Optimization Spiking Neural P System for Solving TSP

Feng Qi and Mengmeng Liu(✉)

Shandong Normal University, Jinan, China
qfsdnu@126.com, 1783797657@qq.com

Abstract. Spiking neural P systems are a class of distributed and parallel computing models that incorporate the idea of spiking neurons into P systems. Membrane computing (MC) combining with evolutionary computing (EC) is called evolutionary MC. In this work, we will combine SNPS with heuristic algorithm to solve the travelling salesman problem. To this aim, an extended spiking neural P system (ESNPS) has been proposed. A certain number of ESNPS can be organized into OSNPS. Extensive experiments on TSP have been reported to experimentally prove the viability and effectiveness of the proposed neural system.

Keywords: OSNPS · GA · Membrane algorithm · TSP

1 Introduction

Membrane computing is one of the recent branches of natural computing. The obtained models are distributed and parallel computing devices, usually called P systems. There are three main classes of P systems investigated: cell-like P systems [1], tissue-like P systems [2] and neural-like P systems [3]. Spiking Neural P system (SNPS, for short) is a class of neural-like P systems, which are inspired by the method of biological neuron processing information and communicating with others by means of electrical spikes. Evolutionary computing (EC) is based in Darwin's theory of evolution, simulating the evolution process and structuring a kind of heuristic optimization algorithms with characteristics of self-organization, adaptive and self-learning, such as genetic algorithm, ant colony optimization, particle swarm optimization and so on.

MC combining with EC is called evolutionary membrane computing [4], in which the membrane algorithm is a research direction. Membrane algorithm is a kind of hybrid optimization algorithm which combines the structure of membrane system, evolution rules, calculation mechanism and the principle of evolutionary computation.

The research on the membrane algorithm can be dated back to 2004 and Nishida combined a membrane structure with the way of tabu search to solve the traveling salesman problems [5]. In 2008, a one-level membrane structure combining with a quantum-inspired evolutionary algorithm was put forward to solve knapsack problems [6]. In 2013, a tissue membrane system was used to solve parameter optimization problems [7]. These investigations indicate the feasibility of the P systems for multifarious optimization problems. But, at present, the membrane algorithm is mainly focused on the cell-like P system and tissue-like P system. The research of membrane algorithm on the neural-like P system is relatively few. In 2014, Professor Zhang designed an optimization spiking neural P system [8], which can be used to solve the knapsack problem-a famous NP complete problem. The results show that the design optimization spiking neural P system has obvious advantages in solving knapsack problems.

The Traveling Salesman Problem (TSP) is a widely studied NP-hard combinatorial problem, and it's famous for being difficult to solve. So, it is meaningful both in theory and applications to develop techniques to solve such problems. In this paper, we combine SNPS and genetic algorithm (GA) to solve the TSP. First, we design the optimization SNPS (OSNPS), achieving the connection between the GA algorithm and the membrane system. Second, we implement our ideas on the platform MATLAB. The ideas of this article not only contribute to the membrane algorithm of the neural-like P system, but also find a new way to solve the TSP.

2 Related Background

Generally, an SNP system is composed of neurons, spikes, synapses, and rules. Neurons may contain a number of spikes, spiking rules and forgetting rules, and directed connections between neurons and neurons are accomplished by synapses. A neuron can send information to its neighboring neurons by using the spiking rule. By using the forgetting rule, a number of spikes will be removed from the neuron, and thus they are removed from the system.

An SNP system of degree $m \geq 1$ is a construct of the form:

$$\prod = (O, \sigma_1, \sigma_2, \cdots \sigma_m, \text{syn}, \text{in}, \text{out})$$

Where

- $o = \{a\}$ is the alphabet, a is spike;
- $\sigma_1, \sigma_2, \cdots \sigma_m$ are neurons of the form $\sigma_i = (n_i, R)$ with $1 \leq i \leq m$. n_i is a natural number representing the initial number of spikes in neuron σ_i; R is set of rules in each neuron of the following forms:
  - (a) $E/a^c \rightarrow a; d$ is the spiking rule, where E is the regular expression over $\{a\}$; (c and d are integer and $c \geq 1, d \geq 0$)
  - (b) $a^s \rightarrow \lambda$ is the forgetting rule, with the restriction that for any $s \geq 1$ and any spiking rule $E/a^c \rightarrow a; d, a^s \notin L(E)$, where L(E) is set of regular languages associated with regular expression E and λ is the empty string;
- $\text{syn} \subseteq \{1, 2, ..., m\} \times \{1, 2, \ldots, m\}$ is set of synapses between neurons, where $i \neq j$, $z \neq 0$ for each $(i, j) \in \text{syn}$, and for each $(i, j) \in \{1, 2, ..., m\} \times \{1, 2, ..., m\}$ there is at most one synapse (i, j) in syn.
- $\text{In}, \text{out} \in \{1, 2, ..., m\}$ indicate the input and output neurons respectively.

In SNP systems, spiking rules $(E/a^c \rightarrow a; d)$ can be applied in any neuron as follows: if neuron σ_i contains k spikes a with $a^k \in L(E)$ and $k \geq c$, the spiking rule $E/a^c \rightarrow a; d$ is enabled to be applied. By using the rule, c spikes a are consumed, thus $k - c$ spikes a remain in the neuron σ_i, and after d time units, one spike a is sent to all neurons σ_j such that $(i, j) \in syn$. For any spiking rule, if $E = a^c$, the rule is simply written as $a^c \rightarrow a; d$ and if $d = 0$, we can omit it, and then the spiking rules can be written as $a^c \rightarrow a$.

Rules of the form $a^s \rightarrow \lambda, s \geq 1$ are forgetting rules with the restriction $a^s \notin L(E)$ (that is to say, a neuron cannot apply the spiking rules and forgetting rules at the same moment), where L(E) is a set of regular languages associated with regular expression E . λ is the empty string. If neuron σ_i contains exactly s spikes, the forgetting rule $a^s \rightarrow \lambda$ can be executed, and then s spikes are removed from the neuron.

The TSP is a class of problem that finding a shortest closed tour visiting each city once and only once. Given a set $\{c_1, c_2, \ldots c_n\}$ of n cities and symmetric distance $d(c_i, c_j)$ which gives the distance between city c_i and c_j, the goal is to find a permutation π of these n cities that minimizes the following function:

$$\sum_{i=1}^{n-1} d\left(c_{\pi(i)}, c_{\pi(i+1)}\right) + d\left(c_{\pi(n)}, c_{\pi(1)}\right) \tag{1}$$

3 OSNPS for TSP

3.1 The Structure of OSNPS

The SNPS can be represented graphically. A directed graph is used to represent the structure: the neurons connect with each other by the synapses; the output neuron emits spikes to the environment using outgoing synapse.

Inspired by the fact that spiking neural P system can generate string languages or spike trains [9], an extended spiking neural P system (ESNPS, for short) has been proposed to produce a binary string, and corresponding probability string is used to represent a chromosome. An ESNPS of degree $m \geq 1$ is shown in Fig. 1.

Each ESNPS consists of m neurons. $\sigma_1, \sigma_2, \ldots \sigma_m$ are neurons of the form $\sigma_i = (1, R_i, P_i)$ with $1 \leq i \leq m$, where $R_i = \{r_i^1, r_i^2\}$ ($r_i^1 = \{a \rightarrow a\}$ and $r_i^2 = \{a \rightarrow \lambda\}$) is a set of rules and $P_i = \{p_i^1, p_i^2\}$ is a 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 $p_i^1 + p_i^2 = 1$. If the ith neuron spikes, we get its output 1 and probability p_i^1, otherwise, we get its output 0 and p_i^2. That is to say we get 1 by probability p_i^1 and we get 0 by probability p_i^2.

For example, as for an ESNPS of degree m = 5, its probability matrix is shown in below. If we get the spike train [0 0 1 1 0], then the corresponding probability vector is [0.49 0.65 0.42 0.79 0.45].

$$\left[\begin{array}{c|ccccc} p_i^1 & 0.51 & 0.35 & 0.42 & 0.79 & 0.55 \\ p_i^2 & 0.49 & 0.65 & 0.58 & 0.21 & 0.45 \end{array}\right]$$

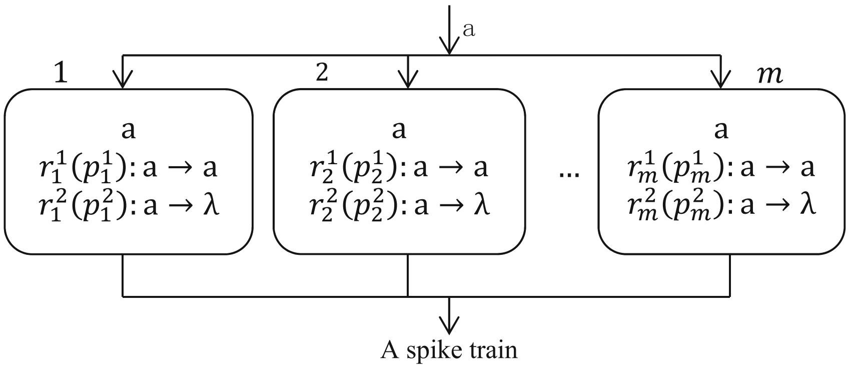

Fig. 1. An example of ESNPS structure

From Fig. 2, we can see that a certain number of ESNPS can be organized into OSNPS by introducing a guider to adjust the selection probabilities and adding a subsystem (σ_{m+1} and σ_{m+2}) to be the spikes supplier. OSNPS consists of H ESNPS, $ESNPS_1$, $ESNPS_2$…, $ESNPS_H$. Each ESNPS is identical (Fig. 1) and the operation steps are illustrated in Subsect. 3.2. Thus, each ESNPS outputs a spike train at each moment of time, and then OSNPS will output H binary string, and we can get the corresponding probability matrix.

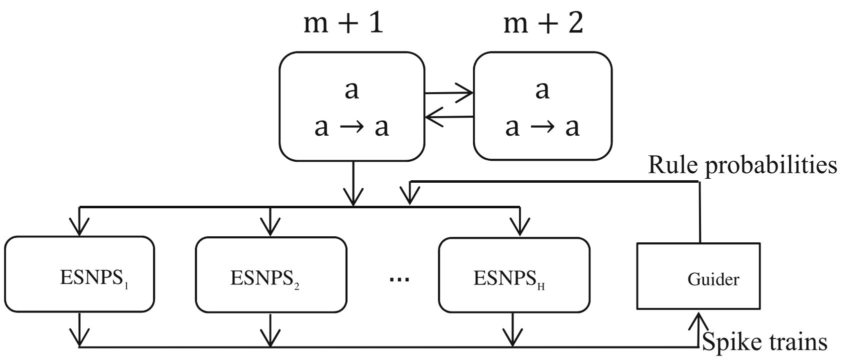

Fig. 2. The structure of OSNPS

In the OSNPS, $\sigma_{m+1} = \sigma_{m+2} = (1, \{a \rightarrow a\}), \sigma_{m+1}$ and σ_{m+2} spike at each time, send spike to each ESNPS and reload each other continuously. We record the spike train matrix T_t (t is current evolution generation) outputted by OSNPS and the corresponding probability matrix P_t. If we can adjust the probabilities, we can control the outputted matrix. In this paper, we put GA algorithm as the guider algorithm to adjust the probability.

We introduce the idea of smallest position value (SPV) [10] method into the genetic algorithm and we give a Table 1 to explain this encoding and decoding method. We put 2 4 1 3 5 as the city sequence.

Table 1. An example of SPV

Dimension j	Position P_{ij}	Sequence
1	0.65	2
2	0.32	4
3	0.87	1
4	0.46	3
5	0.21	5

The input of the guider is a spike train T_t with $H \times m$ bits. The output of the guider is the rule probability matrix $P_t = \left[p_{ij}\right]_{H\times m}$, which is made up of the rule probabilities of H ESNPS. Where p_{ij} is the probability of spiking rule or forgetting rule. For example, as for an ESNPS of degree m = 5, one of the vectors mentioned above [0.49 0.65 0.42 0.79 0.45] could be a part of the P_t.

3.2 The Operation Steps

1. Initialize system parameters;
2. Neuron σ_{m+1} and neuron σ_{m+2} spike and supply neurons for H ESNPS. At the same time, H ESNPS spike and output spike training matrix T_t(0–1matrix);
3. Put T_t into the guider and rearrange it as corresponding probability matrix P_t; We put P_t as the initial population of GA and convert P_t to real matrix M_t by using the SPV idea;
4. Calculate fitness function;
5. Selection operation: Roulette wheel selection algorithm and optimal individual preservation strategy are used; we select the first ten percent of the best individual to save;
6. Cross operation: Adopting OX crossover algorithm;
7. Mutation operation: Using transposition mutation techniques;
8. Judge whether the termination condition (the max generation) is met or not. If it reached, output the final result, end; otherwise $t = t + 1$ and go to step 9;
9. We combine the updated probability matrix P_t with the corresponding 0–1 matrix T_t to update the probability of each ESNPS and go to step 1;

In the implementation process of OSNPS, each of the neuron in ESNPS according to the rules of probability to spike spiking rules or forgetting rules, which will increase the population diversity (Fig. 3).

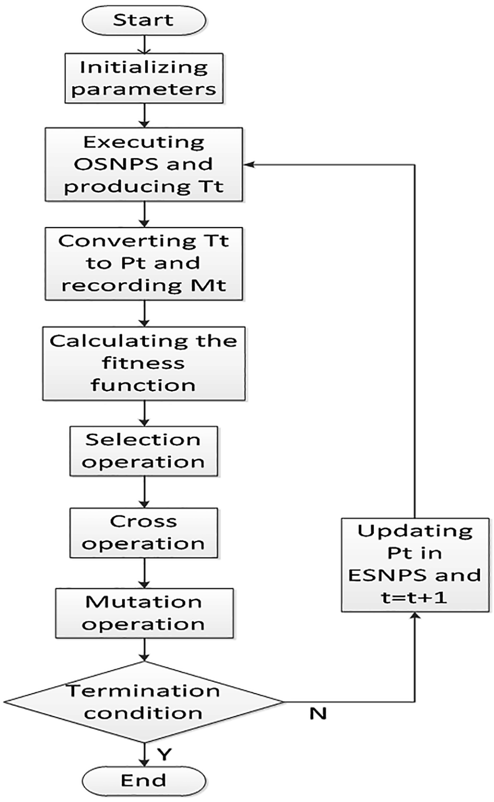
Start
Initializing parameters
Executing OSNPS and producing Tt
Converting Tt to Pt and recording Mt
Calculating the fitness function
Selection operation
Cross operation
Mutation operation
Termination condition
N
Y
Updating Pt in ESNPS and t=t+1
End

Fig. 3. The system flow diagram

4 Experimental Results

In this section, our system was implemented using matlab and tested on a personal PC with Pentium IV 3.0 GHz CPU and 512 MB memory. The population size is taken as 30; Crossover probability $p_c = 0.8$ and the mutation probability $p_m = 0.2$. The maximum iteration number N is taken as 500. Since OSNPS mainly uses the combination of SNPS and GA algorithm, we make a contrast experiment between the improved GA algorithm in the guider and the OSNPS system (Figs. 4 and 5).

Through experiments, we can see that when the number of cities is 30, the results of OSNPS are better than guider algorithm, but OSNPS find the optimum in 402 generation and guider algorithm in 247 generation (Fig. 6).

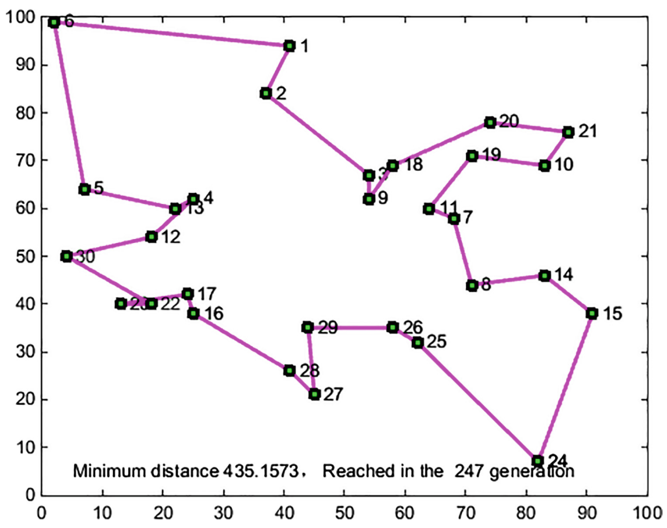

Fig. 4. 30 cities in guider algorithm

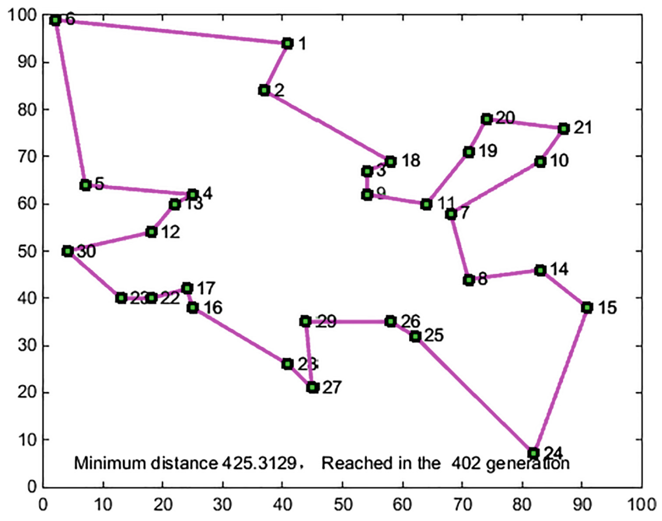

Fig. 5. 30 cities in OSNPS

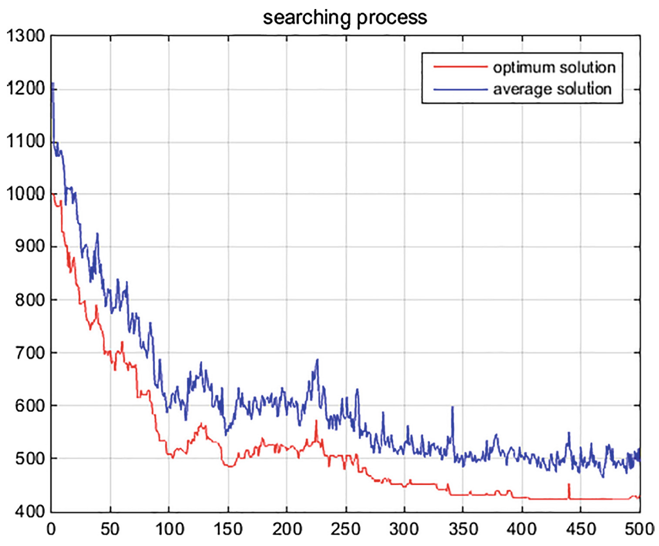

Fig. 6. The searching process of OSNPS

5 Conclusion

In this paper, we proposed the OSNPS for solving the TSP. The OSNPS achieve the connection between the GA algorithm and the membrane system. Experimental results show that the OSNPS can effectively solve TSP and prevent GA algorithm from falling into local optimum. The ideas of this article not only contribute to the membrane algorithm of the neural-like P system, but also find a new way to solve the TSP.

Certainly, the OSNPS has some drawbacks in solving the TSP. When the scale of the problem is getting bigger and bigger, the advantage of OSNPS is increasingly obscure and the system need more time to solve problems than standard GA. So the future work is to improve the SNP system or GA algorithm to optimize experimental results.

Acknowledgment. This work was supported by the Natural Science Foundation of China (No. 61502283). Natural Science Foundation of China (No. 61472231).

References

1. Păun, G.: Computing with membranes. J. Comput. Syst. Sci. **61**(1), 108–143 (2000)
2. Freund, R., Păun, G., Pérez-Jiménez, M.J.: Tissue-like P systems with channel-states. Theor. Comput. Sci. **330**, 101–116 (2005)
3. Ionescu, M., Păun, G., Yokomori, T.: Spiking neural P systems. Fund. Inform. **71**(2), 279–308 (2006)
4. Zhang, G., Gheorghe, M., Pan, L., et al.: Evolutionary membrane computing: a comprehensive survey. Inf. Sci. **279**(1), 528–551 (2014)

5. Nishida, T.Y.: An application of P systems: a new algorithm for NP-complete optimization problems. In: Proceedings of 8th World Multi-Conference Systems, Cybernetics and Informatics, pp. 109–112 (2004)
6. Zhang, G.X., Gheorghe, M., Wu, C.Z.: A quantum-inspired evolutionary algorithm based on P systems for knapsack problem. Fund. Inform. **87**(1), 93–116 (2008)
7. Zhang, G., Cheng, J., Gheorghe, M., Meng, Q.: A hybrid approach based on differential evolution and tissue membrane systems for solving constrained manufacturing parameter optimization problems. Appl. Soft Comput. **13**(3), 1528–1542 (2013)
8. Zhang, G., Rong, H., Neri, F., et al.: An optimization spiking neural P system for approximately solving combinatorial optimization problems. Int. J. Neural Syst. **24**(05), 1440006 (2014)
9. Reeves, C.R.: A genetic algorithm for flowshop sequencing. Comput. Oper. Res. **22**(1), 5–13 (1995)
10. Chen, H., Freund, R., Ionescu, M., et al.: On string languages generated by spiking neural P systems. Fund. Inform. **75**(75), 141–162 (2007)