A Novel algorithm of Genetic Ant Colony Optimization (GACO) for Traveling Salesman Problem

Zne-Jung Lee*

Abstract

In this paper, a novel algorithm of genetic ant colony optimization (GACO) is proposed for traveling salesman problem (TSP). TSP is to minimize the cost of travel of a salesman in visiting all the cities in a given set, and return to the starting city. Basically, the proposed algorithm combines ant colony optimization (ACO) with genetic algorithm (GA) and can explore and exploit search spaces. It has both the advantage of ACO, the ability to find feasible solutions and to avoid premature convergence, and that of GA, the ability to avoid being trapped in local optima. In this paper, variant test problems are drawn from randomly generated data and the most well known TSP problems chosen from TSPLIB. Simulation results are reported, and the proposed algorithm seems to have admirable performance.

Keywords：Genetic Algorithm, Ant colony optimization, Traveling Salesman Problem.

* Assistant professor, Director of Department of Information Management, Kang-Ning Junior College of Medical Care and Management

應用基因蟻群最佳化方法解旅行者售貨員問題

李 仁 鐘*

摘 要

本文提出應用基因蟻群最佳化方法解旅行者售貨員問題。基本上旅行者售貨員問題乃是給定一個有 N個城市的集合，一個售貨員想要確實地拜訪每一個城市然後再回到開始的城市，以求得最短的距離。本文中提出以結合基因演算法及蟻群最佳化方法來解旅行者售貨員問題，此演算法兼具基因演算法及蟻群最佳化方法優點，其搜尋時兼具廣域及區域搜尋優點，可找到可行解、避免提早收斂、落入區域最佳解中而無法找到廣域最佳解。本文中所測試的旅行者售貨員問題包含經由亂數產生的資料及由TSPLIB中所提供的測試題型。經由模擬結果顯示，本文所提出的方法優於其他演算法則。

關鍵詞：基因演算法、蟻群最佳化演算法、旅行者售貨員問題

*康寧醫護暨管理專科學校 助理教授兼資訊管理科主任

I、Introduction

Traveling salesman problem (TSP) is the optimization problem of finding a shortest closed tour that visits all the given cities. It is known as a classical NP-complete problem, which has extremely large search spaces and is very difficult to solve [1]. Classical methods for solving TSP usually result in exponential computational complexities. Recently, population-based search algorithms maintain a population of structure as key elements in the design and implementation of problem solving algorithms. Each individual in the population is evaluated according to its cost. These algorithms are sufficiently complex to provide powerful adaptive search approaches, and usually can be embedded with other algorithms to speed up their search performances for optimization problems.

Two of the most used population-based algorithms are genetic algorithms and ant colony optimization. Genetic algorithms (GAs) or more generally, evolutionary algorithms [2] have been touted as a class of general-purpose search strategies for optimization problems. GAs use a population of solutions, from which, using crossover, mutation and selection strategies, better and better solutions can be produced. GAs can handle any kind of objective functions and any kind of constraints without much mathematical requirements about the optimization problems, and have been widely used as search algorithms in various applications. Various GAs have been proposed in the literature [3, 4] and shown superior performances over other methods. As a consequence, GAs seemed to be nice approaches for solving TSP. However, GAs may cause certain degeneracy in search

performance if their operators are not carefully designed [4]. Recently, there are many search activities over artificial ants, which are agents with the capability of mimicking the behavior of real ants [5~7]. The agents are sufficiently intelligent to exploit pheromone information that has been left on the traversed ground. Agents can then choose a route according to the amount of pheromone. The larger amount of pheromone is on a route, the larger is the probability of selecting the route by agents. With such concept, a population-based algorithm, Ant Colony Optimization (ACO), has been widely used as a new cooperative search algorithm [5].

In this paper, a novel algorithm of genetic ant colony optimization (GACO) for traveling salesman problem is proposed. In the proposed algorithm, GAs and ACO perform a multiple directional search by maintaining a set of solutions, referred to as population, that provide the capability of escaping from local optima [8]. They can find feasible solutions in the whole search spaces, and conduct fine-tuning in the local spaces. To ameliorate the search performance, we combine both algorithms for solving TSP.

This paper is organized as follows. In section 2, the problem of TSP is introduced. The basics of genetic algorithms and ant colony optimization are introduced in section 3. In section 4, the proposed GACO algorithm is described. In section 5, the proposed algorithm is employed to solve the examples of TSP, and the results are listed. The performance shows the superiority of the proposed algorithm. Finally, section 6 concludes this paper.

II、The Formulation of TSP

The definition of a traveling salesman problem (TSP) is: given N cities, if a salesman starting from his home city is to visit each city exactly once and then return home, find the order of a tour such that the total distance (cost) traveled is minimum [1]. TSP has extremely large search spaces and is very difficult to solve. Exact methods and heuristic (probabilistic) methods can be used to solve TSP. Exact methods, like cutting planes, branch and bound [9], can only optimally solve small sized problems while the heuristic methods, like 2-opt [10], Markov chain [11] and simulated annealing [12] are good for large sized problems.

Genetic algorithms and ant colony optimization can be used to solve large TSP and can get good solutions. The first efforts by using GAs for solving TSP are those of Goldberg using partial mapped crossover [13]. Dorigo also applied ant system to solve TSP, and it is successful for designing an optimization algorithm [5]. In this paper, a novel algorithm of genetic ant colony optimization (GACO) for traveling salesman problem is proposed. We present evidence to show that the performance of the proposed algorithm for solving TSP is better than that of running GAs or ACO.

III、The Basics of GAs and ACO

GAs and ACO algorithms are population-based search algorithms with capable of very wide applications, and they can be hybridized with other algorithms [14,15]. Many researches have been done to hybrid these algorithms to improve the

quality of solutions. It consists in making several algorithms work together to profit from the best characteristics of each of them. The basics of ACO and GA are described in this section.

1、Genetic Algorithms (GAs)

Genetic algorithms start with a set of randomly selected chromosomes as the initial population that encodes a set of possible solutions. In GAs, variables of a problem are represented as genes in a chromosome, and the chromosomes are evaluated according to their costs using some measures of profit or utility that we want to optimize. The recombination typically involves two genetic operators: crossover and mutation. The genetic operators alter the composition of genes to create new chromosomes called offspring. The selection operator is an artificial version of natural selection, a Darwinian survival of the fittest among populations, to create populations from generation to generation, and chromosomes with better cost have higher probabilities of being selected in the next generation. After several generations, GA can converge to the best solution. Let $P(t)$ and $C(t)$ are parents and offspring in generation t. A usual form of general GA is shown in the following [3,8]:

Procedure: General GA

Begin

$t \leftarrow 0$;

Initialize $P(t)$;

Evaluate $P(t)$;

While (not match the termination conditions) do

Recombine *P*(*t*) to yield *C(t)*;

Evaluate *C*(*t*);

Select *P*(*t*+1*)* form *P(t)* and *C(t)*;

$t \leftarrow t+1$;

End;

End;

Recently, genetic algorithms with local search have also been considered as good alternatives for solving optimization problems. The local search for TSP, 2-opt approach, can be implemented after crossover and mutation operators [10].

2、Ant Colony Optimization (ACO)

ACO is a class of algorithms using artificial ants with the capability of mimicking the behavior of real ants [15]. Real ants mark the paths that they walk on by laying down pheromone in a quantity. Other ants can observe the pheromone trail and are attracted to follow it. Because ants choose paths based on the amount of pheromone, the paths will marked again and will attract more ants. The pseudo code of ACO algorithm is stated as [15]:

Procedure: ACO algorithm

Begin

While (ACO has not been stopped) do

Ants_generation_and_activity( );

Pheromone_evaporation( );

Daemon actions( );

End;

End;

In the algorithm, agents find solutions starting from a start node and moving to feasible neighbor nodes in the process of ants_generation_and_activity. During the process, information collected by agents is stored in the so-called pheromone trails. In the process, agents can release pheromone while building the solution (online step-by-step) or while the solution is built (online delayed). An ant-decision rule, made up of the pheromone and heuristic information, governs agents' search toward neighbor nodes stochastically. Pheromone_evaporation is a process of decreasing the intensities of pheromone trails over time. This process is used to avoid locally convergence and to explore more search space. Daemon actions are optional for ACO, and they are often used to collect useful global information by offline depositing additional pheromone. For TSP, 2-opt approach can be also adopted for the daemon action.

IV、The Proposed Algorithm

The proposed algorithm combines ACO and GAs to improve the search ability. The diagram of the proposed algorithm is shown in Figure 1. In the proposed algorithm, the used parameters are defined first. Then, ant starts from a random chosen city. In the process of ants_generation_and_activity, each ant moves based on the ant-decision table. The ant-decision table governs which not visited city j is chosen for ant k from currently located city i, and it is shown as:

$$s = \begin{cases} \arg\{\max_{i=allowed_k(t)} [\tau_{ij}(t) \cdot \eta_{ij}^{\beta}]\} & when(q \le q_0) \\ S & otherwise \end{cases} \quad (1)$$

Where $\tau_{ij}(t)$ is the pheromone trail at time t, $\eta_{ij}=1/d_{ij}$ and d_{ij} is the distance between city i and city j, β is a parameter representing the importance of the heuristic information, q is a random number uniformly distributed in [0,1], q_0 is a pre-specified parameter ($0\leq q_0\leq 1$), $allowed_k(t)$ is the set of feasible cities currently not visited by ant k at time t, and S is an index of a city selected from $allowed_k(t)$ according to the probability distribution given by

$$P_{ij}^{k}(t)=\begin{cases}\dfrac{\tau_{ij}(t)\eta_{ij}^{\beta}}{\sum_{u\in allowed_k(t)}\tau_{ij}(t)\eta_{ij}^{\beta}} & \text{if } s\in allowed_k(t)\\ 0, & \text{otherwise;}\end{cases} \tag{2}$$

In finding feasible solutions, ants online update and evaporate pheromone trails as:

$$\tau_{ij}(t) \leftarrow (1-\psi)\,\tau_{ij}(t)+\psi\tau_k(t), \tag{3}$$

where $0<\psi\leq 1$ is a constant governing the pheromone decay process, $\tau_k(t)=1/C_k$ is the value of the current pheromone trail, and C_k is the cost produced by the k^{th} ant. Once all ants have built solutions, a genetic algorithm is used to improve the generated solutions. When a new global best solution is found, the offline pheromone-updating rule is performed as:

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

where $0<\rho\leq 1$ is a parameter governing the pheromone decay process, $\Delta\tau(t)=1/C_{best}$, and C_{best} is the global best cost gotten from the beginning of the search process. These steps are repeated until stop criterion is satisfied.

For GA, we use a path representation where the cities are listed in the order in which they are visited. For example, assuming there are 5 cities 1, 2, 3, 4, 5, if a salesman goes from city 3, through city 5, city 1, city 4, city 2 and returns back to city 3, the chromosome will be (3 5 1 4 2). The initial population is generated by

randomly placing 1 to *N* into *N* length chromosomes, and it is guaranteed that each city appears exactly once. Noted that the global best solution is also placed into initial population to bias the search direction. The evaluation function for TSP is the sum of distances between every pair of cities in the tour. Basically, the partial mapped crossover (PMX), inversion mutation and (u+λ)–ES (evolution strategy) survival [8,13,16], where u is the population size and λ is the number of offspring created, are employed as the operators for TSP. In our study, we just set maximum generation as the stop criterion for GA.

V、Simulation and Results

We report the computational results obtained on a number of variant test problems in this section. In the following simulations, the default following values are used: the size of the initial population for GAs and ant number for ACO are both set as 50, the crossover probability P_c=0.8, maximum generation=100, the mutation probability P_m=0.1, ρ=0.1, ψ =0.1, q_0=0.8, and β=2. Several test problems are considered to compare the performances of existing algorithms. These algorithms include general GA, GA with 2-opt local search (GA-2-opt), ACO, ACO with 2-opt local search (ACO-2-opt), and the proposed algorithm. Since those algorithms are search algorithms, it is not easy to stop their search in a fair basis from the algorithm itself. In our comparison, we simply stopped these algorithms after a fixed time of running. Experiments were run on PCs with Pentium 4 processor, and were stopped after two hours of running.

First, the test problems are randomly generated. For those problems, the simulation results are listed in Table 1. Results are averaged over 10 trials. Noted that the best solution is chosen from all of the simulation results. The percentage error is calculated as: % Error = [(averaged solution - best solution)/ best solution] x100. From Table 1, algorithms with local search can perform better performance than those algorithms without it. Furthermore, the proposed algorithm does find the best solutions for all test problems, but other existing algorithms conduct not so well. Finally, the test problems taken from TSPLIB [17] with known optimal solution are used in this paper. For those problems, the simulation results are listed in Table 2. Results are also averaged over 10 trials. Since optimal solutions are known for these problems, the percentage error is calculated as: % Error = [(averaged solution - optimal solution)/ optimal solution] x100. From Table 2, the proposed algorithm does find the optimal solutions, and it outperforms other existing algorithms.

VI、Conclusions

In this paper, we presented a novel algorithm of genetic ant colony optimization (GACO) for traveling salesman problem. It keeps the advantages of ant colony optimization and GAs. From our simulation for those test problems, the proposed algorithm indeed can find the best solutions or optimal solutions. In other words, the proposed algorithm seems to have admirable performance.

Reference

S. J. Louis and L. Gong, " Case injected genetic algorithms for traveling salesman problems," *Information Sciences*, vol. 122, pp. 201-225, 2000.

T. Bäck, U. Hammel, and H. P. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE Trans. On Evolutionary Computation,* vol. 1, no. 1, pp. 3-17, 1997.

M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design,* John Wiley & Sons Inc., 1997.

L. Jiao and L. Wang, "Novel genetic algorithm based on immunity," *IEEE Transactions on Systems, Man and Cybernetics, Part A,* vol. 30, no. 5, pp. 552 –561, 2000.

M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. On Evolutionary Computation*, vol. 1 pp. 53-66, 1997.

R. Beckers, J. L. Deneubourg, and S. Goss, "Trails and u-turns in the selection of the shortest path by the ant Lasius Niger," *Journal of Theoretical Biology*, vol. 159, pp. 397-415, 1992.

S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels, "Self-organized shortcuts in the argentine ant," *Naturwissenchaften,* vol. 76, pp.579-581, 1989.

Zne-Jung Lee, Shun-Feng Su, and Chou-Yuan Lee, "Efficiently Solving General Weapon-Target Assignment Problem by Genetic Algorithms with Greedy Eugenics," IEEE Transactions on Systems, Man and Cybernetics, Part B, pp. 113-121, 2003.

M. Padberg and R. Rinaldi, "Optimization of a 532-city symmetric travelling salesman problem by branch and cut," *Operations Research Letters,* vol. 6, no.1, pp. 1-7, 1987.

S. Lin and B. Kernighan, "An effective heuristic algorithm for the travelling-salesan problem," *Operations Research*, vol. 21, no. 2, pp. 498-516, 1973.

O. Martin, S. Otto and E. Felten, "Large-step markov chains for the traveling salesman problem," *Complex Systems,* vol. 5, no. 3, pp. 299-326, 1991.

S. Kirkpatrick, C. D. Gelatt and M. P. Vechi, "Optimization by simulated annealing," *Science,* vol. 220, no.4598, pp. 671-680, 1983.

D. Goldberg, R. Lingle and Alleles, "loci and the travelling saleman problem," *Proceedings of the Second International Conference on Genetic Algorithms*, Lawerence Eribaum Associates, Mahwah, NJ, 1985.

A. Kolen and E. Pesch, "Genetic local search in combinatorial optimization," *Discrete Applied Mathematics and Combinatorial Operation Research and Computer Science*, vol. 48, pp. 273-284, 1994.

E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence From Natural to Artificial Systems*, Oxford University Press, 1999.

Zne-Jung Lee, Shun-Feng Su, and Chou-Yuan Lee, "A Genetic Algorithm with Domain Knowledge for Weapon-Target Assignment Problems," Journal of the Chinese Institute of Engineers, vol. 25 no. 3, pp. 287-295, 2002.

G. Reinelt, TSPLIB, University of Heidelberg, 1996. http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html,

Table 1. Simulation results of randomized data for various search algorithms. The Bold-faced text indicates the best one among these algorithms.

Problem	Best solution	% Error				
		ACO-2-opt	ACO	GA-2-opt	GA	The proposed algorithm
N=50	566	0	0	0	0.053%	**0**
N=80	725	0.017%	0.024%	0.097%	0.21%	**0**
N=450	1886	0.113%	0.204%	0.32%	0.91%	**0**
N=700	3636	0.317%	0.438%	0.84%	1.439%	**0**

Table 2. Simulation results of TSPLIB problems for various search algorithms. The Bold-faced text indicates the best one among these algorithms.

Problem	Optimum solution	% Error				
		ACO-2-opt	ACO	GA-2-opt	GA	The proposed algorithm
Ry48P (*N*=48)	14422	0	0	0	0.03%	**0**
Ft70 (*N*=70)	38573	0.014%	0.02%	0.08%	0.11%	**0**
Att532 (*N*=532)	27686	0.13%	0.21%	0.37%	0.84%	**0**
Rat783 (*N*=783)	8806	0.37%	0.48%	0.74%	1.29%	**0**

Figure 1. The diagram of the proposed algorithm