INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 10(1):89-99, February, 2015.

An Approximate Algorithm Combining P Systems and Active Evolutionary Algorithms for Traveling Salesman Problems

X. Song, J. Wang

Xiaoxiao Song*
School of Electrical and Information Engineering
Xihua University, Chengdu, Sichuan, P.R. China, 610039
*Corresponding author: sxx_pippen@163.com

Jun Wang
School of Electrical and Information Engineering
Xihua University, Chengdu, Sichuan, P.R. China, 610039
745257101@qq.com

Abstract: An approximate algorithm combining P systems and active evolutionary algorithms (AEAPS) to solve traveling salesman problems (TSPs) is proposed in this paper. The novel algorithm uses the same membrane structure, subalgorithms and transporting mechanisms as Nishida's algorithm, but adopts two classes of active evolution operators and a good initial solution generating method. Computer experiments show that the AEAPS produces better solutions than Nishida's shrink membrane algorithm and similar solutions with an approximate optimization algorithm integrating P systems and ant colony optimization techniques (ACOPS) in solving TSPs. But the necessary number of iterations using AEAPS is less than both of them.
Keywords: P systems, active evolutionary algorithms, traveling salesman problems.

1 Introduction

Membrane computing, a milestone in natural computing, was introduced by Gheorghe Păun [1] in 1998. This computational model, which was inspired by the structure and the behavior of living cells, was proposed. In the following more than ten years, a sizeable group of researchers were seduced by membrane computing. Membrane algorithm is one of the research hotspots after Nishida [2–4] first proposed this concept by combining P systems and meta-heuristic search methodologies. Huang [5, 6] and Cheng [7] combined genetic algorithm and differential evolution with membrane systems to solve some single- and multi-objective optimization problems. Quantum-inspired evolutionary algorithm based on P systems was proposed to solve some classical theoretical [8,9] and practical problems [10–17]. Also some novel membrane algorithms based on particle swarm optimization [18] and artificial fish swarm algorithm [19] were proposed.

Evolutionary algorithm is based on survival of the fittest. Creatures do not have ability to decide their mutation directions and choose advantageous gene to their offspring. But based on the researches in biology, this stochastic evolution theory could not explain some problems in creature's adaptability. The modern biology research shows that the evolution process is not completely stochastic [20–22]. So called stress-induced mutation mechanisms were proposed. Specifically, when creatures are maladapted to their environment, that is, when they are stressed, stress-induced mutation mechanisms produce mutations [23–26]. These facts have been considered in evolutionary algorithm for solving optimization problems [27].

In this paper an approximate algorithm combining P systems and active evolutionary algorithms (AEAPS) is proposed in order to solve traveling salesman problems (TSPs) in the special case of complete graphs with Euclidean distance. It follows the nested membrane structure adopted by Nishida [2], and adopts genetic algorithm (GA), tabu search and active evolutionary algorithms (AEA) as the subalgorithms. Experiment results are compared with Nishida's

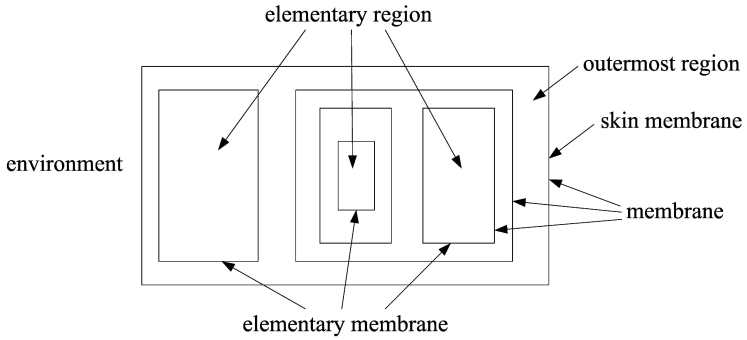

Figure 1: The membrane structure of a cell-like P system

algorithm and an approximate optimization algorithm integrating P systems and ant colony optimization techniques (ACOPS) [28, 29].

In Section 2, a brief introduction of P systems and Nishida's membrane algorithm for TSPs is given. Details of AEA designed for TSPs and AEAPS is discussed in Section 3. The experiment results and analysis are mentioned in Section 4. Conclusions are drawn in Section 5.

2 P Systems and membrane algorithm

2.1 P Systems

P systems could be divided into three groups: cell-like P systems, tissue-like P systems and neural-like P systems [30]. The structure of cell-like P systems is the basic structure of other P systems. The membrane structure of a cell-like P system is shown in Fig. 1. The outermost membrane is the skin membrane. Outside of the skin membrane is the environment. Usually, there are some other membranes inside the skin membrane. We call the spaces between membranes regions. The region just inside the skin membrane is the outermost region, and the region in an elementary membrane is an elementary region. In membrane computing, regions contain multisets of objects and sets of evolution rules.

A cell-like P system is formally defined as follows [1, 31]:

$$\Pi = [V, T, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, i_0]\,. \tag{1}$$

where:
(i) V is an alphabet; its elements are called objects;
(ii) $T \subseteq V$ is the output alphabet;
(iii) μ is a membrane structure consisting of m membranes; m is called the degree of Π;
(iv) w_i, $1 \leq i \leq m$, is a string representing the initial multiset over V associated with region i, $1 \leq i \leq m$;
(v) R_i, $1 \leq i \leq m$, is a finite set of evolution rules associated with region i, $1 \leq i \leq m$;
(vi) i_0 is a number between 1 and m which specifies the output membrane of Π.

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), $b \in V$ and $t \in \{here, out, in\}$. *here* means when the rule is used in one region, b will stay in the region; *out* means that b exits the region and *in* means that b will be communicated to one of the membranes contained in the current region.

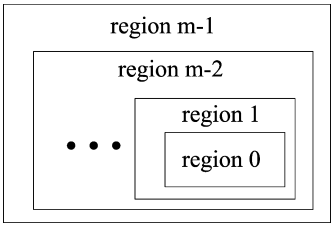

Figure 2: Membrane structure of membrane algorithm

2.2 The membrane algorithm for TSPs

Membrane algorithm is designed with the hierarchical or network structure of membranes and rules of P systems, and the concepts and principles of meta-heuristic search methodologies. It is a new kind of parallel-distributed framework for solving optimization problems. Nishida first proposed a membrane algorithm using cell-like P systems (nested membrane structure) [2] to solve TSPs. Nishida also proposed some improved membrane algorithms based on tissue-like P systems, such as compound membrane algorithm [3] and shrink membrane algorithm [4]. All the basic concepts of improved algorithms are based on the membrane algorithm. For example, compound membrane algorithm has two phases. The first phase is using membrane algorithm generating good initial solutions for phase 2; and the second phase is also similar to membrane algorithm but using good initial solutions. The shrink membrane algorithm incorporates dynamic membrane structure into compound membrane algorithm. The membrane algorithm with nested membrane structure is a special case of multi-deme evolutionary algorithm [32]. In this paper, we only research membrane algorithm with this structure.

In Nishida's membrane algorithm, nested membrane structure, rules in membrane separated regions and transporting mechanisms through membrane from P systems are adopted. The structure of the membrane algorithm is shown in Fig. 2.

In solving TSPs, the membrane algorithm can be described as follows:
1. Generate one initial solution in region 0 and two initial solutions in all regions from 1 to $m-1$ respectively;
2. In one iteration, the solution in region 0 is updated by tabu search and the solutions in regions from 1 to $m-1$ are updated by genetic algorithm, simultaneously;
3. Regions from 1 to $m-2$ send the best solution to adjacent inner region, and the worst solution to adjacent outer respectively. Region 0 sends the worst solution to region 1 and region $m-1$ sends the best solution to region $m-2$.
4. Erases solutions but the best two in regions from 1 to $m-1$ and the best one in region 0.
5. Jump to step 2 if the termination condition is not satisfied; otherwise the output of the algorithm is the solution in region 0.

3 AEAPS for TSPs

In active evolution organisms adapt their behaviour to changing environment. In TSPs, the "environment" means the structure characteristics of a solution, like without crossings in its path of traveling. If a solution is maladapted to the "environment", some active mutation mechanisms should be considered to improve it. In this section we propose two classes of active evolution conditions and show how to deal with these conditions; then we give a simple method of obtaining good initial solutions; finally, every step of AEAPS is described.

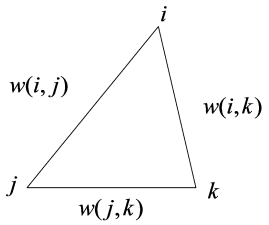

Figure 3: Triangle inequality for (G, w)

3.1 1^{st} class active evolution condition

TSP is one of the well-known combinatorial optimization problems. The TSP problem is about finding the Hamilton cycle. i.e., the optimum shortest path of a given weighted undirected and connected graph (G, w) with N nodes and where w is a distance metric. This distance is symmetric, which means $w(i, j) = w(j, i)$. In the two dimensional space, the distance between vertex i and vertex j is

$$w(i, j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \tag{2}$$

where i and j have the coordinates (x_i, y_i) and (x_j, y_j), respectively.

The value of one solution is

$$V = \sum_{i=1}^{N-1} w(i, i+1) + w(N, 1). \tag{3}$$

Since w is a distance, then it satisfies the triangle inequality, i.e., $w(i, j) \leq w(i, k) + w(j, k)$, for any vertices $i \neq j$, and $k \neq i, j$ (see Fig. 3).

It has been shown that 2-Opt iterative improvement method [10, 33, 35] leads to optimum solutions for the TSP without any crossings. This means that any optimal path contains edges of the graph that do not intersect each other. If we consider four nodes. The total graph associated with these four nodes shows convex and concave quadrilaterals. When the convex quadrilateral is considered, any two edges are disjoint (they do not intersect each other). This is no longer true for the concave quadrilateral. In Fig. 4, we select a as the starting node, and there can be 6 possible solutions. Solutions *abcda* and *adcba* have no crossings, and solutions *abdca*, *acbda*, *acdba* and *adbca* have crossings. One can easily show that the solutions without crossings are better than those with crossings, by referring to the triangle inequality. For example, the path *abdca*, has two edges, (a, b) and (d, c), overlapping the path *abcda* ($(d, c) = (c, d)$). For the path *abdca*, we have $w(b, d) = w(b, o) + w(o, d)$ and $w(c, a) = w(c, o) + w(o, a)$. According to the triangle inequality, we have

$$\begin{aligned} w(b, d) + w(c, a) &= w(b, o) + w(o, d) + w(c, o) + w(o, a) \\ &= w(b, o) + w(c, o) + w(o, d) + w(o, a) \\ &\geq w(b, c) + w(a, d) \end{aligned} \tag{4}$$

So we found solution *abcda* without crossings is better than solution *abdca* with crossings. With this method, other solutions with crossings can also be transformed into a better ones without crossings. One can conclude stating that for the TSP problem, with Euclidian distance, for any solution with crossings there is always a better one without crossings.

In our AEAPS method we use a specific stress-induced mutation to the current solution. This mutation operator is different from the usual one theory of evolution. This is an active evolution

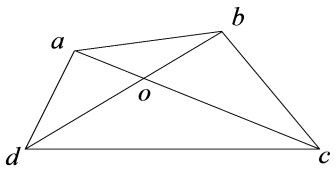

Figure 4: Example with four nodes

$$\text{Old solution}: \quad 1,2,\cdots,i-1,i,i+1,\cdots,j-1,j,j+1,\cdots N$$
$$\Downarrow 1^{\text{st}} \text{ class active evolution}$$
$$\text{New solution}: \quad 1,2,\cdots,i-1,j+1,j,j-1,\cdots,i+1,i,\cdots N$$

Figure 5: A 1^{st} class active evolution condition applied

operator. If in the current path there is an edge from i to $i+1$ that crosses other existing edge in the path, then i is called a 1^{st} class active evolution node. The method of revising the path according to a 1^{st} class active evolution node is the following:
1. Find the nearest N_1 nodes to i and denote by A the set consisting of these nodes;
2. Select one node from the set A and name it j;
3. Find out node $j+1$ which is the next node after j in the solution;
4. Swap the nodes i and $j+1$ in the solution;
5. If the value V of the new solution is less than the old one, keep the new one, otherwise keep the old one.

An illustration of the method is provided in Fig. 5.

3.2 2^{nd} class active evolution condition

If the distance from i to the next node, $i+1$, is larger than some value D, i is called a 2^{nd} class active evolution node. In this case insert a new node between i and $i+1$. Also compute$D_i = e \times total_distance_i/(N-1)$, where e is a parameter, $total\_distance_i$ is the total distance between i and the other nodes and N is the number of nodes.

In AEAPS we consider an approach similar to 2.5-Opt iterative improvement scheme [36] for dealing with 2^{nd} class active evolution nodes. This method is described below, where i is a 2^{nd} class active evolution node and $i+1$ is the next node after i in the current solution:
1. Find the nearest N_2 nodes to i and put them all into a set A; similarly build the set B for $i+1$;
2. Select one element from $A \cap B$ and name it j;
3. Eliminate j from the solution and insert it between i and $i+1$;
4. If the value of the new solution is less than the old one, keep the new one, otherwise keep the old one.

An illustration of the use a 2^{nd} class active evolution node is shown in Fig. 6.

3.3 Initial solutions

Nishida proposed a membrane algorithm, called compound membrane algorithm, which has two phases. The function of the first phase is producing good solutions which are used as initial

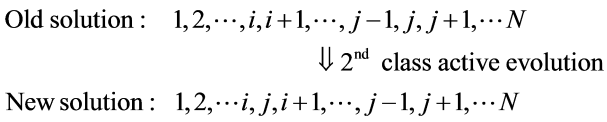

Figure 6: A 2^{nd} class active evolution condition applied

solutions for phase two. The better initial solutions can improve the final output solution. As Nishida said the computation time of compound membrane algorithm is quite prohibitive. We propose a simpler and faster method for generating good initial solutions. The method is as follows:
1. Select one node randomly as the starting node, and name it current node;
2. Find the nearest neighbour of the current node which is not selected, and name it current node;
3. Repeat step 2 until all nodes are selected; a good initial solution has been then found.

If we repeat the above steps for $2 \times m - 1$ times, we get enough initial solutions for each region of the P system. We have one solution in region 0 and two solutions in each of the regions 1 to $m - 1$.

3.4 AEAPS algorithm

AEAPS uses the basic idea of the membrane algorithm proposed by Nishida; a nested membrane structure with m regions is considered. We still use tabu search in region 0 and genetic algorithms in regions 1 to $m-1$ as sub-algorithms. Finally, the same communication mechanisms between adjacent regions are used; the best and worst solutions are sent to adjacent inner and outer regions, respectively. Unlike the Nishida's algorithm, AEAPS adds two classes of active evolution operators in every region and use a new initial solution generation method.

The overall membrane algorithm can be described as follows:
1. Generate initial solutions by using the method mentioned in Section 3.3, one for region 0 and two for each of the regions from 1 to $m - 1$;
2. Modify solutions simultaneously in each of the regions 1 to $m-1$ by using genetic algorithms, simultaneously;
3. Find out all 1^{st} class active evolution nodes in every solution and revise them; then find out all 2^{nd} class active evolution nodes and also revise them all;
4. Use tabu search in region 0;
5. Use the communication mechanisms between adjacent regions (as proposed by Nishida);
6. Remove all solutions but the best one from region 0 and best two in each of the regions 1 to $m - 1$;
7. Jump to step 2 if the number of iterations is not satisfied; otherwise the output of the algorithm is the solution in region 0.

4 Experiments and Results

We have tested the searching efficiency of AEAPS on two benchmark problems, eil51, with 51 nodes, and kroA100, with 100 nodes, from TSPLIB [37], running 10 times each. The parameters in experiments are chosen as follows: $m = 50$, $e = 0.1$, $N_1 = 25$, $N_2 = 35$. The number of iterations is 300. Table 1 shows the results. A comparison of simulated annealing (SA), shrink

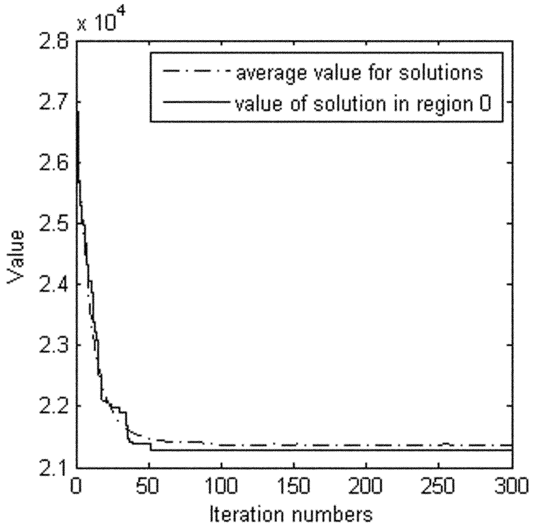

Figure 7: Curves of solving kroA100 problem by AEAPS

membrane algorithm (SMA) and AEAPS is shown in Table 2. We have implemented AEAPS in C and tested the algorithm on a Microsoft Visual C++ 6.0 platform with Windows 7 and using a computer with 2.4GHz CPU and 2G RAM.

Table 1. Results of AEAPS for eil51 and kroA100

	1	2	3	4	5	6	7	8	9	10
$eil51$	431	429	428	428	426	430	429	428	429	427
$kroA100$	21282	21282	21320	21282	21389	21282	21373	21379	21282	21282

Table 2. A comparison of SA, SMA and AEAPS

	SA			SMA			$AEAPS$		
	$Best$	$Average$	$Worst$	$Best$	$Average$	$Worst$	$Best$	$Average$	$Worst$
$eil51$	430	438	445	429	430	433	426	429	431
$kroA100$	21369	21763	22564	21299	21504	21750	21282	21315	21389

Results of SA and SMA from [4] are shown in the tables. From Table 1 and 2, one can see that AEAPS gets better results than SA and SMA for both eil51 and kroA100. Fig. 7 shows the curves of average values for all the solutions and the average value of the solution in region 0 for the kroA100 problem solved by AEAPS. For initial solutions which are generated by the method in Section 3.3, the average value of the initial solutions in AEAPS is much smaller than those using the membrane algorithm [4]. Compared to Nishida's algorithm for solving the same problem by the 50 membranes, AEAPS converges to remarkably fast to good solutions, in approximately 50 steps.

Zhang [28, 29] proposed ACOPS for TSP, which uses a smaller number of function evaluations to achieve better solutions. Experimental comparisons between Nishida's algorithm, ACOPS and AEAPS are listed in Table 3 and Table 4. Results of Nishida's algorithm and ACOPS are from [28]. The results of Nishida's algorithm were calculated by using 50 membranes and 10000 iterations. In ACOPS the number of function evaluations (NoFE) is the

stopping criterion. An equivalent number of iterations was obtained by using the product of average iterations for elementary membranes $(g_{min}+g_{max})/2$ and the number of communications $2*(g_{min}+g_{max})*\text{NoFE}/(N+m)$. In solving ulysses22, eil51, eil76, eil101 and kroA100 problems, the parameters of AEAPS have the values: $m = 50$, $e = 0.1$, $N_1 = 25$, $N_2 = 35$, and the maximal number of iterations is 300. In solving ch150, gr202 and tsp225 problems, the parameters of AEAPS are: $m = 50$, $e = 0.1$, $N_1 = 40$, $N_2 = 50$, and the maximal number of iterations is 500. The results of the AEAPS are obtained from 10 independent runs.

Table 3. Number of iterations in Nishida's algorithm, ACOPS and AEAPS with 8 TSPs

	Nishida's algorithm	*ACOPS*	*AEAPS*
*ulysses*22	1.0e+5	7.7e+2	3.0e+2
*eil*51	1.0e+5	7.3e+2	3.0e+2
*eil*76	1.0e+5	7.5e+2	3.0e+2
*eil*101	1.0e+5	7.6e+2	3.0e+2
*kroA*100	1.0e+5	9.6e+2	3.0e+2
*ch*150	1.0e+5	7.8e+2	5.0e+2
*gr*202	1.0e+5	6.8e+2	5.0e+2
*tsp*225	1.0e+5	3.1e+2	5.0e+2

Table 4. Results of Nishida's algorithm, ACOPS and AEAPS with 8 TSPs

	Nishida's algorithm			*ACOPS*			*AEAPS*		
	Best	*Average*	*Worst*	*Best*	*Average*	*Worst*	*Best*	*Average*	*Worst*
*ulysses*22	75.31	75.31	75.31	75.31	75.32	75.53	75.31	75.31	75.31
*eil*51	429	434	444	429	431	434	426	429	431
*eil*76	556	564	575	546	551	558	543	545	547
*eil*101	669	684	693	641	647	655	631	638	643
*kroA*100	21651	22590	24531	21285	21320	21427	21282	21315	21389
*ch*150	7073	7320	7633	6534	6560	6584	6549	6554	6565
*gr*202	509.7	520.1	528.4	489.2	492.7	497.1	491.3	496.1	498.9
*tsp*225	4073.1	4153.6	4238.9	3899.6	3938.2	4048.2	3938.7	3992.3	4034.1

As compared with Nishida's algorithm, AEAPS uses much smaller number of iterations to achieve better solutions. As compared with ACOPS, AEAPS uses smaller number of iterations only except for the *tsp*225 and gets better results for the first 5 TSPs, similar results for ch150 problem and slightly worse results for the last 2 TSPs.

5 Conclusions

This work is the first attempt to discuss the role of active evolutionary operators in membrane algorithms. We present an approximate algorithm combining nested membrane structure, rules within regions and communication mechanisms of the P systems, and two classes of active

evolution operators and a good initial solution generating method. AEAPS is used to solve Euclidian TSPs, well-known NP-hard problems. The experiment results show that AEAPS performs better than SA and Nishida's membrane algorithm and similar with ACOPS, which requires a smaller number of iterations. In order to improve the performance of AEAPS, especially in solving large scale TSPs, our future studies will focus on other membrane structure options and communication mechanisms.

Acknowledgments.

This work is supported by the National Natural Science Foundation of China (Grant No. 61170030), the Chunhui Plan of Ministry of Education of China (Grant No. Z2012025), the Open Research Fund of Key Laboratory of Xihua University (Grant No. SZJJ2012-002), the Research Projects of Eduction Department of Sichuan Province (Grant No. 13ZB0017) and the Key Scientific Research Foundation of Xihua University (Grant No. Z1120943). The authors also gratefully acknowledge helpful comments and suggestions made by reviewers, which have significantly improved the presentation.

Bibliography

[1] Păun, G. (2000); Computing with membranes, *Journal of Computer and System Sciences*, ISSN 0022-0000, 61(1): 108–143.

[2] Nishida, T.Y. (2004); An application of P-system: A new algorithm for NP-complete optimization problems, *Proceedings of the 8th World Multi-Conference on Systems, Cybernetics and Informatics*, V: 109–112.

[3] Nishida, T.Y. (2005); An approximate algorithm for NP-complete optimization problems exploiting P-systems, *Proceedings of the 6th International Workshop on Membrane Computing*, ISBN 978-3-540-30948-2, 26–43.

[4] Nishida, T.Y. (2006); Membrane algorithms: Approximate algorithms for NP-complete optimization problems, *Applications of Membrane Computing*, ISBN 978-3-540-29937-0, 303–314.

[5] Huang, L.; He, X.X.; Wang, N.; Xie, Y. (2007); P systems based multi-objective optimization algorithm, *Progress in Natural Science*, ISSN 1002-0071, 17(4): 458–465.

[6] Huang, L.; Wang, N. (2006); An optimization algorithm inspired by membrane computing, *ICNC 2006, LNCS*, ISBN 3-540-45901-4, 4222: 49–52.

[7] Cheng, J.X.; Zhang, G.X.; Zeng, X.X. (2011); A novel membrane algorithm based on differential evolution for numerical optimization, *International Journal of Unconventional Computing*, ISSN: 1548-7199, 7(3): 159–183.

[8] Zhang, G.X.; Liu, C.X.; Gheorghe, M.; Ipate, F. (2009); Solving satisfiability problems with membrane algorithm, *Proceedings of the 4th International Conference on Bio-Inspired Computing: Theories and Applications*, ISBN 978-1-4244-3866-2, 29–36.

[9] Zhang, G.X.; Gheorghe, M.; Wu, C.Z. (2008); A quantum-inspired evolutionary algorithm based on P systems for knapsack problem, *Fundamenta Informaticae*, ISSN 0169-2968, 87(1): 93–116.

[10] Liu, C.X.; Zhang, G.X.; Zhu, Y.H.; Fang, C.; Liu, H.W. (2009); A quantum-inspired evolutionary algorithm based on P systems for radar emitter signals, *Proceedings of the 4th International Conference on Bio-Inspired Computing: Theories and Applications*, ISBN 978-1-4244-6438-8, 1–5.

[11] Liu, C.X.; Zhang, G.X.; Liu, L.W.; Gheorghe, M.; Ipate, F. (2010); An improved membrane algorithm for solving time-frequency atom decomposition, *WMC 2009. LNCS*, ISSN 0302-9743, 5957: 371–384.

[12] Liu, C.X.; Zhang, G.X.; Liu, H.W. (2009); A memetic algorithm based on P systems for IIR digital filter design, *Proceedings of the 8th IEEE International Conference on Pervasive Intelligence and Computing*, ISBN: 978-0-7695-3929-4, 330–334.

[13] Huang, L.; Suh, I.H. (2009); Controller design for a marine diesel engine using membrane computing, *International Journal of Innovative Computing Information and Control*, ISSN 1349-4198, 5(4): 899–912.

[14] Zhang, G.X.; Liu, C.X.; Rong, H.N. (2010); Analyzing radar emitter signals with membrane algorithms, *Mathematical and Computer Modelling*, ISSN 0895-7177, 52(11-12): 1997–2010.

[15] Yang, S.P.; Wang, N. (2012); A P systems based hybrid optimization algorithm for parameter estimation of FCCU reactor-regenerator model, *Chemical Engineering Journal*, ISSN 1385-8947, 211: 508–518.

[16] Zhang, G.X.; Gheorghe, M.; Li, Y.Q. (2012); A membrane algorithm with quantum-inspired subalgorithms and its application to image processing, *Natural Computing*, ISSN 1567-7818, 11(4): 701–717.

[17] Zhang, G.X.; Cheng, J.X.; Gheorghe, M.; Meng, Q. (2013); A hybrid approach based on differential evolution and tissue membrane systems for solving constrained manufacturing parameter optimization problems, *Applied Soft Computing*, ISSN 1568-4946, 13(3): 1528–1542.

[18] Zhang, G.X.; Zhou, F.; Huang, X.L. (2012); A Novel membrane algorithm based on particle swarm optimization for optimization for solving broadcasting problems, *Journal of Universal Computer Science*, ISSN 0948-695X, 18(13): 1821–1841.

[19] Tu, M.; Wang, J.; Song, X.X.; Yang, F.; Cui, X.R. (2013); An artificial fish swarm algorithm based on P systems, *ICIC Express Letters, Part B: Applications*, ISSN 1881-803X, 4(3): 747–753.

[20] Cairns, J.; Overbaugh, J.; Miller, S. (1988); The origin of mutations, *Nature*, ISSN 0028-0836, 335: 142–145.

[21] Hall, B.G. (1988); Adaptive evolution that requires multiple spontaneous mutations. I. Mutations involving an insertion sequence, *Genetics*, ISSN 0016-6731, 120(4): 887–897.

[22] Hall, B.G. (1991); Is the occurrence of some spontaneous mutations directed by environmental challenges?, *The new biologist*, ISSN 1043-4674, 3(8): 729–733.

[23] Ponder, R.G.; Fonville, N.C.; Rosenberg, S.M. (2005); A switch from high-fidelity to error-prone DNA double-strand break repair underlies stress-induced mutation, *Molecular Cell*, ISSN 1097-2765, 19(6): 791–804.

[24] Slack, A.; Thornton, P.C.; Magner, D.B.; Rosenberg, S.M.; Hastings, P.J. (2006); On the mechanism of gene amplification induced under stress in Escherichia coli, *Plos Genetics*, ISSN 1553-7390, 2(4): 385–398.

[25] Galhardo, R.S.; Hastings, P.J.; Rosenberg, S.M. (2007); Mutation as a stress response and the regulation of evolvability, *Critical Reviews in Biochemistry and Molecular Biology*, ISSN 1040-9238, 42(5): 399–435.

[26] Rosenberg S.M.; Shee, C.; Frisch, R.L.; Hastings, P.J. (2012); Stress-induced mutation via DNA breaks in Escherichia coli: a molecular mechanism with implications for evolution and medicine, *Bioessays*, ISSN 0265-9247, 34(10): 885–892.

[27] Shi, L.; Li, H.Y.; Yang, J.A. (2004); Active evolution based genetic algorithm, *Mini-micro Systems*, ISSN 1000-1220, 5(25): 790–793.

[28] Zhang, G.X.; Cheng, J.X.; Gheorghe M. (2010); An approximate algorithm combining P systems and ant colony optimization for traveling salesman problems, *Proceedings of the 8th Brainstorming Week on Membrane Computing*, ISBN 978-84-614-2357-6, 321–340.

[29] Zhang, G.X.; Cheng, J.X.; Gheorghe M. (2011); A membrane-inspired approximate algorithm for traveling salesman problems, *Romanian Journal of Information Science and Technology*, ISSN 1453-8245, 14(1): 3–19.

[30] Păun, G. (2007); Tracing some open problems in membrane computing, *Romanian Journal of Information Science and Technology*, ISSN 1453-8245, 10(4): 303–314.

[31] Păun, G.; Rozenberg, G. (2002); A guide to membrane computing, *Theoretical Computer Science*, ISSN 0304-3975, 287(1): 73–100.

[32] Erick, C.P. (1998); A survey of parallel genetic algorithms, *Calculateurs Paralleles*, ISSN 1260-3198, 10(2): 141–171.

[33] Croes, G.A. (1958); A method for solving traveling salesman problems, *Operations Research*, ISSN 0030-364X, 6(6): 791–812.

[34] Lin, S.; Kernighan, B.W. (1973); An Effective Heuristic Algorithm for the Traveling-Salesman Problem, *Operations Research*, ISSN 0030-364X, 21(2): 498–516.

[35] Helsgaun K. (2000); An effective implementation of the Lin-Kernighan traveling salesman heuristic, *European Journal of Operational Research*, ISSN 0377-2217, 126(1): 106–130.

[36] Bentley, J.J. (1992); Fast algorithm for geometric traveling salesman problems, *Informs Journal on Computing*, ISSN 1091-9856, 4(4): 387–411.

[37] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/