Journal of Universal Computer Science, vol. 18, no. 13 (2012), 1821-1841
submitted: 15/11/11, accepted: 30/3/12, appeared: 1/7/12 © *J.UCS*

A Novel Membrane Algorithm Based on Particle Swarm Optimization for Solving Broadcasting Problems

Gexiang Zhang, Fen Zhou, Xiaoli Huang, Jixiang Cheng
(School of Electrical Engineering, Southwest Jiaotong University
Chengdu, China
{zhgxdylan, zhoufen20041673, chengjixiang0106}@126.com
huangxiaoli-22@163.com)

Marian Gheorghe
([1]Department of Computer Science, University of Sheffield
Sheffield, UK
[2]Department of Computer Science, University of Pitesti
Pitesti, Romania
m.gheorghe@dcs.shef.ac.uk)

Florentin Ipate, Raluca Lefticaru
(Department of Computer Science, University of Pitesti
Pitesti, Romania
florentin.ipate@ifsoft.ro, raluca.lefticaru@gmail.com)

Abstract: This paper presents the application of membrane algorithms to broadcasting problems, which are regarded as NP-hard combinatorial optimization problems. A membrane algorithm, called HPSOPS, is proposed by appropriately combining membrane systems and a hybrid particle swarm optimization with wavelet mutation (HPSOWM). HPSOPS is designed with the hierarchical membrane structure and transformation/communication-like rules of membrane systems, the representation of individuals and the evolutionary mechanism of HPSOWM. Experimental results from various broadcasting problems show that HPSOPS performs better than its counterpart HPSOWM and genetic algorithms reported in the literature, in terms of search capability, efficiency, solution stability and precision.

Key Words: Membrane computing, Membrane algorithm, Broadcasting problem, Particle swarm optimization, Membrane systems
Category: F.1.1, F.2.1, I.2.8, I.2.11

1 Introduction

As a young branch of natural computing, membrane computing, initiated in 1998, aims to abstract computing models, called membrane systems or P systems, from the structure and the functioning of the living cell as well as from the cooperation of cells in tissues, organs, and other populations of cells [Păun 2000, Păun et al. 2010]. A P system is a distributed parallel computing model consisting of a hierarchical or network structure of membranes, multisets of objects and a set of rules. Each membrane defines a region where objects are placed inside. The rules, related to specific objects and membranes, are responsible for evolving a P system in a nondeterministic and maximally parallel manner. A computation is a sequence of configurations starting from an initial state of the system and ending when no more rules can be applied. Until now various P systems have been proven to be equivalent to Turing machines, i.e., computationally complete or universal [Arroyo et al. 2002, Pan and Ishdorj, 2004] and various simulation tools of P systems have been designed [Nepomuceno-Chamorro 2004].

An important aspect of membrane systems is that they are distributed and parallel computing devices. To complement the overwhelming majority of researches in this area dealing with the computability and universality of membrane systems, an appropriate model, represented by the membrane systems, for distributed computing was shown by Ciobanu in [Ciobanu 2003, Nicolescu 2012, Ciobanu et al. 2006, Dinneen et al. 2010]. This model emphasizes the algorithmic aspects related to the distributed systems computational power provided by membrane systems. Because in membrane systems the process of passing objects through membranes in both directions is similar to message passing, Ciobanu considered a system of communicating membranes with antiport carriers, and the main meaning regarding this choice is that the membrane will send and receive information. Some basic algorithms of distributed computing, including algorithms for broadcast, convergecast, flooding, leader election, mutual exclusion in distributed systems, the fault tolerant systems and the consensus problem, were presented in [Ciobanu 2003]. Broadcast, one of the basic algorithms, represents the core theory of distributed computing over communicating membrane systems. Following Ciobanu, Lefticaru et al. [Lefticaru et al. 2010] presented a method to determine the format of the rules of P systems utilized to specify a broadcast, using a genetic algorithm (GA). In [Lefticaru et al. 2010], the broadcasting algorithm defined within a P system framework was investigated by considering a number of variants of P systems, and the dependencies between the format of the rules in each compartment and the number of its neighbors, as well as a method to automatically generate the rules in each compartment depending on the number of neighbors.

To further solve the broadcasting problem shown in [Lefticaru et al. 2010], this paper proposes a novel membrane algorithm, called hybrid particle swarm optimization with wavelet mutation based on a P system (HPSOPS). To the best of our knowledge, this is the first attempt to use a membrane algorithm to solve the broadcasting problem defined within a membrane system framework. HPSOPS is designed with one-level membrane structure (OLMS) [Zhang et al. 2008], transformation /communication-like rules in cell-like P systems and the evolutionary rules of the hybrid particle swarm optimization with wavelet mutation (HPSOWM) [Ling et al. 2008]. The choice of HPSOWM as a base algorithm to design HPSOPS is based on the following considerations. HPSOWM, introduced in [Ling et al. 2008], is a hybrid particle swarm optimization incorporated with the wavelet mutation. As compared with particle swarm optimization approaches, HPSOWM has a better performance with respect to the quality of solutions and convergence. The better the base algorithm is, the better the performance of the P system designed by the membrane algorithm is. We use a large number of broadcasting problems with various types and sizes to test the HPSOPS performance. Experimental results show that HPSOPS outperforms HPSOWM and genetic algorithms in [Lefticaru et al. 2010], in terms of search capability, efficiency, solution stability and precision.

This paper is organized as follows. Section 2 first gives a brief introduction of cell-like P systems, and then the broadcasting problem through a P system is described in detail. In section 3, HPSOPS is presented in detail. Section 4 discusses the parameter setting of HPSOPS. Section 5 provides a performance comparison between HPSOPS and HPSOWM, GA.

2 Broadcasting problems through a P system

This section starts with a brief introduction of P systems, and then the broadcasting problem is presented in detail.

2.1 Cell-like P systems

A P system is a new class of distributed and parallel computing devices inspired by the structure and functionality of living cells. The membrane structure of a cell-like P system is a hierarchical arrangement of membranes embedded in a main membrane called the skin membrane. Each membrane defines a region containing multisets of objects and related to a set of transformation/ communication-like rules. The multisets of objects evolve and move from a region to a neighboring one by applying the rules in a nondeterministic and maximally parallel way. Each object that may evolve must follow its corresponding rules. A computation starts from an initial configuration to a new one, and stops when no more evolution rules are available. The result of the computation is obtained in the output membrane or emit from the skin membrane [Păun 2000].

The membrane structure of a cell-like P system can be formalized as follows [Păun 2000, Păun 2002]

$$\prod = (O, T, \mu, w_1, ..., w_m, R_1, ..., R_m, i_0)$$

where

- O is an alphabet of objects.

- $T \subseteq O$ (the output alphabet).
- μ is a membrane structure with m membranes and the regions labeled by the elements of a given set H. m is called the degree of Π.
- ω_i, $1 \leq i \leq m$, are strings which represent multisets over O associated with the regions $1, 2, \ldots, m$ of μ.
- R_i is a set of evolution rules.
- i_0 is the output membrane.

A computation of the system is composed of a series of computing steps between configurations. Each computation starts by processing the initial multisets, and follows the non-deterministic and maximally parallel manner. A computation halts when there is no rule applicable in any region [Păun 2000, Zhang et al. 2008].

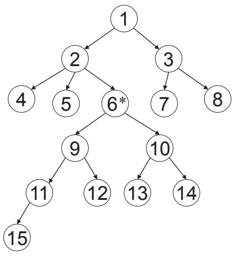


Figure 1: The tree describing a membrane structure (the start node for broadcasting is 6)

2.2 Broadcasting problems

A basic broadcasting problem is to send a message from one node of a network to all the other nodes without revisiting them. In this paper, we study the broadcasting problem defined within a P system framework. In a P system environment, we abstract the P system membrane into a tree structure and then the broadcasting problem for P systems involves sending the message through the tree structure. The broadcasting problem for P systems does not consider the format of the rules that may lead to a variety of types of P systems, even more important, it mainly discusses specific complexity aspects of the communication processes involved [Ciobanu 2003, Lefticaru et al. 2010].

Given the membrane structure of a cell-like P system, we assume that a membrane m_i has to send an object to all the membranes of the system. Clearly, we have two cases of broadcast, i.e., m_i is the skin membrane (the root node), and m_i is any membrane (node). Because the second case is a generalization of the first one, we discuss the algorithm for the second case.

In order to clearly illustrate the broadcasting problem, we consider a generic node j surrounded by neighbors p, i, k; one of them may be a parent and the others are children, or all of them are children. The message, denoted as O, might come from any of them and then travels to the others. In order to conceive various rules allowing the message received from one of its neighbors to travel through j towards its other neighbors, four distinct cases illustrated by different types of P systems were proposed in [Lefticaru et al. 2010]. Experiments show that the following case has more appropriate features than the others. Therefore we also consider this case in this paper.

Case: the compartment j contains the multiset p, i, k and the rules:

$$pc \rightarrow (j'Oc^{n_{j,p}})_p|\neg p'$$
$$ic \rightarrow (j'Oc^{n_{j,i}})_i|\neg i'$$
$$kc \rightarrow (j'Oc^{n_{j,k}})_k|\neg k'$$

In this case j receives from p the multiset $p'Occ$. The symbol p' acts as an inhibitor of the first rule, preventing it from resending O back to p. c is an object associated with a communication between two membranes. Two copies of c allow the second and third rules to be executed. In the above rules $n_{j,h}$ are integer values defining the number of non-visited neighbors of h, excluding j, $h \in \{p, i, k\}$. It is easy to work out the relationship between the format of a rule and the number of non-visited descendants of the neighbor associated with the rule. These rules are applied in one step.

In order to further describe the broadcasting problem in detail, we give an example with the membrane structure of a P system defined as:

$$\mu = [[[]_4[]_5[[[[]_{15}]_{11}[]_{12}]_9[[]_{13}[]_{14}]_{10}]_6]_2[[]_7[]_8]_3]_1$$

then the membrane structure is mapped into the tree structure as shown in Fig. 1. According to the broadcasting principle, the message generated in a membrane is sent to its parent membrane, and to all its directly contained son membranes (if any). We assume that membrane 6 has to send an object to all membranes of the system. The message from node 6 is first sent to nodes 2, 9, and 10. In the following step, from these compartments the messages are sent to nodes 1, 4, 5, nodes 11, 12, and 13, 14, respectively. The process continues like this until all the membranes are visited, and then the entire broadcasting ends.

We present the description of the first two steps of the broadcasting algorithm in detail in this case.

First of all, we consider a general situation: a membrane j is included in p and contains k membranes $i_1, ..., i_k$. We use $M_j = \{p, i_1, ..., i_k\}$ to denote the multisets in membrane j. The initial mutisets for the membrane structure $\mu = [[[]_4[]_5[[[[]_{15}]_{11}[]_{12}]_9[[]_{13}[]_{14}]_{10}]_6]_2[[]_7[]_8]_3]_1$ are

$M_1 = \{2,3\}$, $M_2 = \{1,4,5,6\}$, $M_3 = \{1,7,8\}$, $M_4 = \{2\}$, $M_5 = \{2\}$, $M_6 = \{2,9,10\}$, $M_7 = \{3\}$, $M_8 = \{3\}$, $M_9 = \{6,11,12\}$, $M_{10} = \{6,13,14\}$, $M_{11} = \{9,15\}$, $M_{12} = \{9\}$, $M_{13} = \{10\}$, $M_{14} = \{10\}$, $M_{15} = \{11\}$.

Step 1: The starting membrane is injected with an object O and a number of objects c, each of which is for each neighbor. For example, if the starting membrane is $j = 6$, shown in Fig. 1, then we get the multiset $\{2, 9, 10, O, c, c, c\}$ according to the initial multiset M_6 and the additional symbols mentioned above; the rules are:

$$\begin{aligned} R_6 = \{r_{6,2} &: 2c \to (6'O)_2(c_2)^{n_{6,2}}\}|\neg 2', \\ r_{6,9} &: 9c \to (6'O)_9(c_9)^{n_{6,9}}\}|\neg 9', \\ r_{6,10} &: 10c \to (6'O)_{10}(c_{10})^{n_{6,10}}\}|\neg 10'\} \end{aligned}$$

After these rules are applied in membrane 6, the objects 2, 9, 10, c, c, c are consumed and only an O remains in this membrane, which means that the message has been received.

Step 2: If in step 1, we consider $n_{6,2} = 3$, $n_{6,9} = 2$ and $n_{6,10} = 2$, then in membranes 2, 9, 10 which are neighbors of 6, the multisets will be $\{1, 4, 5, 6, 6', O, c, c, c\}$, $\{6, 11, 12, 6', O, c, c\}$, and $\{6, 13, 14, 6', O, c, c\}$, respectively. The rules in membranes 2,9,10 will be

$$\begin{aligned} R_2 = \{r_{2,1} &: 1c \to (2'O)_1(c_1)^{n_{2,1}}\}|\neg 1', \\ r_{2,4} &: 4c \to (2'O)_4(c_4)^{n_{2,4}}\}|\neg 4', \\ r_{2,5} &: 5c \to (2'O)_5(c_5)^{n_{2,5}}\}|\neg 5' \\ r_{2,6} &: 6c \to (2'O)_6(c_6)^{n_{2,6}}\}|\neg 6'\} \end{aligned}$$

$$\begin{aligned} R_9 = \{r_{9,6} &: 6c \to (9'O)_6(c_6)^{n_{9,6}}\}|\neg 6', \\ r_{9,11} &: 11c \to (9'O)_{11}(c_{11})^{n_{9,11}}\}|\neg 11', \\ r_{9,12} &: 12c \to (9'O)_{12}(c_{12})^{n_{9,12}}\}|\neg 12'\} \end{aligned}$$

$$\begin{aligned} R_{10} = \{r_{10,6} &: 6c \to (10'O)_6(c_6)^{n_{10,6}}\}|\neg 6' \\ r_{10,13} &: 13c \to (10'O)_{13}(c_{13})^{n_{10,13}}\}|\neg 13', \\ r_{10,14} &: 14c \to (10'O)_{14}(c_{14})^{n_{10,14}}\}|\neg 14'\} \end{aligned}$$

The rules $r_{2,1}, r_{2,4}, r_{2,5}, r_{9,11}, r_{9,12}, r_{10,13}$ and $r_{10,14}$ are applied. We obtain the following multisets $\{O\}, \{6, 6', O\}, \{6, 6', O\}, \{6, 6', O\}$, in the regions 6, 2, 9 and 10, respectively. If in step 1, we consider $n_{6,2} = 1$ or $n_{6,2} = 2$, then at least one of the rules $r_{2,1}, r_{2,4}$ or $r_{2,5}$ cannot be applied, because a c is missing. And then in the corresponding hierarchy of compartments the message O is not received. The multiset in region 2 becomes $\{5, 6, 6', O\}$, where 5, a neighbor of compartment 2, is the non-visited compartment.

If in step 1, we consider $n_{6,2} > 3$, and then the multiset is $\{1, 4, 5, 6, 6', O, c^{n_{6,2}}\}$. After the three rules $r_{2,1}, r_{2,4}, r_{2,5}$ are applied, the multiset becomes $\{6, 6', O,\ c^{n_{6,2}-3}\}$. The process starts again from the compartments which have been impacted on by the communication rules in step 2.

From this example we can discover the following rules about the value $n_{j,i}$: if the values $n_{j,i}$ are chosen appropriately, each membrane will get an O and no c finally; if $n_{j,i}$ is less than the expected value, at least one compartment will not receive the message O; if the value of $n_{j,i}$ is greater than the expected one, we will receive a certain number of c in some compartments; when the inhibitors i' are present in compartments, some values of $n_{j,i}$ do not count, i.e., $n_{2,6}, n_{9,6}, n_{10,6}$; The number of relevant $n_{j,i}$ values is the same as the number of pairs of parent-child relationships in the membrane structure and is equal to the number of compartments minus 1.

Following the rules mentioned above, we can obtain the solution for the membrane structure given in Fig. 1 as follows:

$$n_{6,2} = 3, n_{6,9} = 2, n_{6,10} = 2, n_{2,1} = 1, n_{2,4} = 0, n_{2,5} = 0, n_{9,11} = 1,$$
$$n_{9,12} = 0, n_{10,13} = 0, n_{10,14} = 0, n_{1,3} = 2, n_{11,15} = 0, n_{3,7} = 0, n_{3,8} = 0$$

the other values of $n_{j,i}$ do not count.

By using the above values of $n_{j,i}$, the P system will end with the mutisets shown as follows:

$$M_1 = \{2, 2', O\}, M_2 = \{6, 6', O\}, M_3 = \{1, 1', O\}, M_4 = \{2, 2', O\},$$
$$M_5 = \{2, 2', O\}, M_6 = \{O\}, M_7 = \{3, 3', O\}, M_8 = \{3, 3', O\},$$
$$M_9 = \{6, 6', O\}, M_{10} = \{6, 6', O\}, M_{11} = \{9, 9', O\}, M_{12} = \{9, 9', O\},$$
$$M_{13} = \{10, 10', O\}, M_{14} = \{10, 10', O\}, M_{15} = \{11, 11', O\}$$

where M_j $(j = 1, ..., 15)$ is the multiset inside the compartment j.

Considering the non-determinism of the P system, for the same values of some parameters, there will be different messages sent. i.e., in Fig. 1, if $n_{6,2} = 2$, $M_2 = \{1, 4, 5, 6, 6', O, c, c\}$; if $r_{2,1}$ and $r_{2,4}$ are applied, the compartment 5 will not receive any message. Similarly, if $r_{2,4}$ and $r_{2,5}$ are applied, four compartments in the subtree rooted in 1 will not be visited.

The fitness function simulating the broadcasting in the tree, starting from the root and using the parameters $n_{i,j}$, was given in [Lefticaru et al. 2010] and is shown in (1). At the end of the broadcasting, each candidate solution was evaluated by counting the unvisited nodes and the extra messages sent to the nodes.

$$fitness = \lambda \cdot no\_of\_unvisited\_nodes + no\_of\_extra\_messages \quad (1)$$

where $no\_of\_unvisited\_nodes$ denotes the number of nodes which have not received the message and do not contain any object O at the end of the computation; $no\_of\_extra\_messages$ is the number of extra objects c that cannot be consumed and are still present in the nodes at the end of the computation; λ is a positive penalty (or weight) parameter which weights the importance of the $no\_of\_unvisited\ \_nodes$ or the $no\_of\_extra\_messages$. According to the investigation in [Lefticaru et al. 2010], λ is set to 10 in the following experiments.

3 HPSOPS

Both P systems and evolutionary algorithms (EAs) are parallel models inspired by nature. P systems can provide a suitable framework for parallel-distributed computation and EAs have very extensive applications in solving numerous optimization problems, so it is advisable to combine P systems with EAs to produce a new kind of algorithms, called membrane algorithms. Since membrane algorithms were introduced, they have attracted much attention and they are still under investigation in methodological aspects and practical applications in recent years. In [Zhang et al. 2008], a membrane algorithm, QEPS, was presented by combining quantum-inspired evolutionary algorithms with P systems, where one-level membrane structure (OLMS) was introduced, to solve a class of well-known combinatorial optimization problems, knapsack problems. QEPS was also successfully applied to solve satisfiability problems, also a class of well-known combinatorial optimization problems [Zhang et al. 2009]. In [Zhang et al. 2010], a modified QEPS (MQEPS) with a local search was used for time-frequency atom decomposition for analyzing radar emitter signals. In [Cheng et al. 2011] and [Zhang et al. 2011], OLMS was combined with differential evolution and ant colony optimization to solve numerical optimization and traveling salesman problems, respectively. In [Zhou et al. 2010], a preliminary study on the membrane algorithm combining P systems with PSO was made. This paper goes further to discuss the interaction of P systems and PSO. In [Zhang et al. 2010], the first attempt to comparatively analyze QEPS and QIEA by using population diversity and convergence measures shows that QEPS has a stronger capacity of balancing exploration and exploitation than QIEA. This analysis provided a very illustrative way for a better understanding of the role played by P systems in the context of membrane algorithms.

PSO, developed by Kennedy and Eberhart [Kennedy and Eberhart 1995], is an optimization algorithm from the learning process of swarm intelligence and human cognition. Comparing with other evolutionary algorithms, PSO has comparable or even better search performance for many NP-hard optimization problems due to its faster and more stable convergence rate. However, observations reveal that PSO sharply converges in the early stage of the searching process, thus it will quickly saturate or even terminate the further searching process. In order to overcome this drawback of PSO, HPSOWM was proposed in [Ling et al. 2008] in which the PSO was hybridized with wavelet-theory-based mutation. Experimental results conducted on a suite of benchmark test functions and three industrial applications empirically show that HPSOWM significantly outperforms several EAs in terms of convergence speed, solution quality, and solution stability. To make full use of the advantages of HPSOWM, this paper uses it to design a membrane algorithm, HPSOPS, with the hierarchical framework of cell-like P systems.

Based on the concepts and principles of HPSOWM and P systems, HPSOPS employs the evolutionary rules of HPSOWM, transformation/communication-like rules in P systems and an OLMS in which the skin membrane contains m elementary membranes defining m regions. The objects employed will be organized as multisets. The evolutionary rules of HPSOWM as well as transformation/ communication-like rules of P systems will be responsible for evolving the system and selecting the best individuals [Zhang et al. 2008].

The pseudocode algorithm for HPSOPS is presented in Fig. 2 and the detailed description is as follows:

```
       Begin
          t ← 1
(i)       Initialize membrane structure and X(t), V(t)
(ii)      Allocate individuals for each elementary membrane
          While (not termination condition) do
             For i = 2 : m+1
(iii)           Perform HPSOWM in the ith elementary membrane
             End
(iv)         Execute communication rules
             t ← t+1
          End
       End
```

Figure 2: The pseudocode algorithm for HPSOPS

(i). In this step, the membrane structure $[_1[_2]_2[_3]_3[_4]_4 \cdots [_{m+1}]_{m+1}]_1$ composed of a skin membrane denoted by 1 and m regions inside the skin membrane is constructed. A particle swarm X with n particles in a D-dimensional search space, $X = \{x_1, x_2, x_3, \ldots\ldots x_n\}$, is initialized, where x_i is an arbitrary individual in X and x_i is represented as $x_i = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{iD})$. The velocity is described as $v_i = (v_{i1}, v_{i2}, v_{i3}, \ldots v_{iD})$.

(ii). n individuals are randomly allocated into the m elementary membranes in a non-deterministic way, i.e., at least one individual selected from n individuals in a random way is put into each elementary membrane and each of the rest $n-m$ individuals is randomly chosen into any arbitrary one of the m elementary membranes. Multisets are initialized as follows:

$$w_1 = \lambda,$$
$$w_2 = b_1 b_2 b_3 \ldots b_{n_1},$$
$$w_3 = b_{n_1+1} b_{n_1+2} \ldots b_{n_2},$$
$$\ldots\ldots$$
$$w_{m+1} = b_{n_{(m-1)}+1} b_{n_{(m-1)}+2} \ldots b_{n_m}, \quad n_m = n$$

where n is the population size and $b_i, 1 \le i \le n$ is an individual.

(iii). Evolution rules in each of the compartments 2 to $m+1$ are implemented. HPSOWM is independently performed in all elementary membranes. The pseudocode algorithm for HPSOWM is shown in Fig. 3, and the detailed description is as follows.

(a) The evolutionary generation g_i, $1 \leq i \leq m$ for HPSOWM in the $t \leftarrow 1$ elementary membrane is set to a uniformly random integer.

(b) Find *pbest*: compare the evaluated fitness value of each particle with *pbest*, which is the best solution of each particle in the history. If the current value is better than *pbest*, the current fitness value and the current particle location are assigned to *pbest* and the *pbest*'s location, respectively.

(c) Find *gbest*: if the current fitness value is better than *gbest*, which is the best solution of the whole population, *gbest* is assigned to the current fitness.

(d) Update the velocities of the particles according to (2).

$$v_{id}(t+1) = w \cdot v_{id}(t) + c_1 \cdot r_1 \cdot (p_{id}(t) - x_{id}(t)) + c_2 \cdot r_2 \cdot (p_{gd}(t) - x_{id}(t)) \quad (2)$$

where r_1 and r_2 are uniform random numbers in the range of (0, 1); c_1 and c_2 are acceleration coefficients, usually $c_1 = c_2 = 2.05$; w is a weight coefficient varying in [0.1, 0.9]; the value of v_{id} can be clamped to the range $[-V_{\max}, V_{\max}]$ to ensure that particles are scattered in the search space (if the prescribed ranges are exceeded by v_{id} , the ones will be set again in the range).

(e) Update the locations of the particles according to (3).

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad 1 \leq i \leq n_i \quad 1 \leq d \leq D \qquad (3)$$

where D denotes the dimension of the particle. Generally speaking, the range of the dth dimensional position is set in $[{para_j}^{\min}, {para_j}^{\max}]$ (if the prescribed ranges exceed by x_{id}, the ones will be set again in the range).

(f) Wavelet mutation operation is implemented. The details of the operation are as follows. Each particle element of the swarm will have a chance to mutate that is governed by a probability of mutation $p_m \in [0, 1]$. For each particle element, a random number between 0 and 1 will be generated such that if it is less than or equal to p_m, a mutation will take place on that element. For instance, if $\mathrm{x}_p(t) = [x_{p1}(t), x_{p2}(t), \ldots, x_{pD}(t)]$ is the selected pth particle, and the element of particle $x_{pj}(t)$ is randomly selected for the mutation (the value of $x_{pj}(t)$ is inside the particle element's boundaries $[{para_j}^{\min}, {para_j}^{\max}]$), the resulted particle is given by $\overline{\mathrm{x}}_p(t) = [\overline{x}_{p1}(t), \overline{x}_{p2}(t), \ldots \overline{x}_{pD}(t)]$, where

$$\overline{x}_{pj}(t) = \begin{cases} x_{pj}(t) + \delta \times ({para_j}^{\max} - x_{pj}(t)) & \text{if } \delta > 0 \\ x_{pj}(t) + \delta \times (x_{pj}(t) - {para_j}^{\min}) & \text{if } \delta \leq 0 \end{cases} \qquad (4)$$

in which $j = 1, 2, \ldots, D$, and

$$\delta = \frac{1}{\sqrt{a}} \psi(\frac{\phi}{a}) \qquad (5)$$

where $\psi(x)$ is called a "mother wavelet"; a is the dilation parameter. In this paper, the Morlet wavelet defined in (6) is chosen as the mother wavelet in the wavelet mutation operation.

$$\psi(x) = e^{-x^2/2} \cos(5x) \tag{6}$$

By using the Morlet wavelet in (6) as the mother wavelet, function (5) can be described as

$$\delta = \frac{1}{\sqrt{a}} e^{-(\frac{\phi}{a})^2/2} \cos(5(\frac{\phi}{a})) \tag{7}$$

where ϕ can be randomly generated from $[-2.5a, 2.5a]$. The value of the dilation parameter a is set to a variable varying with the value of t/T to meet the fine-tuning purpose, where T is the total number of iterations and t is the current number of iterations. A monotonic increasing function governing a and t/T is presented as follows

$$a = e^{-\ln(g)\times(1-\frac{t}{T})^{\xi_{wm}}+\ln(g)} \tag{8}$$

where ξ_{wm} is the shape parameter of the monotonic increasing function and g is the upper limit of parameter a. The dilation parameter a is governed by the monotonic increasing function (8) which is controlled by the parameters ξ_{wm} and g.

According to the discussion in [Ling et al. 2008], g =10000 is a good choice to the algorithm performance and no formal methods are available to choose the value of the parameter ξ_{wm} which depends on the characteristics of an optimization problem. Thus, considering the conclusions in [Ling et al. 2008] and the characteristics of broadcasting problem, ξ_{wm} is set to 0.5 in the following experiments.

(iv). Communication rules are implemented between elementary membranes and the skin membrane. The detailed description is as follows. The communication rules are used to exchange some information among the m regions or between each region and the skin membrane. HPSOPS employs *pbest* and *gbest* to generate the offspring, therefore HPSOPS applies the communication rules to send the best fit individual in each elementary membrane into the skin membrane and send *pbest* and *gbest* of the best individual from the skin membrane back to each region. After the process, the population in each elementary membrane is updated and will guide the individuals toward better solutions at the next evolutionary step.

(v). Repeat Steps (ii) and (iii) until the termination condition is satisfied. The result in the skin membrane is regarded as the final solution of the problem.

HPSOPS uses a P system-like framework, in which the rules are applied to the objects accordance with the behavior described above. In HPSOPS, the initial particle swarm is assigned to the membrane structure randomly. The initial population is composed of the multisets $w_2, w_3, ..., w_{m+1}$. Each membrane applies rules in step (ii) to select the best individuals at the current generation. The computations in all membranes are parallel and independent. The computation will not halt until the best fit solution remains unchanged for a certain number of generations.

Begin
  Set the iterations for each elementary membranes;
  $t \leftarrow 1$
  While (not termination condition) **do**
    For i=1:n_i
      Store the best position $pbest$ obtained by P_i so far
      Store the best position $gbest$ obtained by all P_i so far
      Update particle's velocity $V(t)$
      Update particle's position $X(t)$
      Perform mutation operation with p_m % wavelet mutation
      Update $\overline{x}_j^p(t)$ based on (4)-(8)
    End
    $t \leftarrow t+1$
  End
End

Figure 3: The pseudocode algorithm for HPSOWM

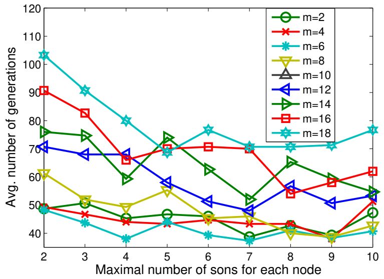
4 Parameter Setting

This section discusses how to set the number m of elementary membranes in HPSOPS through determining the unknown values $n_{i,j}$.

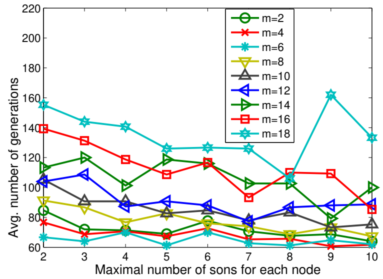
In order to investigate the effect of the parameter m on the HPSOPS performance in solving the broadcasting problem, the number m of elementary membranes varies from 2 to 20 by the interval 2 in the experiments. The population size is set to 20. The subpopulation size in the ith membrane n_i , $1 \leq i \leq m$, is set to a uniformly random integer ranged from 1 to 20-m+1 on condition that the sum of $n_1, n_2, ..., n_m$ equals 20. The evolutionary generation of each elementary membrane g_i, $1 \leq i \leq m$, is set to a uniformly random integer ranged from 1 to 40. HPSOPS uses two stopping criteria: the maximal number 10000 of generations is reached or the optimal solution is found, i.e., $fitness = 0$ (all the tree nodes are visited and no extra message is sent) [Lefticaru et al. 2010].

In the experiments, we consider the trees with different number of nodes, i.e., $N \in \{10, 15, 20, \ldots, 50\}$, and with a random number of sons. This means that each of non-leaf nodes can have a different number of sons, which can be randomly chosen, with an equal probability, from the set $\{1, \ldots, s\}$, where s is the maximal number of sons that can be chosen randomly. In this case, for each number of nodes $N \in \{10, 15, 20, \ldots, 50\}$ (corresponding to the number of compartments in a P system), we consider all the values $s \in \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$. A tree is generated according to the structural criterion.

When the total number of nodes is 10, 15,. . ., 50, respectively, we record the average number of evolutionary generations for obtaining the optimal solution in each number of membranes. Fig. 4 shows the average number of evolutionary generations varies with the maximal number s of sons, where s means that the number of sons for each node varies from 0 to s. These experimental results show that HPSOPS has better performance, i.e., with smaller average number of evolutionary generations for reaching the optimal solutions, when the number of elementary membranes is 6. Thus, the parameter m could be assigned as 6 in the following experiments.



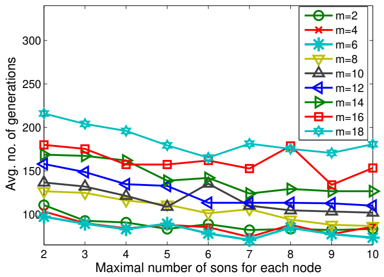
(a) N=10



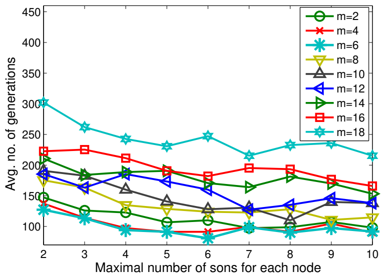
(b) N=15

5 Experimental Results

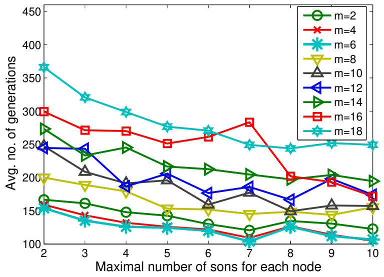
In this section, two cases, $N \leq 50$ and $N > 50$, are considered. The former case is used to compare HPSOPS with HPSOWM and GA. The latter case is applied



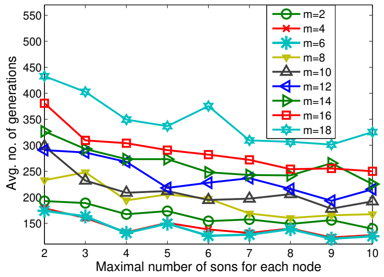
(c) N=20



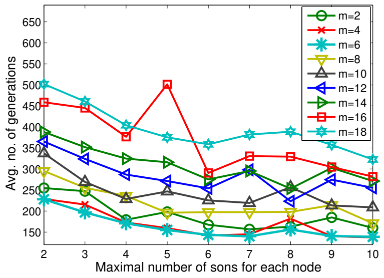
(d) N=25



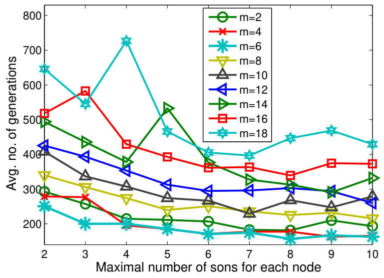
(e) N=30



(f) N=35



(g) N=40

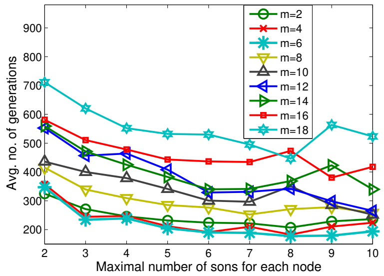


(h) N=45

to further show the performance difference between HPSOWM and HPSOPS. All the experiments are implemented on the platform MATLAB 7.4 by using a PC with 1.7GHz CPU, 512MB RAM and Windows XP OS.

5.1 Comparisons of HPSOPS, HPSOWM and GA for $N \leq 50$

In this subsection, experiments carried out on broadcasting problems are applied to draw a comparison between HPSOPS and its counterpart HPSOWM [Ling et al. 2008], GA [Lefticaru et al. 2010]. The parameter setting is the same as in Section 4. In the experiments, the number of nodes in the tree has 9 choices, i.e., 10, 15, 20, 25, 30, 35, 40, 45 and 50. The maximal number of sons for each node can be chosen as any one of 9 choices: 2, 3,. . ., 10. Thus, there are 81 cases. Each of these cases is executed for 30 independent runs. Experimental results of HPSOPS, HPSOWM and GA are shown in Tables 1-2 and Figs. 5-6. It is worth pointing out that the statistical results of GA are referred to [Lefticaru et al. 2010]. The average generations used to find the optimal solutions are shown in Table 1, where the first row and the first column represent the maximal number of sons and the number of nodes, respectively. Table 2 shows the average rates of successfully finding the optimal solutions over 30 runs. In



(i) N=50

Figure 4: Average generations of HPSOPS for trees with N nodes and variable number of sons, m is the number of elementary membranes

Table 2, the first row and the first column represent the maximal number of sons and the number of nodes, respectively. To clearly illustrate the experimental results, Fig. 5 shows the average generations used by HPSOPS, HPSOWM and GA, respectively. Fig. 6 shows the average successful rates of HPSOPS, HPSOWM and GA, respectively.

As it can be seen from Tables 1, 2 and Fig. 5, HPSOPS uses lower number of evolutionary generations than GA and HPSOWM to successfully solve the broadcasting problems; HPSOWM is superior to GA with respect to the average generations. Tables 1, 2 and Fig. 6 show that the successful rates obtained by HPSOWM and HPSOPS are more stable than those obtained by GA. The successful rates of GA decrease quickly when the number of nodes is greater than 45. The higher the number of nodes is, the lower are the successful rates. But both HPSOPS and HPSOWM achieve the successful rates 100% for all cases. To further test HPSOPS and HPSOWM performances, we will conduct more experiments in the following subsection.

5.2 Comparisons of HPSOWM and HPSOPS for $N > 50$

This subsection provides more experimental results to show the performance difference between HPSOPS and HPSOWM. In the experiments, the number of nodes in the tree increases from 10 to 150 by the interval 5 and the rest parameter setting is the same as that in Section 4. All the tests are executed for 30 independent runs. For each test, we record the mean and standard deviation of the best fitness (MF, SF) over 30 runs, the mean number of generations (MG, SG) over 30 runs for successfully finding the optimal solutions, and the cumulated duration of time for 30 runs (Dur.). successful rates for trees with a variable number of sons s

		2	3	4	5	6	7	8	9	10
10	GA	63.27	58.27	58.80	72.73	66.50	64.47	57.10	57.13	66.30
	HPSOWM	59.3	53.56	50.87	49.04	54.55	61.67	58.21	60.34	58.91
	HPSOPS	48.24	48.67	45.33	38.46	40.67	40.67	42.31	44.67	42.67
15	GA	155.4	148.4	158.9	163.6	159.4	152.6	147.7	152.0	153.9
	HPSOWM	89.4	97.2	84.0	80.4	83.6	94.2	90.4	98.7	91.5
	HPSOPS	78.0	61.3	62.0	58.6	64.6	55.3	68.2	60.1	63.3
20	GA	381.2	379.9	356.5	358.5	353.9	328.9	382.5	407.4	376.8
	HPSOWM	197.4	175.5	187.0	167.6	201.6	204.4	194.0	187.7	199.4
	HPSOPS	102.5	84.4	89.3	78.6	88.3	74.6	72.0	77.3	76.6
25	GA	882.9	772.1	823.4	863.5	833.2	842.6	834.8	822.1	834.2
	HPSOWM	267.3	294.3	237.8	258.7	298.1	306.6	268.3	293.9	274.3
	HPSOPS	120.6	112.3	106.6	94.6	92.8	105.3	96.0	94.7	99.3
30	GA	1549.3	1519.5	1785.0	1475.8	1657.5	1691.4	1423.4	1579.1	1451.1
	HPSOWM	346.4	446.3	316.6	406.7	353.6	383.3	354.7	385.4	367.2
	HPSOPS	161.3	134.4	146.6	135.3	124.6	106.6	114.0	94.7	112.6
35	GA	2864.5	2700.2	2822.2	3073.1	2491.6	2426.2	2681.9	2952.8	2610.6
	HPSOWM	401.5	581.2	513.6	551.5	518.8	553.3	538.5	494.9	532.5
	HPSOPS	180.3	166.5	149.3	148.6	132.2	120.6	145.3	126.5	118.3
40	GA	4112.8	4785.1	4557.3	4120.0	4534.5	4819.1	4281.4	4317.2	4343.1
	HPSOWM	734.4	834.7	704.0	700.5	759.3	878.0	789.0	758.3	774.5
	HPSOPS	236.4	205.3	157.3	160.9	159.3	138.6	146.6	161.3	168.1
45	GA	6733.4	7226.0	6764.3	7079.8	6686.4	6328.5	6766.3	6997.1	6519.5
	HPSOWM	947.4	956.6	1044.8	1017.4	973.1	992.5	967.0	1006.4	984.6
	HPSOPS	288.6	212.6	202.4	189.3	199.3	168.0	170.6	160.6	187.3
50	GA	9223.7	9554.1	9311.4	8869.9	8775.8	8571.3	8782.7	8849.4	8994.0
	HPSOWM	1234.1	1334.5	1534.0	1734.0	1397.3	1963.6	1502.3	1834.2	1334.5
	HPSOPS	325.3	236.4	233.3	246.4	207.3	190.6	179.3	179.3	190.5

Table 1: Average generations of GA, HPSOWM and HPSOPS

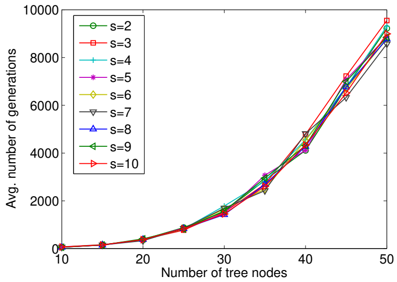
are show in Fig. 7. For page limits, we put the detailed numerical results behind Fig. 7 on the website *http://staffwww.dcs.shef.ac.uk/people/M.Gheorghe/*.

Several conclusions can be drawn from Fig. 7. When HPSOWM is used to solve the broadcasting problems, the successful rates begin to decrease and vary for different problems after the number of nodes is greater than 80. As the number of nodes becomes larger, HPSOWM obtains lower successful rates. When the number of nodes is 100, the successful rates of HPSOWM decrease below 80%. So it is not necessary to perform more experiments. The successful rates of HPSOPS are always at the level of 100% for various broadcasting problems when

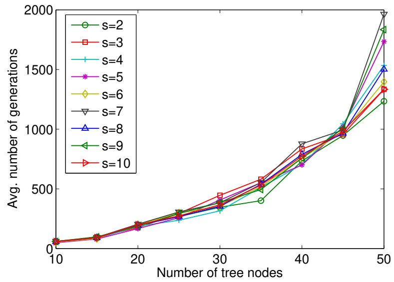
		2	3	4	5	6	7	8	9	10
	GA	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
10	HPSOWM	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	HPSOPS	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	GA	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
15	HPSOWM	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	HPSOPS	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	GA	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
20	HPSOWM	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	HPSOPS	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	GA	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
25	HPSOWM	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	HPSOPS	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	GA	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
30	HPSOWM	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	HPSOPS	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	GA	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
35	HPSOWM	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	HPSOPS	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	GA	100.0	100.0	100.0	100.0	100.0	100.0	100.0	96.7	100.0
40	HPSOWM	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	HPSOPS	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	GA	83.3	90.0	83.3	80.0	93.3	93.3	86.7	83.3	93.3
45	HPSOWM	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	HPSOPS	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	GA	40.0	40.0	33.3	50.0	63.3	53.3	56.7	40.0	50.0
50	HPSOWM	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
	HPSOPS	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

Table 2: Successful rates of GA, HPSOWM and HPSOPS

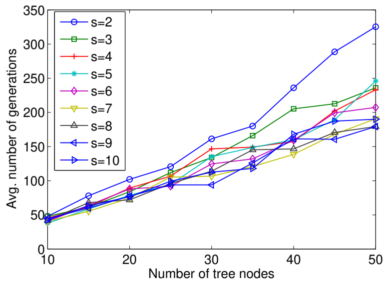
the number of nodes is lower than 110. The successful rates of HPSOPS decrease gradually as the number of nodes increase from 110 to 150. When the number of nodes is 145, HPSOPS can still achieve at least 75% successful rate. Therefore, the results indicate that HPSOPS can obtain slower decrease of successful rates than HPSOWM.



(a) GA



(b) HPSOWM



(c) HPSOPS

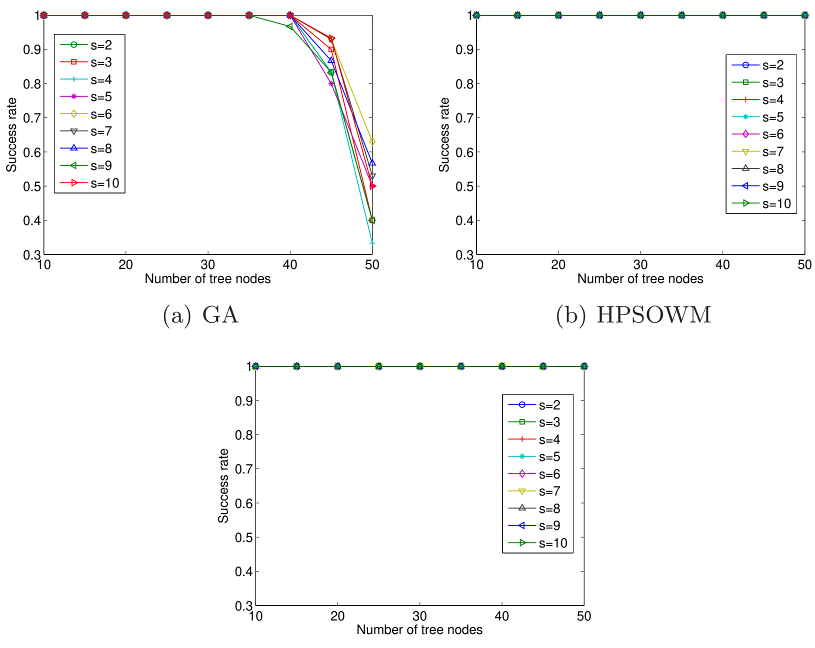
Figure 5: Average generations of different algorithms for trees with $N \leq 50$ nodes and variable number of sons between $1, \ldots, s$

6 Conclusions

This paper discusses the application of a novel membrane algorithm, HPSOPS, to broadcasting problems modeled by tuning the rules of a P system. By introducing the hierarchical framework, OLMS, and transformation/communication-like rules of cell-like P systems into HPSOWM, we design a more effective and efficient approximate algorithm. Extensive experiments conducted on the broadcasting problem with various nodes and sons show that HPSOPS outperforms its counterparts HPSOWM and GA [Lefticaru et al. 2010], in terms of search capability, efficiency, solution stability and precision.

Acknowledgement

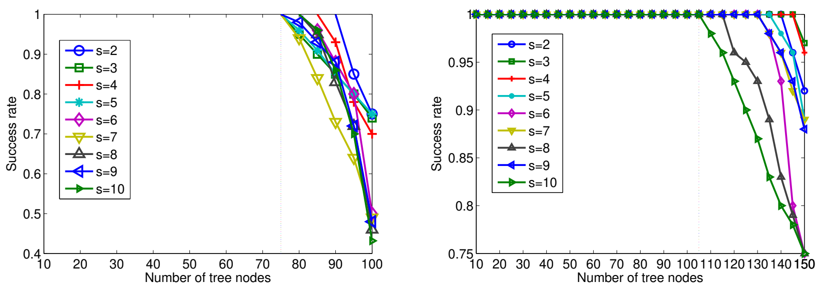
We are grateful to the Guest Editors, Prof. Atulya Nagar and Prof. Robinson Thamburaj, and the anonymous reviewers for their comments that allowed us to improve this paper. The work of GZ was supported by the National



(a) GA (b) HPSOWM

(c) HPSOPS

Figure 6: Successful rates of different algorithms for trees with $N \leq 50$ nodes and variable number of sons between $1, \ldots, s$



(a) HPSOWM (b) HPSOPS

Figure 7: Successful rates of different algorithms for trees with $N > 50$ nodes and variable number of sons between $1, \ldots, s$

Natural Science Foundation of China (61170016), the Program for New Century Excellent Talents in University (NCET-11-0715) and SWJTU supported project (SWJTU12CX008), the project-sponsored by SRF for ROCS, SEM, the Scientific and Technological Funds for Young Scientists of Sichuan (09ZQ026-040), the Fund for Candidates of Provincial Academic and Technical Leaders of Sichuan and the Fundamental Research Funds for the Central Universities (SWJTU11ZT07). The work of MG, FI and RL was partially supported by project MuVeT, Romanian National Authority for Scientific Research (CNCS-UEFISCDI) grant number PN-II-ID-PCE-2011-3-0688.

References

[Arroyo et al. 2002] Arroyo, F., Baranda, A., Castellanos, J., Păun, Gh.,: "Membrane computing: The power of (rule) creation"; Journal of Universal Computer Science, 8, 3(2002): 369–381.

[Cheng et al. 2011] Cheng, J.X., Zhang, G.X., Zeng, X.X.: "A novel membrane algorithm based on differential evolution for numerical optimization"; International Journal of Unconventional Computing, 7, 3(2011): 159–183.

[Ciobanu 2003] Ciobanu, G.: "Distributed algorithms over communicating membrane systems"; Biosystems, 70, 2(2003), 123–133.

[Ciobanu et al. 2006] Ciobanu, G., Păun, Gh., Pérez-Jiménez, M.J., eds.: "Applications of Membrane Computing". Springer-Verlag, Berlin, 2006.

[Dinneen et al. 2010] Dinneen, M.J., Kim, Y.B., Nicolescu, R.: "P systems and Byzantine agreement"; Journal of Logic and Algebraic Programming, 79, 6(2010), 334–349

[Kennedy and Eberhart 1995] Kennedy, J., Eberhart, R.C.: "Particle swarm optimization"; In Proc. of ICNN, (1995), 1942–194 .

[Lefticaru et al. 2010] Lefticaru, R., Ipate, F., Gheorghe, M., Zhang, G.X.: "Tuning P systems for solving the broadcasting problem"; In Proc. of the 10th Workshop on Membrane Computing, (2010), 337–354.

[Ling et al. 2008] Ling, S.H., Iu, H.H.C., Chan, K.Y., Lam, H.K., Yeung, B.C.W., Leung, F.H.:"Hybrid particle swarm optimization with wavelet mutation and its industrial applications"; IEEE Transactions on System, Man, and Cybernetics-Part B: Cybernetics, 38, 3(2008), 743–763.

[Nepomuceno-Chamorro 2004] Nepomuceno-Chamorro, I.A.:"A Java simulator for membrane computing"; Journal of Universal Computer Science, 10, 5(2004): 620–629.

[Nicolescu 2012] Nicolescu, R.: "Parallel and Distributed Algorithms in P Systems"; in M. Gheorghe et al. (Eds.) Lecture Notes in Computer Science (CMC2011), 7184, (2012), 35–50.

[Pan and Ishdorj, 2004] Pan, L.Q., Ishdorj, T.O.: "P systems with active membranes and separation rules"; Journal of Universal Computer Science, 10, 5(2004): 630–649.

[Păun 2000] Păun, Gh.: "Computing with membranes"; Journal of Computer and System Sciences, 61, 1(2000), 108–143.

[Păun 2002] Păun, Gh., Rozenberg, G.: "A guide to membrane computing"; Theoretical Computer Science, 287, 1(2002), 738–1100.

[Păun et al. 2010] Păun, Gh., Rozenberg, G., Salomaa, A.: "Handbook of membrane computing"; Oxford: Oxford University Press (2010).

[Zhang et al. 2008] Zhang, G.X., Gheorghe, M., Wu, C.Z.: "A quantum-inspired evolutionary algorithm based on P systems for knapsack problem"; Fundamenta Informaticae, 87, 1(2008), 93–116.

[Zhang et al. 2009] Zhang, G.X., Liu, C.X., Gheorghe, M., Ipate, F.: "Solving satisfiability problems with membrane algorithms"; in Proc. of BIC-TA, (2009), 29-36.

[Zhang et al. 2010] Zhang, G.X., Liu, C.X., Rong, H.N.: "Analyzing radar emitter signals with membrane algorithms"; Mathematical and Computer Modelling, 52, 11-12(2010), 1997–2010.

[Zhang et al. 2010] Zhang, G.X., Liu, C.X., Gheorghe, M.: "Diversity and convergence analysis of membrane algorithms"; In Proc. of BIC-TA, (2010), 596–603.

[Zhang et al. 2011] Zhang, G.X., Cheng, J.X., Gheorghe, M.: "A membrane-inspired approximate algorithm for traveling salesman problems"; Romanian Journal of Information Science and Technology, 14, 1(2011), 3–19.

[Zhang et al. 2012] Zhang, G.X., Gheorghe, M., Li, Y.Q.: "A membrane algorithm with quantum-inspired subalgorithms and its application to image processing"; Natural Computing, 2012, DOI: 10.1007/s11047-012-9320-2.

[Zhou et al. 2010] Zhou, F., Zhang, G.X., Rong, H.N., Cheng, J.X., Gheorghe, M., Ipate, F., Lefticaru, R.: "A Particle Swarm Optimization Based on P systems"; in Proc. of ICNC 2010, 6(2010), 3003–3007.