptive Optimisation Spiking Neural P System (AOSNPS) enhances a previous simpler algorithm based on a family of P systems proposed in [25].

The remainder of this article is organised in the following way. Section II briefly introduces the 0/1 knapsack problem, provides the basic definition of P systems (Subsections II-A) and describes the algorithm in [25] (Subsections II-B and II-C). Section III describes the proposed AOSNPS. Section IV displays the numerical results of the proposed approach in different scenarios and against multiple algorithms used to solve this problem. Finally, Section V provides the conclusions of this study.

II. Background: 0/1 Knapsack problem and Optimisation by P Systems

In this paper we aim at solving the 0/1 Knapsack Problem which is formulated as follows. 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 $j = 1 \dots m$, we may represent the selection of the items as a vector

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

of binary numbers.

Then, the 0/1 Knapsack problem can be expressed as the maximisation 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 the following subsections, we briefly summarise the relevant theory about P systems and present the Optimisation Spiking Neural P System (OSNPS) introduced in [25] to solve the 0/1 knapsack problem.

A. Spiking Neural P System

The SNPS [20], [26] is an automaton represented by the tuple

$$\Pi = (O, \sigma_1, \cdots, \sigma_m, syn, \sigma_{out})$$

where

1) $O = \{a\}$ is the singleton alphabet, a is called *spike*
2) $\sigma_1, \cdots, \sigma_m$ are neurons, where each neuron σ_i is a pair
$$\sigma_i = (n_i, R_i)$$
with
   (a) $n_i \geq 0$ initial number of spikes contained in σ_i, e.g. if the neuron σ_i has an initial numbers of spikes $n_i = 5$, it contains the work $aaaaa = a^5$
   (b) R_i set of the two following rules:
      (i) **spiking rule**: $E/a^c \to a; d$ where E is a regular expression over O, and $c \geq 1, d \geq 0$. The spiking rule means that a neuron containing a word a^{n_i} may lose $c < n_i$ spikes after a delay of d steps
      (ii) **forgetting rule**: $a^s \to \lambda$, for some $s \geq 1$, with the restriction that for each rule $E/a^c \to a; d$ of type (i) from R_i, we have $a^s \notin L(E)$. The forgetting rule means that a neuron can lose all its s spikes as long as a^s does not belong to the language $L(E)$ identified by the regular expression E.
3) the connections between neurons are described by the synapses syn where $syn \subseteq \{\sigma_1, \dots \sigma_m\} \times \{\sigma_1, \dots \sigma_m\}$ with $(\sigma_i, \sigma_i) \notin syn$ for $i = 1, \dots, m$. The synapses govern the propagation of the spikes, if the neuron σ_i spikes and $(\sigma_i, \sigma_j) \in syn$ then the same spiking action is repeated on the neuron σ_j
4) the output neuron $\sigma_{out} \in \{\sigma_1, \dots \sigma_m\}$.

The distinguishing feature of SNPS is that an input causes a nondeterministic spike train in the output, see [27]. If the output neuron spikes, then we code this behaviour as a 1 and otherwise we code it as a 0. Hence, the spike train can be represented by a sequence of ones and zeros. Details about the architecture of SNPS are described and discussed in detail in [20], [25], [28].

B. Extended Spiking Neural P System

In [25], in order to exploit the potentials of SNPSs for addressing an optimisation problem, the SNPS has been extended and slightly modified. The resulting implementation of P system, namely Extended Spiking Neural P System (ESNPS) is a tuple of the type

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

where

1) $O = \{a\}$ is the singleton alphabet, a is called *spike*
2) $\sigma_1, \dots, \sigma_m$ 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 ($n_i = 1, \forall i$)
(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 σ_1, σ_2, ..., σ_m.
4) $syn = \{(\sigma_i, \sigma_j) | (1 \leq i \leq m+1$ **AND** $j = m+2)$ **OR** $(i = m+2$ **AND** $j = m+1)\}$.
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 an ESNPS is shown in Fig. 1.

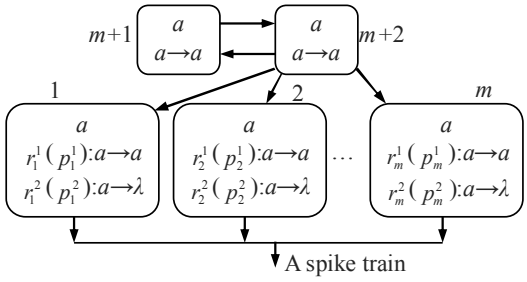

Fig. 1. Logical and functioning scheme of ESNPS

C. Optimisation 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.

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 $\mathbf{P_R} = [p_{ij}^1]_{H \times m}$, which is composed of the spiking rule probabilities of H ESNPSs, i.e.,

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

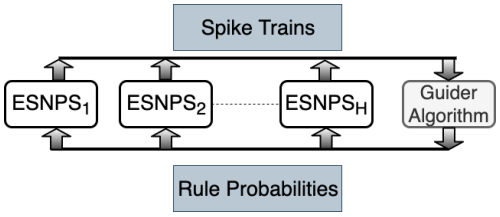

Fig. 2. Structure of OSNPS (and AOSNPS)

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

Algorithm 1 provides the details of the Guider Algorithm used in OSNPS [25]. The basic idea is that one current best solution (binary vector)

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

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 that measures the value/profit of the knapsack. At each objective function call, in order to handle the case when the total weight of all selected items exceeds the capacity C, we implemented the random chromosome repair technique suggested in [29]–[31].

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 Algorithm 1. More details on OSNPS can be found in [25].

III. Adaptive Optimisation Spiking Neural P System

The proposed AOSNPS employs the same structure of OSNPS depicted in Fig. 2 but hybridises the theoretical computer science logic of ESNPSs with that of Evolutionary Algorithms (EAs) [32] to manipulate the probability matrix $\mathbf{P_R}$. More specifically, the Guider Algorithm is an EA embedding two elements inspired by Evolution Strategies (ES) [33], [34] within it: 1) an adaptive learning rate Δ; 2) an adaptive mutation. The novel mechanisms are described in Subsections III-A and III-B, respectively. The new Guider Algorithm embedding these two elements is outlined in Subsection III-C.

A. 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 [25], Δ is a random number between 0.005 and 0.02 set at the beginning of the optimisation and kept constant throughout the OSNPS execution.

Algorithm 1 The Guider Algorithm of OSNPS

Require: 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 $(f\,(\mathbf{x_{k_1}}) > f\,(\mathbf{x_{k_2}}))$ **then**
        $x_j = x_{k_1}$
      else
        $x_j = x_{x_2}$
      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

Ensure: Rule probability matrix $\mathbf{P_R}$

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 II-B.

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 in 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.

B. 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 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 [11], [12]. 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 $[0\ \ 1]$.

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 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 to the mutation.

C. The Evolutionary Guider of Adaptive Optimisation Spiking Neural P System

Based on the adaptive learning rate and the adaptive mutation rule, the Guider Algorithm is an EA operating on the learning probabilities to support the generation of successful solution for the 0/1 knapsack 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 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]–[31].

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 and described in Subsection III-A, the re-initialisation of the neuron probability which is triggered only by the satisfaction of the conditions described in Subsection III-B. Hence, we may consider AOSNPS as a co-evolutionary memetic algorithm, see [8], [35], [36], where the candidate solutions of the 0/1 knapsack problem evolve with the probabilities of the neurons in each of the P systems. Each P system can be seen as an a local search algorithm (each of them depends on different probabilities). In this unconventional memetic algorithm, the probabilities are directly manipulated by an adaptive EA, the Evolutionary Guider Algorithm, while the fitness refers only to the solutions generated by the P systems. The latter have the role of searching the decision space on the basis of the probabilities determined by the Evolutionary Guider Algorithms.

The pseudocode of the proposed Evolutionary Guider is shown in Algorithm 2.

IV. Numerical Results

In order to validate the proposed AOSNPS, we tested it against the following algorithms designed/used in the literature to solve the 0/1 kanpsack problem

- Genetic Quantum Algorithm (GQA) [37]
- Novel Quantum Evolutionary Algorithm (NQEA) [38]
- Optimisation Spiking Neural P System (OSNPS) [25]

For each algorithm we used the recommended parameter setting used in the original paper. The proposed AOSNPS has been run with $H = 50$ ESNPS with 1002 neurons each. 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

Algorithm 2 The Evolutionary Guider Algorithm of AOSNPS

Require: 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 $(f(\mathbf{x_{k_1}}) > f(\mathbf{x_{k_2}}))$ **then**
          $x_j = x_{k_1}$
        else
          $x_j = x_{k_2}$
        end if
        {**Adaptive Learning Rate**}
        if $(x_j == 1)$ **then**
          $p_{ij}^1 = 0.5 + 0.5p_{ij}^1$
        else
          $p_{ij}^1 = 0.5p_{ij}^1$
        end if
      else
        if $(x_j^{max} == 1)$ **then**
          $p_{ij}^1 = 0.5 + 0.5p_{ij}^1$
        else
          $p_{ij}^1 = 0.5p_{ij}^1$
        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

Ensure: Rule probability matrix $\mathbf{P_R}$

runs. Each run has been continued for 400000 function calls, that 8000 generations for the AOSNPS. Table I 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 [39]. A "+" indicates that AOSNPS significantly outperforms competitor, a "-" indicates that the competitor significantly outperforms AOSNPS, and a "=" indicates that there is no significant difference in performance.

Numerical results in Table I show that the proposed memetic implementation is very efficient to address the 0/1 knapsack problem since AOSNPS 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 Evolutionary Guider 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 [40] 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 [41]:

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

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

The Holm-Bonferroni procedure in Table II 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 I

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

TABLE II

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

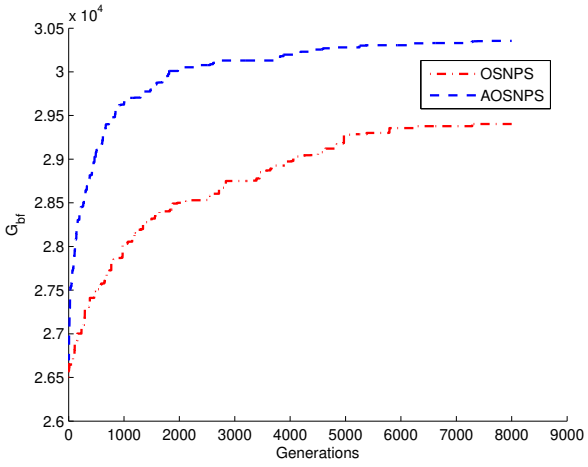

Fig. 3. Average performance trend (best cost detected G_{bf} over the generations) for OSNPS and AOSNPS.

In order to emphasise the advantages of the proposed memetic approach over the version [25], we present in Fig. 3 the performance trend of OSNPS and AOSNPS. The trends clearly show the superiority of the newly proposed AOSNPS throughout the entire evolution.

Finally, in order to illustrate how the concept of convergence can be exported to the context of P systems, we plotted the average Hamming distance D_{hm} calculated over all binary strings produced by the H ESNPSs at each generation gen:

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

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; H is the number of individuals (number of ESNPSs); the symbol $\oplus$ represents the OR operator.

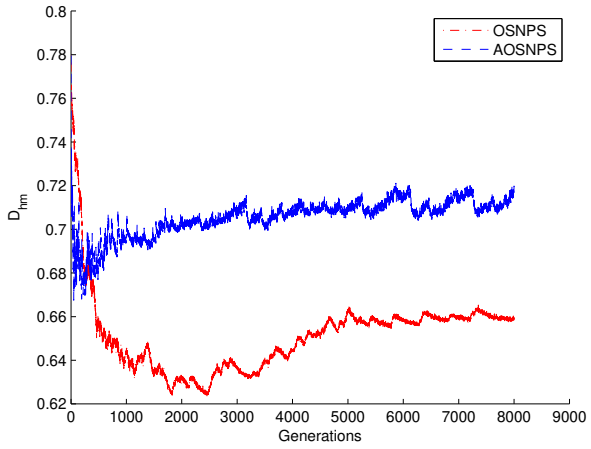

Fig. 4. Average Hamming distance D_{hm} (measurement of the population diversity) for OSNPS and AOSNPS.

It can be noticed that a large value of D_{hm} indicates a high variety between each pair of (binary) candidate solutions in a population. The plot in Fig. 4 shows that AOSNPS achieves and maintains much higher population diversity values than OSNPS. Higher diversity values appear to be beneficial to overcome suboptimal basins of attraction and enhance upon the current best solution.

V. CONCLUSION

This paper proposes a novel MA generated by the combination multiple spiking neural P systems and an adaptive Evolutionary Guider Algorithm for solving the 0/1 knapsack problem. Each neural P systems generates a (binary) candidate solution on the basis of the probabilities encoded in its neurons while the Evolutionary Guider Algorithm supervises the search by optimising the functioning of the spiking neural P systems.

The resulting algorithm, Adaptive Optimisation Spiking Neural P Systems (AOSNPS), has been thoroughly tested over multiple problem instances and against various algorithms used to solve the 0/1 knapsack problems. Numerical results show that this unconventional approach is indeed powerful and

is able to outperform its competitors for most of the problems under consideration. Furthermore, a direct comparison with another algorithm based on a similar structure shows the superiority of AOSNPS in terms of both solution detected and population diversity.

A further important contribution of this study is the novel memetic logic that successfully combines theoretical computer science and metaheuristic optimisation.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (61972324, 61672437, 61702428), by the Sichuan Science and Technology Program (2017FZ0010, 2018GZ0185, 2018GZ0086), New Generation Artificial Intelligence Science and Technology Major Project of Sichuan Province (2018GZDZX0044) and Artificial Intelligence Key Laboratory of Sichuan Province (2019RYJ06).

REFERENCES

[1] P. Moscato and M. Norman, "A Competitive and Cooperative Approach to Complex Combinatorial Search," Tech. Rep. 790, 1989.

[2] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: A literature review," *Swarm and Evolutionary Computation*, vol. 2, pp. 1–14, 2012.

[3] Y. Ong, M. H. Lim, and X. Chen, "Memetic computation—past, present future [research frontier]," *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, pp. 24–31, May 2010.

[4] X. Chen, Y. Ong, M. Lim, and K. C. Tan, "A multi-facet survey on memetic computation," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 591–607, Oct 2011.

[5] K. Tang, Y. Mei, and X. Yao, "Memetic algorithm with extended neighborhood search for capacitated arc routing problems," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1151–1166, Oct 2009.

[6] L. Ma, J. Li, Q. Lin, M. Gong, C. A. Coello Coello, and Z. Ming, "Cost-aware robust control of signed networks by using a memetic algorithm," *IEEE Transactions on Cybernetics*, pp. 1–14, 2020, to appear.

[7] C. Ting, R. Liaw, T. Wang, and T. Hong, "Mining fuzzy association rules using a memetic algorithm based on structure representation," *Memetic Computing*, vol. 10, no. 1, pp. 15–28, 2018.

[8] X. Ma, X. Li, Q. Zhang, K. Tang, Z. Liang, W. Xie, and Z. Zhu, "A survey on cooperative co-evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 3, pp. 421–441, June 2019.

[9] E. Özcan, B. Bilgin, and E. E. Korkmaz, "A comprehensive analysis of hyper-heuristics," *Intell. Data Anal.*, vol. 12, no. 1, pp. 3–23, Jan. 2008.

[10] M. N. Le, Y. S. Ong, Y. Jin, and B. Sendhoff, "Lamarckian memetic algorithms: local optimum and connectivity structure analysis," *Memetic Computing Journal*, vol. 1, no. 3, pp. 175–190, 2009.

[11] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore, and M. Sumner, "A fast adaptive memetic algorithm for on-line and off-line control design of pmsm drives," *IEEE Transactions on System Man and Cybernetics-part B, special issue on Memetic Algorithms*, vol. 37, no. 1, pp. 28–41, 2007.

[12] F. Neri, V. Tirronen, T. Kärkkäinen, and T. Rossi, "Fitness diversity based adaptation in multimeme algorithms:a comparative study," in *2007 IEEE Congress on Evolutionary Computation*, 2007, pp. 2374–2381.

[13] Q. H. Nguyen, Y. S. Ong, L. M. Hiot, and N. Krasnogor, "Adaptive Cellular Memetic Algorithms no access," *Evolutionary Computation*, vol. 17, no. 2, pp. 231–256, 2009.

[14] G. Acampora, V. Loia, and M. Gaeta, "Exploring e-learning knowledge through ontological memetic agents," *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, pp. 66–77, May 2010.

[15] Z. Zhu, S. Jia, and Z. Ji, "Towards a memetic feature selection paradigm [application notes]," *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, pp. 41–53, May 2010.

[16] L. Feng, Y. Ong, M. Lim, and I. W. Tsang, "Memetic search with interdomain learning: A realization between cvrp and carp," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 644–658, Oct 2015.

[17] Z. Zhu, J. Xiao, S. He, Z. Ji, and Y. Sun, "A multi-objective memetic algorithm based on locality-sensitive hashing for one-to-many-to-one dynamic pickup-and-delivery problem," *Information Sciences*, vol. 329, pp. 73 – 89, 2016, special issue on Discovery Science.

[18] F. Caraffini, F. Neri, and L. Picinali, "An analysis on separability for memetic computing automatic design," *Information Sciences*, vol. 265, pp. 1–22, 2014.

[19] W. E. Hart, N. Krasnogor, and J. E. Smith, "Memetic Evolutionary Algorithms," in *Recent Advances in Memetic Algorithms*, W. E. Hart, N. Krasnogor, and J. E. Smith, Eds. Berlin, Germany: Springer, 2004, pp. 3–27.

[20] M. Ionescu, G. Păun, and T. Yokomori, "Spiking neural P systems," *Fundamenta Informaticae*, vol. 71, no. 2-3, pp. 279–308, 2006.

[21] 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*, vol. 30, no. 3, pp. 1182–1194, 2015.

[22] G. Păun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108–143, 2000.

[23] G. Zhang, M. Gheorghe, L. Pan, and M. J. Pérez-Jiménez, "Evolutionary membrane computing: A comprehensive survey and new results," *Information Sciences*, vol. 279, pp. 528–551, 2014.

[24] G. Zhang, M. J. Pérez-Jiménez, and M. Gheorghe, *Real-life Applications with Membrane Computing*, ser. Emergence, Complexity and Computation. Springer, 2017.

[25] 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*, vol. 24, no. 5, pp. 1 440 006:01–16, 2014.

[26] Y. Jiang, Y. Su, and F. Luo, "An improved universal spiking neural P system with generalized use of rules," *Journal of Membrane Computing*, vol. 1, no. 4, pp. 270–278, 2019.

[27] J. Wang, P. Shi, H. Peng, M. J. Pérez-Jiménez, and T. Wang, "Weighted fuzzy spiking neural p systems," *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 2, pp. 209–220, April 2013.

[28] V. Manca, "Metabolic computing," *Journal of Membrane Computing*, vol. 1, no. 3, pp. 223–232, 2019.

[29] K. H. Han and J. H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 580–593, 2002.

[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*, vol. 8, no. 2, pp. 156–169, 2004.

[31] G. Zhang, J. Cheng, and M. Gheorghe, "Dynamic behavior analysis of membrane-inspired evolutionary algorithms," *International Journal of Computers, Communications and Control*, vol. 9, no. 2, pp. 227–242, 2014.

[32] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computation*. Berlin, Germany: Springer-Verlag, 2003.

[33] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies – a comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[34] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation," in *Proceedings of IEEE International Conference on Evolutionary Computation, 1996.* IEEE, 1996, pp. 312–317.

[35] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: model, taxonomy, and design issues," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 474–488, 2005.

[36] J. E. Smith, "Coevolving memetic algorithms: A review and progress report," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 37, no. 1, pp. 6–17, 2007.

[37] K.-H. Han and J.-H. Kim, "Genetic quantum algorithm and its application to combinatorial optimization problem," in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, vol. 2, July 2000, pp. 1354–1360 vol.2.

[38] H. Gao, G. Xu, and Z. Wang, "A novel quantum evolutionary algorithm and its application," in *2006 6th World Congress on Intelligent Control and Automation*, vol. 1, June 2006, pp. 3638–3642.

[39] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[40] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65–70, 1979.

[41] R. A. Fisher, *The Design of Experiments*, 9th ed., 1971 (1935).