An Approximate Algorithm Combining P Systems and Ant Colony Optimization for Traveling Salesman Problems

Ge-xiang Zhang[1], Ji-xiang Cheng[2], Marian Gheorghe[3]

[1] School of Electrical Engineering
Southwest Jiaotong University
Chengdu, Sichuan, 610031, P.R.China
zhgxdylan@126.com

[2] School of Information Science & Technology
Southwest Jiaotong University
chengjixiang0106@126.com

[3] Department of Computer Science
University of Sheffield
Sheffield S1 4DP, UK
m.gheorghe@dcs.shef.ac.uk

Summary. This paper proposes an approximate optimization algorithm combining P systems with ant colony optimization, called ACOPS, to solve traveling salesman problems, which are well-known and extensively studied NP-complete combinatorial optimization problems. ACOPS uses the pheromone model and pheromone update rules defined by ant colony optimization algorithms, and the hierarchical membrane structure and transformation/communication rules of P systems. First, the parameter setting of the ACOPS is discussed. Second, extensive experiments and statistical analysis are investigated. It is shown that the ACOPS is superior to Nishida's algorithms and its counterpart ant colony optimization algorithms, in terms of the quality of solutions and the number of function evaluations.

1 Introduction

As a young and vigorous branch of natural computing, membrane computing focuses on abstracting computing models and membrane 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 [1, 2]. The molecular interactions of neurons inspired the development of the neural P systems [3]. In recent years, the interaction between membrane computing and other nature-inspired computing paradigms has been considered from various perspectives. The integration of meta-heuristic search methodologies and P systems has given birth to membrane algorithms [4], which prove to be efficient and effective ways to solve various real world problems. Generally, there are two main methods utilized in providing solutions to optimization problems, approximate and complete approaches. The complete algorithms are guaranteed to find an optimal solution in bounded time for every finite size instance of an optimization problem and they may require exponential computing time in the worst case for an NP-hard problem [5]. These approaches are ineffective in practical circumstances and other ways of tackling these problems are considered. The approximate algorithms instead, focus on producing good solutions in significantly less time rather than obtaining optimal solutions which are hard to compute. These approaches are very important in solving various continuous and combinatorial optimization problems [5].

The membrane algorithm, or the approximate optimization algorithm based on P systems, is a fertile research direction which explores the great potential of membrane computing as a distributed processing mechanism. Until now, relatively limited work has been produced in this field. In [6, 7, 8], an approximate algorithm using a nested membrane structure (NMS) and a local search technique to solve traveling salesman problems was presented. The algorithm was also applied to obtain good approximate solutions to the min storage problem [9]. In [10, 11], an optimization algorithm combining the NMS and conventional genetic algorithms was presented to solve single-objective and multi-objective numerical optimization problems. In [12], the similarities between distributed evolutionary algorithms and P systems were analyzed and new variants of distributed evolutionary algorithms are suggested and applied for some continuous optimization problems. In our previous work [4, 13], a quantum-inspired evolutionary algorithm based on P systems (QEPS) was proposed to solve knapsack and satisfiability problems. In the QEPS, a one-level membrane structure (OLMS) was introduced and a comparison between the OLMS and the NMS was experimentally investigated. Further variants of the QEPS were applied to analyze radar emitter signals and design digital filters [14, 15, 16]. In [17], the application of membrane algorithms to controller design was discussed. All these investigations support the claim made by Păun and Pérez-Jiménez [18] that the membrane algorithm is a rather new research direction with a well-defined scope, a set of open questions, and therefore further studies are necessary to prove the usefulness of P systems-based approaches for real-world applications.

The already established way of conceiving membrane algorithms is to explore the interactions between P systems and various meta-heuristic techniques for solving different problems; their performance is assessed by comparing the results produced by them and their complexity aspects with those obtained by using already available optimization techniques. In this paper, an approximate optimization algorithm integrating P systems and ant colony optimization techniques, called ACOPS, is proposed in order to solve traveling salesman problems (TSPs). This is the first attempt to investigate the interaction between P systems and swarm intelligence approaches in defining membrane algorithms. We use the pheromone model and pheromone update rules of ant colony optimization (ACO) algorithms, and the hierarchical membrane structure and transformation/communication rules of P systems, to specify the ACOPS algorithm. Also we discuss the parameter setting of the ACOPS. A TSP is a well-known and extensively studied NP-complete combinatorial optimization problem. Experiments conducted on fairly large TSP instances and statistical analysis undertaken show that the ACOPS outperforms Nishida's algorithms and its counterpart ACO algorithms. This work is an example of a successful use of membrane computing, in combination with efficient searching methods, for designing approximate optimization algorithms.

This paper is organized as follows. Section 2 first gives a brief introduction of TSP, P systems, and ACO, and then discusses the ACOPS in detail. Section 3 addresses some discussions with regard to parameter setting and experimental results. Conclusions are drawn in Section 4.

2 ACOPS

This section starts with a brief description of the TSP problem, some basic P systems concepts, and a presentation of the ant optimization algorithm. The rest of this section is dedicated to a presentation of the ACOPS.

2.1 Traveling Salesman Problems

TSP is one of the well-known and most intensively studied combinatorial optimization problems in the areas of optimization, operational research, theoretical computer science, and computational mathematics [19, 20]. A TSP can be described as follows. Given a set C of N cities, i.e., $C = \{c_1, c_2, \cdots, c_N\}$, and a set D of the pairwise travel costs d_{ij}, $i, j = 1, 2, \cdots, N, i \neq j$, i.e., $D = \{d_{ij}\}$, it is requested to find the minimal cost of the path taken by a salesman visiting each of the cities just once and returning to the starting point. More generally, the task is to find a Hamiltonian tour with a minimal length in a connected, directed graph with a positive weight associated to each edge. If $d_{ij} = d_{ji}$, the TSP is symmetric in the sense that traveling from city X to city Y costs just as much as traveling in the opposite direction, otherwise, it is asymmetric.

In the TSP, the cost could be associated with distance, time, money, energy, etc.. A number of industrial problems such as network structure design, machine scheduling, cellular manufacturing, and frequency assignment, can be formulated as TSPs; consequently TSP is often used as a standard benchmark for optimization algorithms [21]. In the theory of computational complexity, TSP belongs to the class of NP-complete problems. In this paper, we will consider the symmetric TSPs, in which the distance is used as the cost.

2.2 P Systems

Various P systems in the literature can be classified in three groups: cell-like P systems, tissue- like P systems and neural-like P systems [22]. A cell-like P system, considered in this paper, is characterized by three components: the membrane structure delimiting compartments, the multisets of abstract objects placed in compartments, and the evolution rules applied to objects. The membrane structure of a cell-like P system, shown in Fig. 1, is a hierarchical arrangement of membranes [1]. The outermost membrane is the skin membrane separating the system from its environment. As usual, there are several membranes inside the skin membrane. Each of these membranes defines a region, which forms a different compartment of the membrane structure and contains a multiset of objects, other membranes and a set of evolution rules. The membrane without any membrane inside is called an elementary membrane.

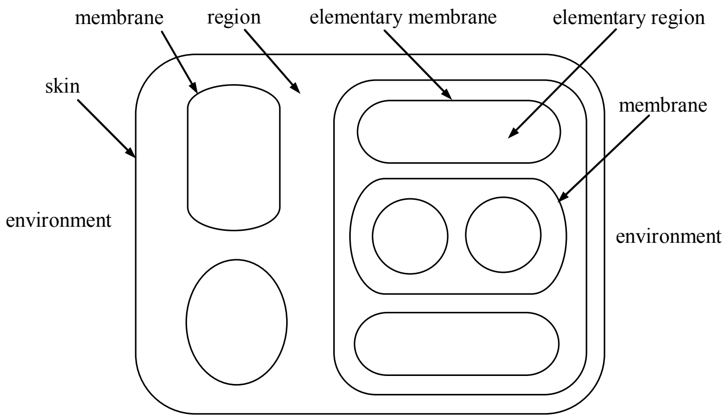


Fig. 1. The membrane structure of a cell-like P system [1]

A cell-like P system with an output set of objects and using transformation and communication rules is formally defined as follows [1, 23]

$$\Pi = (V, T, \mu, w_1, \cdots, w_m, R_1, \cdots, R_m, i_0),$$

where

(i) V is an alphabet; its elements are called objects;
(ii) $T \subseteq V$ the output alphabet);
(iii) μ is a membrane structure consisting of m membranes, with the membranes and the regions labelled in a one-to-one manner with elements of a given set Λ – usually the set $\{1, 2, \cdots, m\}$; m is called the degree of Π;
(iv) w_i, $1 \leq i \leq m$, are strings representing multisets over V associated with the regions, $1, 2, \cdots, m$, of μ;
(v) R_i, $1 \leq i \leq m$, are sets of rules associated with the regions, $1, 2, \cdots, m$, of μ;
(vi) i_0 is a number between 1 and m which specifies the output membrane of Pi.

The rules of R_i, $1 \leq i \leq m$, have the form $a \to v$, where $a \in V$ and $v \in (V \times \{here, out, in\})^*$. The multiset v consists of pairs (b, t), $\in V$ and

$t \in \{here, out, in\}$, where *here* means that b will stay in the region where the rule is used; *out* shows that b exits the region and *in* means that b will be communicated to one of the membranes contained in the current region which is chosen in a non-deterministic way.

A P system, regarded as a model of computation, provides a suitable framework for distributed parallel computation that develops in steps. In the process of computation, multisets of simple objects or are rewritten; the rules associated to regions are employed in a non-deterministic and maximally parallel manner; the rules involving both transformation and communication are responsible for evolving the current objects and transfer them among regions according to some targets; the output region will contain the result of the computation [13].

2.3 Ant Colony Optimization

Ant colony optimizations, ACO for short, a successful evolutionary paradigm of swarm intelligence inspired from the social behaviors of insects and of other animals [19], was initiated by Dorigo in the early 1990s to solve various combinatorial optimization problems [5]. ACO is inspired by the behavior of real foraging ants, which employ pheromone trails to mark their paths to food resources. The main ACO algorithms in the literature include the earliest ant system, MAX-MIN ant system, rank-based ant system, hyper-cube ant system, and ant colony system (ACS). According to the studies in [5, 19, 20, 24], the ACS is one of the most powerful ACO algorithms. Therefore, we consider the use of ACS to construct the ACOPS. In what follows the TSP is taken as an example to describe the ACS.

In the ACS, the TSP is mapped onto a graph called a construction graph in such a manner that feasible solutions of the problem correspond to paths on the construction graph. Artificial ants successively produce better feasible solutions by modifying pheromones deposited on the edges of the TSP graph. The pseudocode algorithm for ACS is shown in Fig. 2, where each step is described as follows.

(i) In this step, the pheromone values are initialized to a value τ_0 ($\tau_0 = 1/ND$) at step $t = 0$, where N is the number of cities in a TSP and D is an arbitrary solution.

(ii) In the *While* loop, M represents the number of ants. Initially the M ants are randomly placed in the N nodes of the TSP graph as the initial state of a tour construction. Each ant uses a pseudorandom proportional rule to choose the next city it will visit. For instance, the kth ant in the ith city chooses the next city j by using the following formula

$$j = \begin{cases} \arg\max\limits_{l \in \mathcal{N}_i^k}\{[\tau_{il}]^\alpha [\eta_{il}]^\beta\}, & \text{if } q \leq q_0 \\ J, & \text{otherwise} \end{cases} \tag{1}$$

where τ_{il} is the pheromone value of the edge connecting the ith node and the lth node; η_{il} is a heuristic information value; the parameters α and $\beta(\alpha > 0$

```
      Begin
          $t \leftarrow 1$
(i)       Initialize pheromone values
          While (not termination condition) do
              For $k$=1, 2, … , $M$
                  For $n$=1, 2, … , $N$
(ii)                  Construct a tour
(iii)                 Local pheromone update
                  End For
              End For
(iv)          Global pheromone update
              $t \leftarrow t+1$
          End While
      End Begin
```

Fig. 2. Pseudocode algorithm for ACS

and $\beta > 0$) determine the relative importance of the pheromone value τ_{il} and the heuristic information η_{il}; $\mathcal{N}_i^k$ ($\mathcal{N}_i^k \subseteq \mathcal{N}$) is the set of all nodes that the kth ant in the ith city can visit, where $\mathcal{N}$ is the set of all the nodes in the TSP graph; $q_0 (0 \le q_0 \le 1)$ is a user-defined parameter specifying the distribution ratio of the two choices; q is a random number generated by using a uniform distribution function in the interval [0, 1]; J means that the next city j is chosen by using a random proportional rule, i.e., the kth ant in the ith city visits the city j at the next step according to the probability

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum\limits_{l \in \mathcal{N}_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, & j \in \mathcal{N}_i^k \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

(iii) After an ant constructs a tour, it will update the pheromone value τ_{ij} of the tour by applying a rule as follows

$$\tau_{ij} = (1 - \upsilon)\tau_{ij} + \upsilon\tau_0 \tag{3}$$

where $\upsilon (0 < \upsilon < 1)$ is a local pheromone decay coefficient. The local pheromone update is used to encourage subsequent ants to choose other edges and, hence, to produce different solutions, by decreasing the pheromone value on the traversed edges. Thus, this step is helpful to the exploration of more solutions.

(iv) This step is to update the pheromone values of all the edges in the TSP graph by employing the best solution searched. To be specific, the pheromone value $\tau_{ij}(t)$ of the edge connecting the ith node and the jth node at generation t is modified as the pheromone value $\tau_{ij}(t + 1)$ at generation $t + 1$, i.e.,

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t) \tag{4}$$

where $\rho(0 < \rho \leq 1)$ is a global pheromone decay coefficient and is also called the evaporation rate of the pheromone, and $\Delta\tau_{ij}(t)$ is

$$\Delta\tau_{ij}(t) = \begin{cases} 1/D_b, & \text{if } (i,j) \in T_b \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where D_b is the minimal distance searched, that is, the best solution searched, and T_b is the shortest path corresponding to D_b. The global pheromone update is to guide ants toward the best path searched.

2.4 ACOPS

In this subsection, we use the hierarchical framework of cell-like P systems and its evolution rules in a slightly modified way, and the parameterized probabilistic model, i.e., the pheromone model, of ACO, to specify the ACOPS algorithm. More specifically, the ACOPS applies the OLMS [13] to organize objects and evolution rules. The objects consist of ants or TSP construction graphs. The evolution rules, which are responsible to evolve the system and select the best ant, include a tour construction, and transformation/communication rules implemented by using local and global pheromone update rules.

More precisely the P system-like framework will consist of:

(i) a membrane structure $\mu = [_0[_1]_1, [_2]_2, \cdots, [_m]_m]_0$ with $m+1$ regions delimited by m elementary membranes and the skin membrane denoted by 0;
(ii) a vocabulary that consists of all the ants;
(iii) a set of terminal symbols, T, TSP construction graphs;
(iv) initial multisets $w_0 = \lambda$,
$w_1 = A_1A_2 \cdots A_{M_1}$,
$w_2 = A_{M_1+1}A_{M_1+2} \cdots A_{M_2}$,
$\cdots\cdots$
$w_m = A_{M_{(m-1)}+1}A_{M_{(m-1)}+2} \cdots A_{M_m}$
where $A_i, 1 \leq i \leq M$, is an ant; M_j, $1 \leq j \leq m$, is the number of ants in the w_j; $\sum_{j=1}^{m} M_j = M$, where M is the total number of ants in this computation;
(v) rules which are categorized as
  a) tour construction rules in each of the compartments 0 to m; these are transformation-like rules which construct tours for the ants (see (ii) in the ACS presentation);
  b) communication rules which use pheromone values to update the edges of the TSP graphs. There are three levels of communication rules. The first level corresponding to the local pheromone update strategy of the ACS is utilized to exchange information between the current ant and its subsequent ant. The second and third levels of communication rules come from the global pheromone update strategy in the ACS. The second one implements information exchange between the best ant and the rest

within a certain membrane. The third one performs the communication between the ants in the elementary membranes and those in the skin membrane.

In the ACOPS the initial colony of ants is scattered across the membrane structure. The initial colony will consist of the multisets $w_1, \cdots, w_m$. Each of the ants in the elementary membranes uses the rules of type (a) to sequentially construct its tours. Going through N(the number of cities) steps, an ant will sketch a whole path for the N cities. If all the ants have their paths, the current generation is assessed compartment by compartment to select the best fit ant for each elementary membrane. The best ant is used to adjust the pheromone values in the TSP graph to communicate with the other ants in the same elementary membrane. Every $g_i (i = 1, 2, \cdots, m)$ generations for the ith compartment, the best ant is sent out to the skin membrane. Thus m ants from m elementary membranes form the initial objects in the skin membrane. These ants evolve independently g_0 generations to elect a best one to communicate with the ants in each elementary membrane. The process will stop according to a preset termination condition, such as a certain number of iterations. The pseudocode algorithm for the ACOPS is summarized in Fig. 3, where each step is described as follows.

```
        Begin
          $t \leftarrow 1$
(i)       Initialize the membrane structure
          While (not termination condition) do
(ii)          Scatter ants into elementary membranes
(iii)         Determine iterations for each of elementary membranes
              For $i$ =1, 2, .., $m$
(iv)              Perform ACS inside the $i$th elementary membrane
              End
(v)           Form a colony of ants in the skin membrane
(vi)          Perform ACS in the skin membrane
(vii)         Execute global communication
              $t \leftarrow t+1$
          End
        End
```

Fig. 3. Pseudocode algorithm for the ACOPS

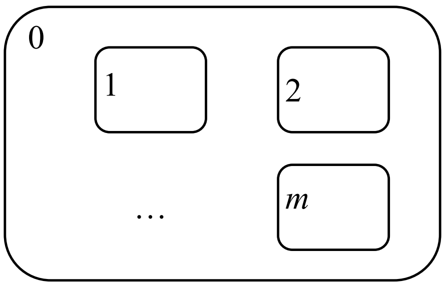


Fig. 4. The one level membrane structure

(i) In this step, the OLMS, shown in Fig. 4, is constructed. How to choose the parameter m will be discussed in Section 3.

(ii) The M ants forming a colony are scattered across the m elementary membranes in such a random way that guarantees each elementary membrane contains at least two ants. This is helpful to perform the second level of the communication process. Thus, the number of ants in an elementary membrane varies from 2 to $M - 2m + 2$.

(iii) This step determines the number of iterations for each elementary membrane to independently perform ACS. To be specific, the number $g_i (i = 1, 2, \cdots, m)$ of iterations for the ith elementary membrane is generated randomly between $g_{\min}$ and $g_{\max}$, i.e.,

$$g_i = g_{\min} + \lfloor \text{rand}(0, 1) \cdot (g_{\max} - g_{\min}) \rfloor \tag{6}$$

where $g_{\min}$ and $g_{\max}$ are lower and upper limits of iterations for elementary membranes, respectively; $\lfloor \cdot \rfloor$ is a function rounding an element to the nearest smaller integer.

(iv) In each of the m elementary membranes, the ACS algorithm shown in Fig. 2 is performed independently, i.e., the tour construction, local pheromone update and global pheromone update are sequentially carried out for $g_i (i = 1, 2, \cdots, m)$ iterations.

(v) The colony of ants in the skin membrane is formed by using the best ants of elementary membranes. Each compartment sends the best ant out into the skin membrane and therefore there are m ants in total.

(vi) The ACS algorithm shown in Fig. 2 is performed independently in the skin membrane for g_0 iterations as in step (iv). The parameter g_0 is determined using (6).

(vii) The global communication is used to exchange some information between the ants in the skin membrane and those in the elementary membranes. To be specific, the best one ant in the skin membrane is employed to update the pheromone values of the TSP graph in each of elementary membranes. This operation has a positive effect on the ants in the compartments toward better fitness, the shorter path.

3 Experiments and Results

The performance of the ACOPS is tested on various TSP instances. First of all, it is discussed how to set the number of elementary membranes by using 4 TSP benchmarks. Then 20 benchmarks are applied to compare ACOPS and its counterpart ACO algorithm. Subsequently, experimental comparisons between ACOPS and Nishida's algorithms are performed on 8 benchmark TSP problems. In these experiments, several parametric and non-parametric tests are employed to analyze the ACOPS behavior.

3.1 Parameter Setting

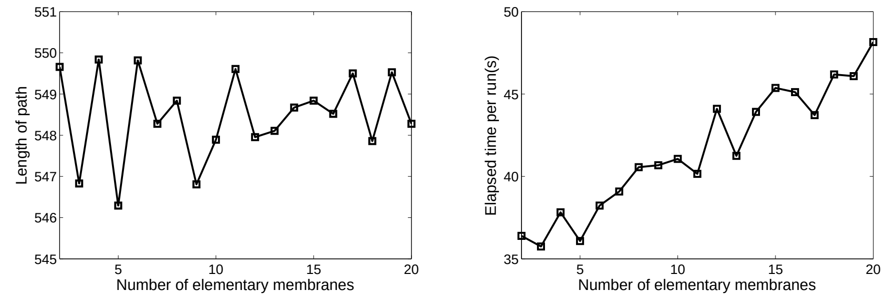
This subsection uses four TSP benchmarks, Eil76, Eil101, Ch130 and Ch150, to discuss how to choose the number m of elementary membranes and the number $g_i (i = 1, 2, \cdots, m)$of generations for each elementary membrane. The four TSPs have N =76, 101, 130 and 150 cities, respectively. According to the studies in the literature, the parameters in the experiments are chosen as follows: $M = 40$, $\alpha = 1$, $\beta = 3$, $\rho = 0.6$, $\upsilon = 0.1$ and $q_0 = 0.9$. We use the number 10000 of function evaluations as the termination criterion for all tests.

We first investigate the effect of the number of elementary membranes on the ACOPS performance. On the basis of the ACOPS description, the parameter m varies from 2 to 20. The parameters $g_{\min}$ and $g_{\max}$ are set to 10 and 30, respectively. The performances of the ACOPS for each of the 19 cases are evaluated by using the best solutions and their corresponding elapsed time per run, and the mean of best solutions and their corresponding mean of elapsed time per run, of 20 independent runs. Experimental results are shown in Fig. 5 – Fig. 12.

It can be seen from Fig. 5 – Fig. 12 that there are some general trends. Both the best solutions and the mean of best solutions over 20 runs have fluctuant behavior. To be specific, there is a first rapid fall and then several waves of higher values follow. The elapsed time per run has a general increase as m goes up from 2 to 20. It is worth pointing out that the general trends become clearer as the complexity of the problem increases. From these experimental results, a trade-off value for the parameter m between the quality of solutions and the elapsed time could be about 4.

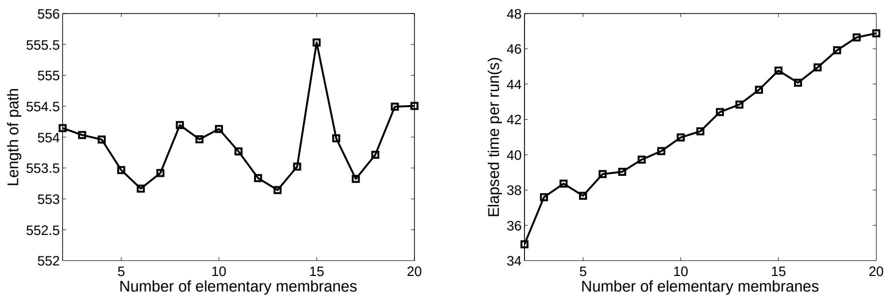
In what follows we set the number of elementary membranes to 4 to conduct a further investigation on the effects of the number of communications (NoC) between the skin membrane and the elementary membranes, i.e., the number of global communications, on the ACOPS performance. Let the NoC vary from 1 to 40. The number of function evaluations (NoFE) as the stopping criterion is 10000. The parameter $g_{\min}$ is set 10. Thus, according to the ACOPS description, the $g_{\max}$ can be obtained from the following formula

$$g_{\max} = \frac{2 \cdot \mathrm{NoFE}}{\mathrm{NoC} \cdot (N + m)} - g_{\min} \tag{7}$$



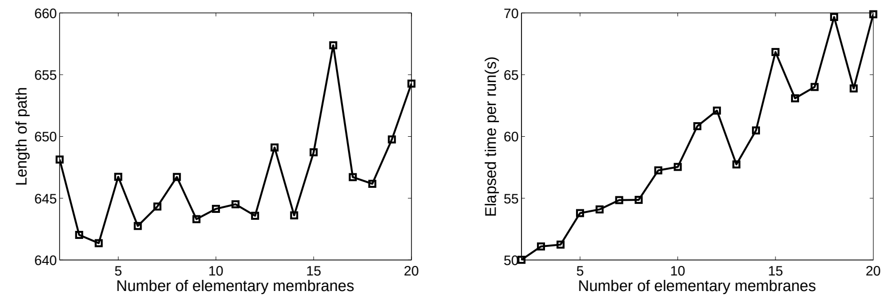
(a) Best solutions (b) Elapsed time per run

Fig. 5. Experimental results of Eil76 with different membranes



(a) Mean of best solutions (b) Mean of elapsed time per run

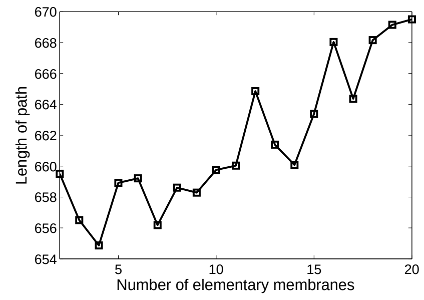
Fig. 6. Experimental results of Eil76 with different membranes



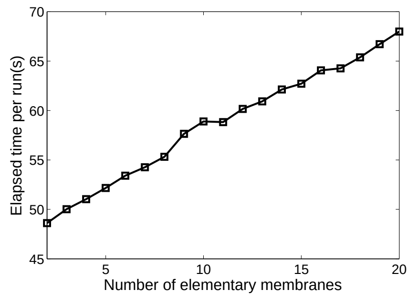
(a) Best solutions (b) Elapsed time per run

Fig. 7. Experimental results of Eil101 with different membranes

where N and m are the total number of ants and the number of elementary membranes (also is the number of ants in the skin membrane), respectively. We also use the four TSP benchmarks, Eil76, Eil101, Ch130 and Ch150, to carry out the

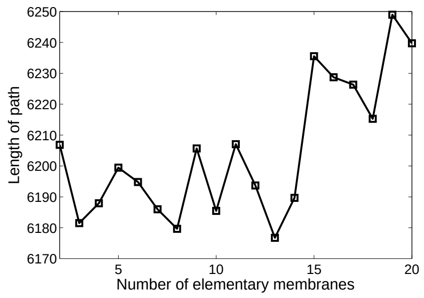


(a) Mean of best solutions

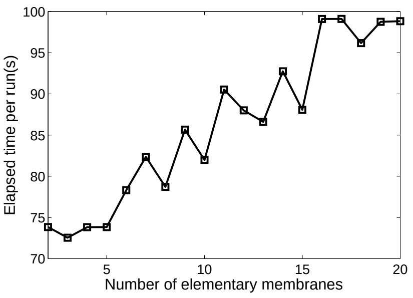


(b) Mean of elapsed time per run

Fig. 8. Experimental results of Eil101 with different membranes

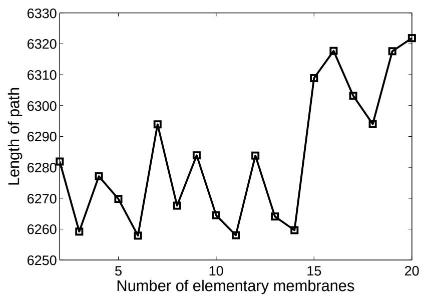


(a) Best solutions

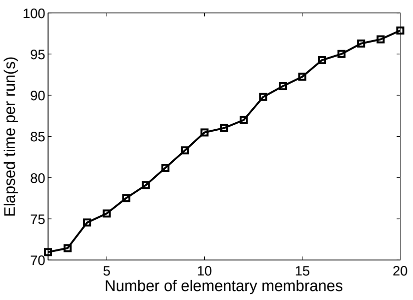


(b) Elapsed time per run

Fig. 9. Experimental results of Ch130 with different membranes



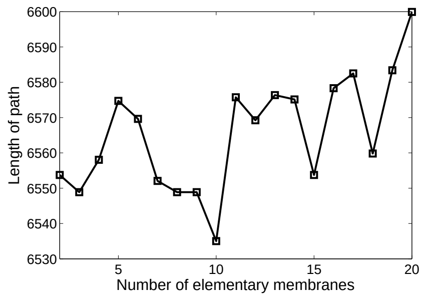
(a) Mean of best solutions



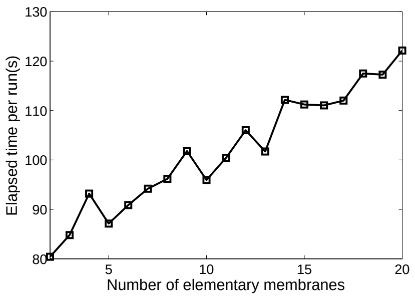
(b) Mean of elapsed time per run

Fig. 10. Experimental results of Ch130 with different membranes

experiments. For each of the 40 cases, we record the best solutions and their corresponding mean of elapsed time per run, and the mean of best solutions and their corresponding mean of elapsed time per run, to assess the ACOPS performance.

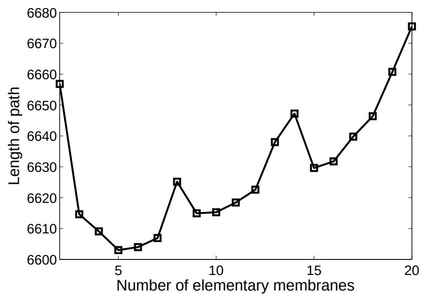


(a) Best solutions

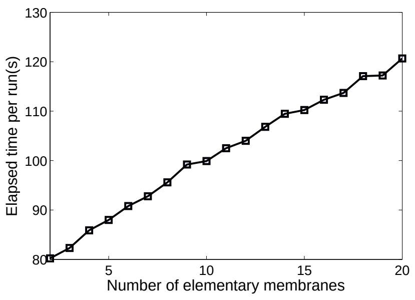


(b) Elapsed time per run

Fig. 11. Experimental results of Ch150 with different membranes



(a) Mean of best solutions



(b) Mean of elapsed time per run

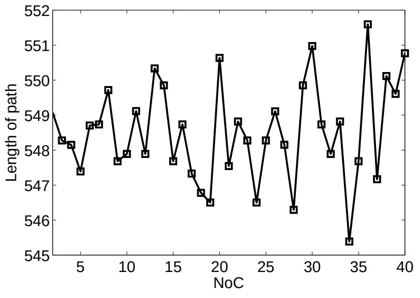
Fig. 12. Experimental results of Ch150 with different membranes

The independent runs for each case are 20. Experimental results are shown in Fig. 13 – Fig. 20.

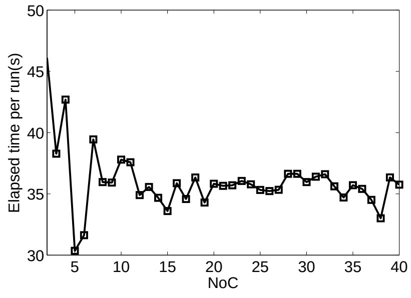
Figures 13 – 20 show that the best solutions and the mean of best solutions over 20 runs have a behavior oscillating between various maxima and minima. The elapsed time per run goes through a drastic fluctuation and then stays a relatively steady level. We note that the trends become clearer as the complexity of the problem increases. Considering a trade-off between quality of solutions and the elapsed time, the recommended value for the NoC could be chosen in the range [15, 35]. Thus, given a certain value of NoFE, an appropriate value for the parameter gmax could be determined in an interval, according to (7).

3.2 Comparisons with ACO and Statistical Analysis

To draw a comparison between the ACOPS and its counterpart ACO, we use 20 symmetric TSP benchmark problems to conduct experiments. The test problems, shown in Table 1, were chosen either because there were data available online in the literature, or the optimal solutions are known. They are challenging enough

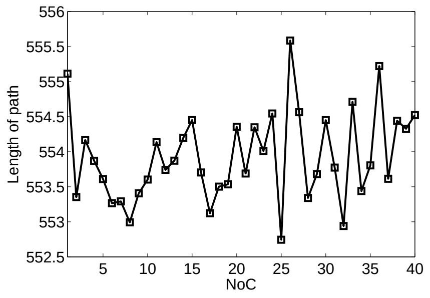


(a) Best solutions

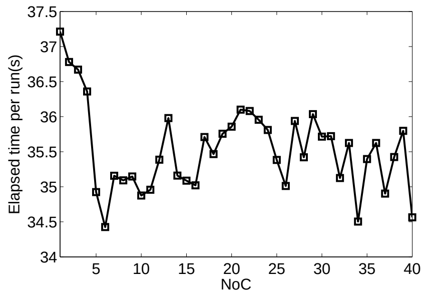


(b) Elapsed time per run

Fig. 13. Experimental results of Eil76 with NoC

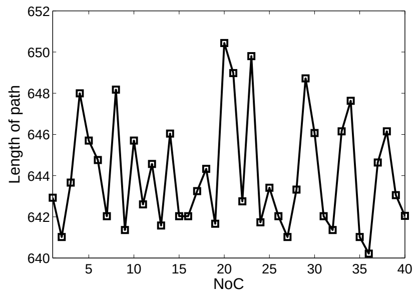


(a) Mean of best solutions

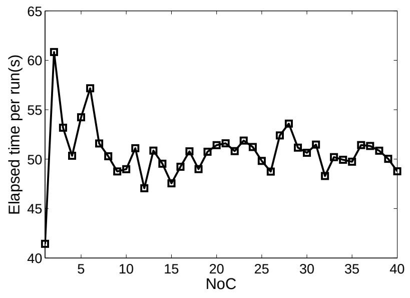


(b) Mean of elapsed time per run

Fig. 14. Experimental results of Eil76 with NoC



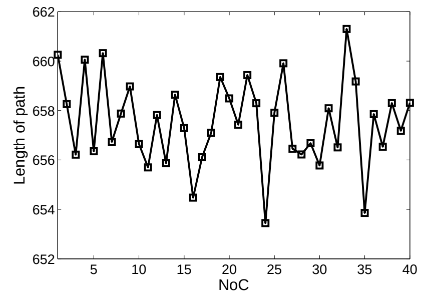
(a) Best solutions



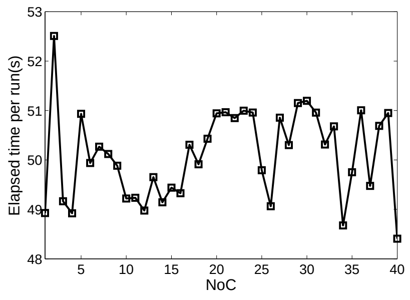
(b) Elapsed time per run

Fig. 15. Experimental results of Eil101 with NoC

for making fair comparisons between the two algorithms, in terms of solving difficult instances of TSPs. In the following experiments, the ACOPS and ACO have identical settings for parameters: $M = 40$, $\alpha = 1$, $\beta = 3$, $\rho = 0.6$, $\upsilon = 0.1$, $q_0 = 0.9$

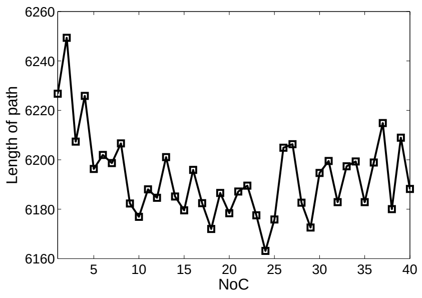


(a) Mean of best solutions

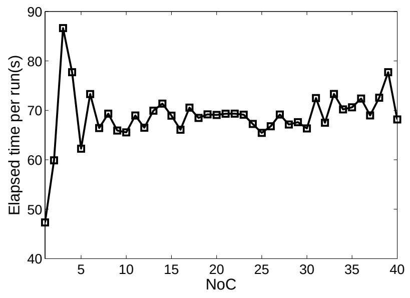


(b) Mean of elapsed time per run

Fig. 16. Experimental results of Eil101 with NoC

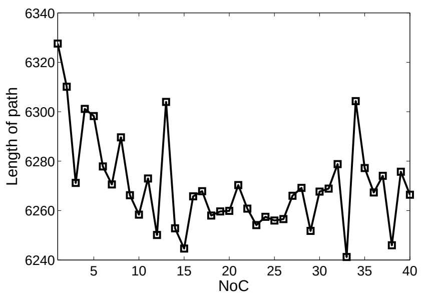


(a) Best solutions

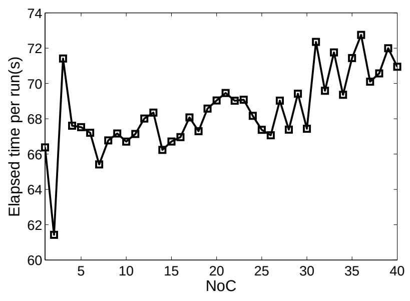


(b) Elapsed time per run

Fig. 17. Experimental results of Ch130 with NoC



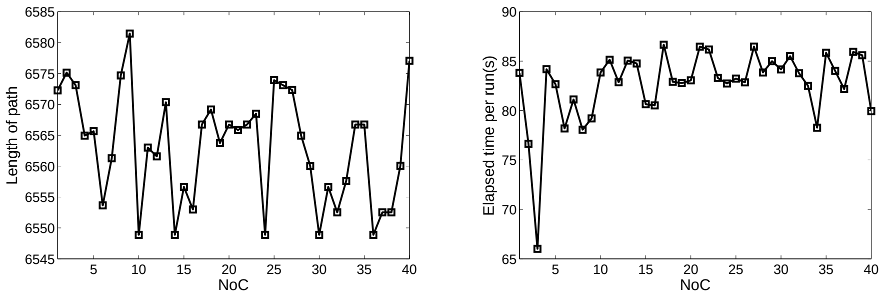
(a) Mean of best solutions



(b) Mean of elapsed time per run

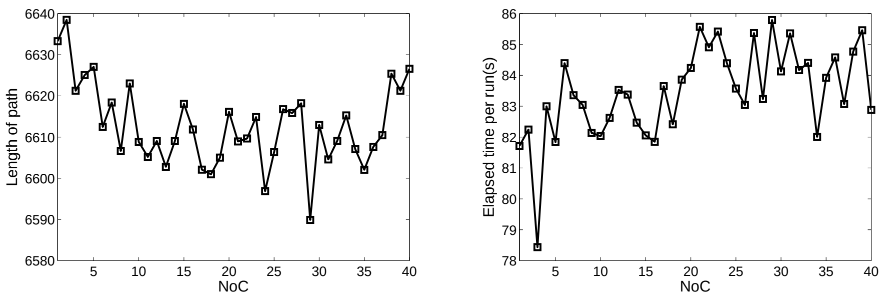
Fig. 18. Experimental results of Ch130 with NoC

and the NoFE, also listed in Table 1, for different TSPs as the termination criterion. Additionally, in the ACOPS, the number of elementary membranes, the parameters $g_{\min}$ and $g_{\max}$ are set to 4, 10 and 30, respectively. The performances



(a) Best solutions (b) Elapsed time per run

Fig. 19. Experimental results of Ch150 with NoC



(a) Mean of best solutions (b) Mean of elapsed time per run

Fig. 20. Experimental results of Ch150 with NoC

of the ACOPS and ACO are evaluated by using the following statistical results over 20 independent runs: the best, the worst and the average length of paths. Experimental results are listed in Table 1.

As shown in Table 1, the ACOPS achieves better results than the ACO in 19 out of 20 instances, in terms of the best and mean solutions. We go further to apply statistical techniques to analyze the behavior of the two algorithms, ACOPS and ACO, over the 20 TSPs. Parametric and non-parametric approaches are two main ways of statistical methods [25]. The parametric approach, also called single-problem analysis, employs a parametric statistical analysis *t*-test to check whether there is a significant difference between two algorithms applied to an optimization problem. The non-parametric approach, also called multiple-problem analysis, utilizes non-parametric statistical tests, such as Wilcoxon's and Friedman's tests, to compare different algorithms whose results represent average values for each problem, regardless of the inexistence of relationships among them. Thus, a 95% confidence Student *t*-test is first used to check whether there are significant differences between ACOPS and ACO. Two non-parametric tests, Wilcoxon's and

Table 1. A comparison between ACOPS and ACO ('+' and '-' represent significant difference and no significant difference, respectively. '—' means no optimum available)

TSP	NoFE	ACO			ACOPS			*t*-test	Optimum
		Best	Average	Worst	Best	Average	Worst		
ulysses16	1e+4	74.11	74.11	74.11	73.99	74.02	74.23	3.47e-6(+)	74
att48	2e+4	33588.34	33654.16	33740.35	33523.71	33644.97	34060.49	7.05e-1(-)	33524
eil76	3e+4	545.39	546.22	551.93	544.37	551.62	555.55	3.48e-7(+)	538
kroA100	4e+4	21577.69	21776.91	22320.91	21285.44	21365.64	21552.00	4.00e-8(+)	21282
eil101	4e+4	642.66	652.30	684.19	640.98	648.48	664.24	2.03e-1(-)	629
lin105	4e+4	14383.00	14472.38	14482.31	14383.00	14444.77	14612.43	8.61e-2(+)	14379
ch130	4.5e+4	6204.09	6268.43	6333.16	6148.99	6205.54	6353.69	1.02e-3(+)	6110
gr137	4.5e+4	718.92	725.02	749.93	709.91	718.85	738.35	6.63e-3(+)	—
pr144	5e+4	58587.14	58612.82	58687.80	58535.22	58596.00	58761.43	2.24e-1(-)	58537
ch150	5e+4	6595.00	6630.59	6689.79	6548.89	6570.86	6612.46	1.05e-7(+)	6528
rat195	6e+4	2370.24	2392.69	2434.39	2348.32	2355.23	2373.79	1.61e-7(+)	2323
d198	6e+4	16172.77	16266.93	16530.79	16073.13	16192.89	16381.91	3.75e-2(+)	15780
kroa200	6e+4	29597.01	29988.74	30466.71	29453.10	29552.92	29688.13	1.92e-6(+)	29437
gr202	6e+4	496.48	496.96	499.53	488.41	494.21	499.44	6.82e-4(+)	—
tsp225	7e+4	4067.96	4146.32	4262.76	3904.46	3971.68	4044.32	2.11e-9(+)	3916
gr229	7e+4	1739.77	1763.80	1802.44	1725.84	1756.28	1792.91	2.11e-1(-)	—
gil262	8e+4	2452.82	2487.58	2512.85	2407.68	2431.58	2450.65	4.86e-10(+)	2378
a280	9e+4	2626.44	2683.21	2787.61	2595.31	2636.49	2728.06	8.46e-4(+)	2579
pr299	10e+4	51050.78	52103.27	53698.23	49370.69	51021.74	52251.21	7.69e-4(+)	48191
lin318	10e+4	44058.08	45297.99	46410.50	42772.12	43433.54	45194.62	5.95e-9(+)	42029

Friedman's tests, are applied to check whether the two algorithms are significantly different or not. The level of significance considered is 0.05. Results of *t*-test are listed in Table 1. Results of Wilcoxon's and Friedman's tests are shown in Table 2. In Table 1 and 2, the symbols '+' and '-' represent significant difference and no significant difference, respectively. The *t*-test results in Table 1 demonstrate that there are 16 significant differences between the two algorithms. The *p*-values of the two non-parametric tests in Table 2 are far smaller than the level of significance 0.05, which indicates that the ACOPS really outperforms the ACO. It is worth noting that the study in [25] shows that the non-parametric statistical tests are more appropriate than parametric statistical tests in the analysis of the behavior of optimization algorithms over multiple optimization problems.

Table 2. Results of non-parametric statistical tests for ACOPS and ACO in Table 1. '+' represents significant difference

Tests	ACOPS vs ACO
Wilcoxon test(*p*-value)	1.6286e-004(+)
Friedman test(*p*-value)	5.6994e-005(+)

3.3 Comparisons with Nishida's Algorithms and Statistical Analysis

In [6, 7, 8], Nishida proposed a membrane algorithm combining an NMS P systems structure and a local search. Eight TSP benchmarks were applied to test the performances of the algorithm and its variants. In his experiments, the NMS with 2, 10, 30, 50, 70 and 100 membranes, respectively, was discussed. The maximal number of iterations, which can be equivalent to a certain number of function evaluations (NoFE), was used as the termination criterion. The number of trials was

20. The ACOPS algorithm stops according to a prescribed NoFE. The parameters of the ACOPS are assigned as follows: $M = 40$, $\alpha = 1$, $\beta = 3$, $\rho = 0.6$, $\upsilon = 0.1$, $q_0 = 0.9$, $m = 4$, $g_{\min} = 10$ and $g_{\max} = 30$. Experimental comparisons between the ACOPS and the Nishida's algorithm are listed in Tables 3-5, in which NoM represents the number of membranes. The results of the ACOPS are obtained from 20 independent runs. In Tables 3 and 4, the results of Nishida's algorithm are obtained from [7, 8] and the NoFE for each cases are calculated according to the number of iteration and the number of membranes. The performance of the ACOPS is also tested on the Eil51 and KroA100 TSPs with different NoFE. Table 5 lists the experimental results of Nishida's algorithm and the equivalent NoFE for each of the 8 TSP instances calculated by using 50 membranes and 100000 iterations. The results of Wilcoxon's and Friedman's tests are given in Table 6.

Table 3. Comparisons of Nishida's algorithm and ACOPS with Eil51 TSP

	Nishida's algorithm					**ACOPS**				
NoM	2	10	30	50	70	4				
NoFE	1.2e+5	7.6e+5	2.36e+6	3.96e+6	5.56e+6	1e+4	2e+4	3e+4	4e+4	5e+4
Best	440	437	432	429	429	429.4841	429.4841	428.9816	428.9816	428.9816
Worst	786	466	451	444	443	435.5985	436.3928	434.9739	433.6050	433.8558
Average	522	449	441	435	434	432.3858	431.8023	431.3146	430.5506	430.4495

Table 4. Comparisons of Nishida's algorithm and ACOPS with KroA100 TSP

	Nishida's algorithm						**ACOPS**					
NoM	2	10	30	50	70	100	4					
NoFE	3e+5	1.9e+6	5.9e+6	9.9e+6	1.39e+7	1.99e+7	1e+4	2e+4	4e+4	6e+4	8e+4	1e+5
Best	23564	21776	21770	21651	21544	21299	21331	21285	21285	21285	21285	21285
Worst	82756	24862	23940	24531	23569	22954	22332	21665	21552	21475	21427	21575
Average	34601	23195	22878	22590	22275	21941	21593	21407	21367	21337	21320	21362

Table 5. Comparisons of Nishida's algorithm and ACOPS with 8 TSPs

TSP	**Nishida's algorithm**				**ACOPS**			
	NoFE	Best	Average	Worst	NoFE	Best	Average	Worst
ulysses22	9.9e+7	75.31	75.31	75.31	2e+4	75.31	75.32	75.53
eil51	9.9e+7	429	434	444	4e+4	429	431	434
eil76	9.9e+7	556	564	575	6e+4	546	551	558
eil101	9.9e+7	669	684	693	8e+4	641	647	655
kroA100	9.9e+7	21651	22590	24531	1e+5	21285	21320	21427
ch150	9.9e+7	7073	7320	7633	1.2e+5	6534	6560	6584
gr202	9.9e+7	509.7	520.1	528.4	1.4e+5	489.2	492.7	497.1
tsp225	9.9e+7	4073.1	4153.6	4238.9	7e+4	3899.6	3938.2	4048.2

As compared with Nishida's algorithm, the ACOPS uses much smaller NoFE to achieve better solutions, which is shown in Table 3-5. Small NoFE means low computing complexity. The non-parametric statistical analysis shows that the two algorithms have also a significant difference.

Table 6. Results of non-parametric statistical tests for Nishida's algorithm and ACOPS in Table 5. '+' represents significant difference

Tests	**Nishida's algorithm vs ACOPS**
Wilcoxon test(p-value)	0.0156 (+)
Friedman test(p-value)	0.0339 (+)

4 Conclusions

This work is the first attempt to discuss the interaction between P systems and ant colony optimization. We present an approximate optimization algorithm combining the hierarchical structure of compartments and communication/transformation evolution rules of P systems, and the pheromone model of ant colony optimization. The introduced approach is used to solve the well-known and extensively studied NP-hard problem, traveling salesman problem. The better optimization performance of the ACOPS is verified by comparing it with its counterpart ACO and Nishida's algorithms. In order to thoroughly test the capabilities of this approach, our future studies will focus on the use of the ACOPS to producing solutions to some real-world engineering problems.

References

1. Păun, Gh.: Computing with Membranes, *Journal of Computer and System Sciences*, **61**(1), 2000, 108-143.
2. Păun, Gh., Rozenberg, G., Salomaa, G.:(Eds.), *Handbook of membrane computing.* Oxford: Oxford University Press, 2009.
3. Ionescu, M., Păun, Gh., Yokomori, T.: Spiking neural P systems, *Fundamenta Informaticae*, **71**(2-3), 2006, 279-308.
4. Zhang, G. X., Liu, C. X., Gheorghe, M., Ipate, F.: Solving Satisfiability Problems with Membrane Algorithm, *Proc. of the 4th International Conference on Bio-Inspired Computing: Theories and Applications*, 2009, 29–36.
5. Blum, C.: Ant colony optimization Introduction and recent trends, *Physics of Life Reviews*, **2**(4), 2005, 353-373.
6. Nishida, T. Y.: An application of P-system: A new algorithm for NP-complete optimization problems, *Proc. 8th World Multi-Conference on Systemics, Cybernetics and Informatics*, Orlando, 2004, 109-112.
7. Nishida, T. Y.: Membrane algorithms: Approximate algorithms for NP-complete optimization problems, *Applications of Membrane Computing*, 2006, 303-314.
8. Nishida, T. Y.: Membrane Algorithm: An Approximate Algorithm for NP-Complete Optimization Problems Exploiting P-Systems", *Proc. 6th International Workshop on Membrane Computing*, Vienna, 2005, 26-43.

9. Leporati ,A., Pagani, D.: A membrane algorithm for the min storage problem, *Proc. of Workshop on Membrane Computing*, **4361** Berlin: Springer, 2006, 443-462.
10. Huang, L., He, X. X., Wang, N., Xie, Y.: P Systems Based Multi-objective Optimization Algorithm, *Progress in Natural Science*, **17**(1), 2007, 458–465.
11. Huang, L., Wang, N.: An optimization algorithm inspired by membrane computing, *Proc. of ICNC*, **4222**, 2006, 49-52.
12. Zaharie, D., Ciobanu, G.: Distributed Evolutionary Algorithms Inspired by Membranes in Solving Continuous Optimization Problems, *Proc. of the WMC*, **LNCS 4361**, 2006, 536–553.
13. 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.
14. Liu, C. X., Zhang, G. X., Zhu, Y. H., Fang, C., Liu, H. W.: A Quantum-inspired evolutionary algorithm based on P systems for radar emitter signals, *the Fourth International Conference on Bio-Inspired Computing: Theories and Applications*, Beijing, 2009, 24-28.
15. Liu, C. X., Zhang, G. X., Liu, L. W., Gheorghe, M., Ipate, F.: An Improved Membrane Algorithm for Solving Time-Frequency Atom Decomposition, *WMC 2009, LNCS*, **5957**, Gh. Păun, Ed.: Springer, 2010, 371–384.
16. Liu, C. X., Zhang, G. X., Liu, H. W.: A Memetic Algorithm Based on P Systems for IIR Digital Filter Design," *Proc. of the 8th IEEE International Conference on Pervasive Intelligence and Computing*, 2009, 330-334.
17. Huang, L., Suh, I. H.: Controller design for a marine diesel engine using membrane computing, *International Journal of Innovative Computing Information and Control*, **5**(4), 2009, 899-912.
18. Păun, G., Pérez-Jiménez, M. J.: Membrane computing: Brief introduction, recent results and applications, *Biosystems*, **85**(1), 2006, 11-22, 2006.
19. Dorigo, M., Birattari, M., Stützle, T.: Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique, *IEEE Computational Intelligence Magazine*, **1**(4), 2006, 28-39.
20. Dorigo, M., Blum, C.: Ant colony optimization theory: a survey, *Theoretical Computer Science*, **344**(2-3), 2005, 243-278.
21. Uğur, A., Aydin, D.: An interactive simulation and analysis software for solving TSP using Ant Colony Optimization algorithms, *Advances in Engineering Software*, **40**(5), 2009, 341-349.
22. Păun, Gh.: Tracing some open problems in membrane computing, *Romanian Journal of Information Science and Technology*, **10**(4), 2007, 303-314.
23. Păun, Gh., Rozenberg, G.: A guide to membrane computing, *Theor. Comput. Sci.*, **287**(1), 2002, 73–100.
24. Wu, Z., Zhao, N., Ren, G., Quan, T.: Population declining ant colony optimization algorithm and its applications, *Expert Systems with Applications*, **36**(3), 2009, 6276-6281.
25. Garcia, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization, *J. Heuristics*. 10.1007/s10732 -008-9080-4.