An approximate algorithm for NP-complete optimization problems exploiting P-systems

Taishin Y. Nishida*
Faculty of Engineering
Toyama Prefectural University
Kosugi-machi, 939-0398 Toyama, Japan

Abstract

A new approximate algorithm for optimization problems, called membrane algorithm, are proposed, which is an application of G. Păun's membrane computing or P-system. Membrane algorithm consists of several membrane separated regions and a subalgorithm and a few tentative solutions of the optimization problem to be solved in every region. Subalgorithms improve tentative solutions problem simultaneously. Then the best and worst solutions in a region are sent to adjacent inner and outer regions, respectively. By repeating this process, a good solution will appear in the innermost region. The algorithm terminates if a terminate condition is satisfied. A simple terminate condition is the number of iterations, while a little sophisticated condition becomes true if the good solution is not changed during a predetermined period. Computer experiments show that the algorithm solves the traveling salesman problem as good as simulated annealing algorithm.

Key words: approximate algorithm, traveling salesman problem, P-system

1 Introduction

Studies on approximate algorithms for NP-complete problems are a very important issue in computer science because ([1, 2, 6]):

- There are thousands of NP-complete problems.
- Almost all NP-complete problems correspond to practical problems.
- There are very few (I think no) expectations for P = NP, or strictly solving NP-complete problems in polynomial time.

We suggest a new approximate algorithm for solving NP complete optimization problems. The algorithm uses P-system paradigm [4]. Then it is called *membrane algorithm*. Membrane algorithm borrows nested membrane structures, rules in membrane separated regions, and

*email: nishida@pu-toyama.ac.jp

transporting mechanisms through membranes from P-systems. Membrane algorithm remakes these components to solve NP-complete optimization problems approximately.

In the next section, the outline of membrane algorithm is explained. Details membrane algorithm are defined in order to solve the traveling salesman problem approximately in Section 3. The section also describes results of computer experiments under the definitions. An advanced membrane algorithm is mentioned in Section 4.

2 The outline of membrane algorithm

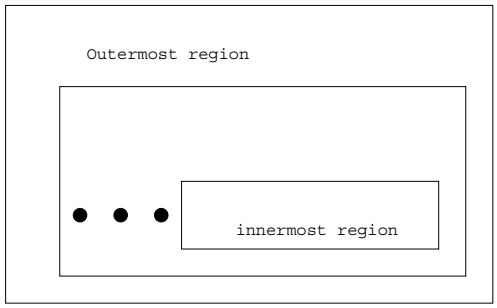

Figure 1: Membrane structure of the suggested algorithm.

Here we explain the new algorithm, called *membrane algorithm*. Membrane algorithm consists of three different kinds of components:

1. A number of regions which are separated by nested membranes (Figure. 1).
2. For every region, a subalgorithm and a few tentative solutions of the optimization problem to be solved.
3. Solution transporting mechanisms between adjacent regions.

After initial settings, membrane algorithm works as follows:

1. For every region, the solutions are updated by the subalgorithm at the region, simultaneously.
2. In every region, the best and worst solutions, with respect to the optimization, are sent to the adjacent inner and outer regions, respectively.
3. Membrane algorithm repeats updating and transporting solutions until a terminate condition is satisfied. A simple terminate condition is the number of iterations, while a little sophisticated condition becomes true if the good solution is not changed during a predetermined period.

The best solution in the innermost region is the output of the algorithm.

Membrane algorithm can have a number of subalgorithms which are any approximate algorithm for optimization problems, for example, genetic algorithm, tabu search, simulated annealing, local search, and so forth. The algorithm is expected to be able to escape from local minima by using a subalgorithm which likes random search at outer regions. On the other hand, the algorithm can improve good solutions in the inner regions by a subalgorithm which likes local search. So, assigning appropriate subalgorithms for a given problem, performance of the algorithm will be excellent.

Because the subalgorithms are separated by membranes and communications occur only between adjacent regions, membrane algorithm will be easily implemented in parallel, distributed, or grid computing systems. This is the second superior point of the algorithm.

3 First experiment of membrane algorithm solving traveling salesman problem

In this section we fix components of membrane algorithm to solve traveling salesman problem (TSP for short). Then we implement and experiment the algorithm on a computer.

3.1 Details of the algorithm

Let m be the number of membranes and let region 0 be the innermost and region $m-1$ be the outermost regions, respectively.

An instance of TSP with n nodes contains n pairs of real numbers (x_i, y_i) $(i = 0, 1, \ldots, n-1)$ which correspond to points in the two dimensional space. The distance between two nodes $v_i = (x_i, y_i)$ and $v_j = (x_j, y_j)$ is the geometrical distance $d(v_i, v_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. A solution is a list of nodes $(v_0, v_1, \ldots, v_{n-1})$. The *value* of a solution $v = (v_0, v_1, \ldots, v_{n-1})$ denoted by $W(v)$ is given by

$$W(v) = \sum_{i=0}^{n-2} d(v_i, v_{i+1}) + d(v_{n-1}, v_0).$$

For two solutions u and v, v is better than u if $W(v) < W(u)$. The solution which has the minimum value in all possible solutions is said to be the *strict solution* of the instance. A solution which has a value close to the strict solution is called an approximate solution.

The algorithm has one tentative solution in region 0 and two solutions in regions 1 to $m-1$.

We use a tabu search as the subalgorithm in the innermost region, region 0. Tabu search searches a neighbour of the tentative solution by exchanging two nodes in the solution. In order to avoid appearing the same solution twice, tabu search has a tabulist which consists of nodes already exchanged. Nodes in the tabulist are not exchanged again. Tabu search resets the tentative solution and the tabulist if one of the three conditions occurs:

1. The value of the neighbouring solution is less than that of the tentative solution. The neighbouring solution becomes the tentative solution.

2. The value of the best solution in the region 1 is less than that of the tentative solution. The best solution in the region 1 becomes the new tentative solution.

3. Neighbour search exceeds a predetermined turns (in this case $\frac{n}{5}$). The tentative solution remains. Only tabulist is reset.

In case 3, no improvement occurs. But tabu search tries to search other neighbours, since there are many unsearched neighbours.

The tentative solutions in regions 1 to $m-1$ (there are two solutions in each region) are improved by a subalgorithm summerized below:

1. If the two solutions have the same value, then a part of one solution (which is selected probabilisticly) is reversed.

2. Recombinates the two solutions and makes two new solutions.

3. Modifies the two new solutions by point mutations. In the i-th region, a mutation occurs under probability $\frac{i}{m}$.

Obviously the subalgorithm described above resembles genetic algorithms. But the subalgorithm always recombinates the two solutions in a region while genetic algorithms randomly select solutions to be recombinated. If the two solutions in a region are identical, recombination makes no new solutions. The step 1 avoids this case and introduces a new solution using reverse operation, which is a kind of mutation.

The overall algorithm looks like:

1. Given an instance of TSP.

2. Randomly makes one tentative solution for region 0 and two tentative solutions for every region 1 to $m-1$.

3. Repeats 3.1 to 3.3 for d times (d is given as a parameter).

   3.1 Modify tentative solutions simultaneously in every region using the subalgorithm at the region.

   3.2 For every region i ($1 \leq i \leq m-2$), sends the best solution of the solutions in the region (old solutions and modified solutions) to region $i-1$ and the worst solution to region $i+1$. (In region 0, sends the worst solution to region 1 and in region $m-1$, sends the best solution to region $m-2$.)

   3.3 For every region 1 to $m-1$ erases solutions but the best two.

4. Outputs the tentative solution in region 0 as the output of the algorithm.

In the above algorithm, steps 3.2 and 3.3 correspond to solution transporting mechanisms between adjacent regions.

Table 1: Results of membrane algorithm and a simulated annealing (SA) for the benchmark problem eil51 (51 nodes). Membrane algorithm repeat step 3 40000 times. The number of trials of membrane algorithm is 10. Membrane 2, 10, 30, and 50 stand for the algorithms with 2, 10, 30, and 50 regions, respectively.

Algorithm	Membrane 2	Membrane 10	Membrane 30	Membrane 50	SA
Best	440	437	433	429	430
Average	544	450	442	435	438
Worst	786	457	450	444	445

Table 2: Results for benchmark problem kroA100 (100 nodes). 100000 iterations and 10 trials.

Algorithm	Membrane 2	Membrane 10	Membrane 30	Membrane 50	SA
Best	24524	22319	21770	21651	21369
Average	32973	23422	23200	22590	21763
Worst	49667	24862	23940	24531	22564

3.2 Computer experiments

We have implemented the algorithm using Java programming language. By using Java, modifications of the algorithm have been easily tested on a computer. For example, we have implemented several recombination methods and have found that edge exchange recombination (EXX) [3] exhibits the best performance.

Tables 1 and 2 show results of the program for TSP benchmark problem eil51[1] and kroA100[2] from TSPLIB [5]. Results of simulated annealing from [7] are also shown in the tables.

Figure 2 shows changes of the average value of solutions for kroA100 problem solved by membrane algorithm with 50 membranes. One can see that the algorithm converges to considerable good solutions in a few steps, about 2000 to 3000 steps.

4 An improved membrane algorithm

In this section we discuss an improved membrane algorithm, called compound membrane algorithm, which corresponds to a tissue P-system.

Compound membrane algorithm has two phases (Figure 3). In the first phase, a number of membrane algorithms make good solutions from randomly generated initial solutions. The good solutions, in turn, become the initial solutions of the second phase. And a better solution is obtained.

[1] The value of the optimum solution is 426.
[2] The value of the optimum solution is 21282.

Figure 2: Changes of the average value of solutions for kroA100 problem solved by membrane algorithm with 50 membranes.

We examine compound membrane algorithm with the following parameters:

- Number of membrane algorithms in the first phase is 100.
- All membrane algorithms have 50 membranes.
- Each membrane algorithm in the first phase terminates if the best solution does not improved during 500 iterations[3].
- The membrane algorithm in the second phase terminates if the best solution does not improved during 5000 iterations[3].

Results of computer experiments of compound membrane algorithm are shown in Table 3. We can see that compound membrane algorithm always outputs almost strict solutions.

On a single processor, computation time of compound membrane algorithm, of course, is much longer than that of simple membrane algorithm. But, because membrane algorithms in the first phase work completely independent, compound membrane algorithm will easily be implemented on distributed computing system and computation time will be twice as short as that of simple membrane algorithm.

[3]These numbers are selected according to the feature that membrane algorithm converges fast (Figure 2).

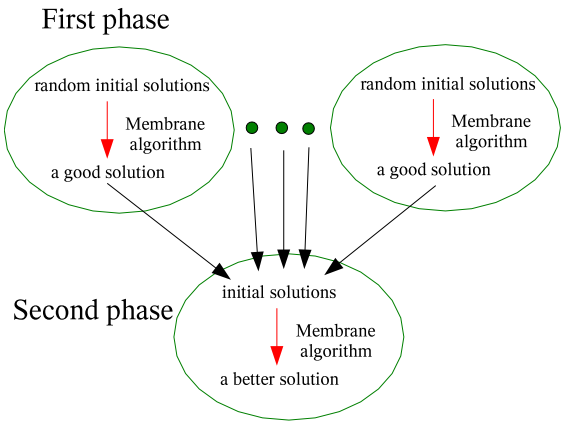

Figure 3: Compound membrane algorithm.

Table 3: Results of compound membrane algorithm. Trials of compound and simple membrane algorithms are 10.

	eil51			kroA100		
	compound	membrane 50	SA	compound	membrane 50	SA
best	429	429	430	21431	21651	21369
average	431	435	438	21616	22590	21763
worst	435	444	445	21816	24531	22564

5 Conclusion

We have proposed and implemented a new algorithm, called membrane algorithm, for solving NP-complete optimization problems. Computer experiments have shown that membrane algorithm gets as good approximate solutions for TSP as simulated annealing algorithm. Convergence of membrane algorithm is fast. An improved membrane algorithm, compound membrane algorithm, always gives almost strict solutions for TSP.

References

[1] C. A. Floudas and P. M. Pardalos (eds), *Encyclopedia of Optimization* (Kluwer, Dordrecht, 2001).

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, (Freeman, 1979).

[3] K. Maekawa *et. al.*, A solution of traveling salesman problem by genetic algorithm (in Japanese), *SICE*, **31**, 598–605, 1995.

[4] Gheorghe Păun, Computing with membrane, *Journal of Computer and System Sciences*, **61**, 108–143, 2000.

[5] Gerhard Reinelt, TSPLIB, URL http://www.iwr.uni-heidelberg.de/group/comopt/software/TSPLIB95/

[6] Arto Salomaa, *Computation and Automata*, (Cambridge University Press, Cambridge, 1985).

[7] M. Yoneda, URL http://www.mikilab.doshisha.ac.jp/dia/research/person/yoneda/research/2002_7_10/SA/07-sareslut.html