An Improved Ant Colony Optimization Algorithm with Crossover Operator

Junen Guo* and Wenguang Diao

Luoyang Institute of Science and Technology, Luoyang, 471023, China

Abstract: Ant colony algorithm has been widely applied to lots of fields, such as combinatorial optimization, function optimization, system identification, network routing, robot path planning, data mining and large-scale integrated circuit design of integrated wiring, etc. And it achieved good results. But it still has one weak point which is the slowing convergence speed. To aim at the lacks, an improved ACO is presented. This paper studies a kind of improved ant colony algorithm with crossover operator which makes crossover operator among better results at the end of each iteration. The experiment results indicate that the improved ACO is effectual.

Keywords: Ant colony optimization, combinatorial optimization, convergence speed, crossover operator, genetic algorithm.

1. INTRODUCTION

Research over the recent years has shown that the ACO has a powerful capacity to find out solutions to combinatorial optimization problems, has the advantage of distributed computing, and is easy to accommodate with other algorithms, thus displaying powerful robustness. Besides, it has displayed a high flexibility and robustness in dynamic environments [1].

Reference [2] used an Ant Colony Optimization algorithm to achieve a satisfactory solution in a reasonable computation time. A variety of simulation experiments were run to choose ACO parameter values and to demonstrate the performance of the proposed method. The simulation results showed that the proposed ACO algorithm is superior to the common Apparent Tardiness Cost-Batched Apparent Tardiness Cost rule for minimizing the TWT and make span. The arrival time distribution and the number of jobs strongly affected the ACO algorithm's performance.

Reference [3] considers the dynamic TSP (DTSP), where cities are replaced by new ones during the execution of the algorithm. Under such environments, traditional ACO algorithms face a serious challenge: once they converge, they cannot adapt efficiently to environmental changes. To improve the performance of ACO on the DTSP, a M-ACO algorithm is used to solve the DTSP.

Reference [4] presents a proposal for the efficient solution to one of the most frequently requested services on social networks; namely, taking different types of relationships into account in order to locate a particular member of the network. The solution is based on a biologically-inspired modification of the ant colony optimization algorithm.

Reference [5] addresses the Surgical Case Assignment Problem with an objective of minimizing the total unexploited and operating cost. A two-stage ant colony optimization (ACO) algorithm is introduced and the results show that ACO outperformed the other algorithms and reached better solutions in a faster computational time.

Reference [6] presents a solution methodology using ant colony optimization (ACO) for a distribution– allocation problem in a two-stage supply chain with fixed cost for a transportation route. Taguchi method for robust design is adopted for finding the optimum combination of parameters of the ACO algorithm.

A fast two-stage ACO algorithm is proposed in reference [7], which overcomes the inherent problems of traditional ACO algorithms. The basic idea is to split the heuristic search into two stages: preprocess stage and path planning stage. In the preprocess stage, the scent information is broadcasted to the whole map and then ants do path planning under the direction of scent information. The algorithm is tested in maps of various complexities and compared with different algorithms. The results show that the proposed algorithm has the good performance and convergence speed.

All these studies have contributed to the improvement of the ACO to some extent, but they have little obvious effect on increasing the convergence speed. The deficiency remains to the bottleneck constraining the ACO from being widely applied in large-scale optimization problems. For this end, on the basis of the solution to the Traveling Salesman Problem (TSP), we studied a kind of improved ant colony algorithm with crossover operator (COACO). The experiment results show that the algorithm proposed in this study can substantially increase the convergence speed of the ACO.

2. THE ANT COLONY OPTIMIZATION ALGORITHM (ACO)

The ACO simulates the behavioral feature of ants which will spontaneously find the shortest path from the colony to

*Address correspondence to this author at the Luoyang Institute of Science and Technology, Luoyang, 471023, China; Tel: 0086-379-65929100; Fax: 0086-379-65929100; E-mail: lit_gjn@yeah.net

the food source in their foraging process. In addition to this simulation, people also improve this feature and apply it to many different fields. Ants select the path by way of the pheromone density. On the various paths from the colony to the food source, originally not a bit of pheromone is left on them. When the first batch of ants select a path for the first time, they do not know which path is the shortest, and therefore their selection is always blind. Subsequently, other batches of ants will use the pheromone left by the first batch of ants as a kind of reference information and select the desired path according to the pheromone density. For those ants that carry the food back to their colony, when they head for the food source once again, they will not travel along the same route but will reselect a new path. In this case, they will select the path according to the pheromone density left before by the ants. In the process of traveling, the ants will select their path depending solely on the pheromone density, and therefore the pheromone density on the path whose pheromone density is dense will be increasingly denser, and more and more ants will follow this path until almost all the ants follow this path, which is called the shorter path.

M. Dorigo *et al.* put forward the ant system algorithm (AS) and the ant colony system algorithm (ACS) successively which simulate the behavior of ants in their foraging process. These scholars defined the AS and the ACS as the ant colony optimization algorithm (ACO).

2.1. The Basic Ant Colony System Model (AS)

Now we take the TSP (travelling salesman problem) in n cities as an example to illustrate the ant colony system model [8]. Suppose m is the quantity of the ants in the colony, d_{ij} (i,j=1,2,…,n) stands for the distance from the city i and city j, and $b_i(t)$ stands for the quantity of ants at the time of t in city i, then we will have $m=\sum_{i=1}^{n} b_i(t)$.

$\tau_{ij}(t)$ stands for the leftover pheromone amount on the path from city i and city j at the time of t. At the beginning, the amounts of pheromone on various paths are equal. Suppose $\tau_{ij}(0)=C$ (C is a constant). In the traveling process, ant k (k=1,2,…,m) will determine its moving direction according to the pheromone amount left on each path. $p_{ij}^k(t)$ stands for the probability of ant k's traveling from city i to city j at the time of t:

$$p_{ij}^k=\begin{cases}\dfrac{[\tau_{ij}(t)]^\alpha[\eta_{ij}]^\beta}{\sum_{k\notin tabu_k}[\tau_{ik}(t)]^\alpha[\eta_{ik}]^\beta}, & j\notin tabu_k\\ 0, & j\in tabu_k\end{cases} \tag{1}$$

In this equation: η_{ij} is the heuristics information when ant travels from city i to city j, and generally $\eta_{ij}=1/d_{ij}$; d_{ij} is the distance from city i to city j; α is the degree of importance of the leftover information on the path from city i to city j; β is the degree of importance of the heuristics information; $tabu_k$ (k=1,2,…,m) is used to record the city which ant k is currently traveling through, which is called tabu table, i.e., the next city which is not allowed to choose; the set $tabu_k$ will be adjusted dynamically in the evolution process.

After n moments, all the ants have completed a tour, and their tabu table for the present tour will be full and at this moment the tabu table should be emptied. Meanwhile, the city which the ant is currently in should be set $tabu_k$, preparing for the next tour. At this moment, compute the path L_k that each ant has covered and retain the shortest path $L_{k_{\min}}$.

$$L_{k_{\min}}=\min\{L_k\}\,(k=1,2,...,m) \tag{2}$$

With the passage of time, the information left before will gradually fade away and parameter $1-\rho$ is used to stand for the information eclipse degree. After the ant has completed a cycle, the pheromone amount on each path will be adjusted according to equation (3):

$$\tau_{ij}(t+n)=(1-\rho)\tau_{ij}(t)+\rho\Delta\tau_{ij} \tag{3}$$

In this equation:

$$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^k \tag{4}$$

$$\Delta\tau_{ij}^k=\begin{cases}\frac{Q}{L_k}, When\,the\,kth\,ant\,passes\,city\,i\,and\,j\\ 0, Others\end{cases} \tag{5}$$

$\Delta\tau_{ij}^k$ represents the amount of the pheromone left by the kth ant when it passes the path from city i to city j, $\Delta\tau_{ij}$ represents the total amount of the pheromone left by all the ants on the path from city i to city j in one complete loop.

When the preset loop number NC has been reached, the algorithm will terminate its execution. The shortest path in NC loops will be the shortest path found out by the algorithm.

2.2. The ACS Algorithm

The ACS is mainly different from the AS in three aspects:

1) The move rules of the ants are different;

2) The global updating rules are different;

3) Local updating rules which adjust the amount of the pheromone on various paths are newly added.

Now we will describe the ACS algorithm.

Step 1: Initiation. The amount of the pheromone on each side is initiated into a tiny constant value; allocate m ants randomly to n cities, and meanwhile set the starting city into the tabu table.

Step 2: Each ant will choose the next city according to equation (6) and modify the tabu table.

$$j = \begin{cases} \arg\max\limits_{j \notin tabu_k} \{[\tau_{ij}(t)][\eta_{ij}(t)]^{\beta}\}, if\ q \le q_0 \\ S, otherwise \end{cases} \tag{6}$$

$q_0 \in [0,1]$ is an initially set parameter; q is a random number and $q \in [0,1]$; S is a random variable determined in accordance with equation (1). This strategy obviously increases the variety of any searching, thus avoiding any premature falling into the local best result and getting bogged down.

Step 3: Local updating of pheromone. After each ant has chosen a city, the amount of pheromone on each side will be updated according to equation (7).

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}^{k} \tag{7}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{l_{jb}}, When\, the\, ant\, travels\, through\, city\, i\, and\, j \\ 0, Otherwise \end{cases} \tag{8}$$

l_{jb} is the path length that ant k has covered from the starting city to the present city. All the other parameters are identical with those of equation (3).

Step 4: Computing of the optimal path. After m ants have travelled through all the cities, compute the length of the optimal path according to equation (9).

$$l_{\min} = \min\{l_k\}, (k = 1, 2, \cdots, m) \tag{9}$$

l_k is the path length that the kth ant has covered in its travelling.

Step 5: Global updating of pheromone. After all the ants have travelled through all the cities, update only the amount of the pheromone on the optimal path according to equation (10).

$$\tau_{ij}^{new} = (1-\alpha)\tau_{ij}^{old} + \alpha\Delta\tau_{ij} \tag{10}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{l_k}, If\, the\, global\, best\, result\, is\, through\, path \\ 0, Otherwise \end{cases} \tag{11}$$

In equation (10), α is the volatilization quotiety of the global pheromone and l_k is the length of the optimal path.

Step 6: If the designated search number is not attained, then empty the tabu table and repeat the above steps.

3. THE COACO ALGORITHM

3.1. Rationale of the COACO Algorithm

3.1.1. Even Distribution Strategy of the Initial Ant Colony

In the ACS algorithm, m ants are put onto n cities in the initiation, and thus some cities may have many ants while some cities may have no ant at all. However, in the ACS algorithm, it is according to equation (6) that each ant chooses the next city. Because the amount of pheromone on each path is initially identical, therefore the ant mainly uses the distance between the two cities as the heuristic factor when it chooses the next city. In this way, when there are relatively more ants in a certain city, the density of the pheromone on a certain path will be strengthened due to the relatively larger number of ants travelling along the path. However, the path is not necessarily the shortest path.

In order to solve this problem, we adopted a method to distribute the ants evenly, i.e., distribute m ants to n cities and make sure that each city receives at least one ant (suppose m≥n). Thus, the search space of the solution is enlarged and the probability of getting the best result is increased.

3.1.2. Nearest Neighbor Node Choosing Strategy

In the ACS algorithm, when the ant chooses the next city, the probability of its transfer from city i to city j is computed, and then the city whose transfer probability is larger will be chosen to travel along, while the distance between the two cities is used as the heuristic factor in the probability computation. Therefore, the computation result of next city j is possibly not the shortest city from city i. When there are many cities, the computation work of transfer probability from city i to other cities is rather awesome, the computation speed is very slow, and therefore the convergence of the algorithm is exceptionally slow. Consequently, we designed the nearest neighbor node choosing rules and in each computation attempt we computed only the probabilities of those cities that are consistent with the rules, thus substantially reducing the time needed for computation and increasing the convergence speed of the algorithm.

The nearest neighbor node choosing rules are as follows: First of all, a coordinate system is established according to the coordinate files of the n cities so as to fix the positions of the n cities, and the distances of every two cities are computed, thus forming an $n \times n$ distance matrix A; then, sort the distances in each column from short to long, and then replace the sorted city distances with the cities' serial numbers; finally, delete the last line (because we presuppose that the distance from city i to city i is infinity, namely unreachable, and therefore the distance between them is at the bottom after sorting), thus forming a $(n-1) \times n$ distance index matrix B. Thus, the ith column from the first line to the (n-1)th line is the numbering arrangement of the cities from near to faraway from city i.

In the COACO algorithm, we designate that only the probabilities of d cities nearer to city i will be computed. Thus, when there are many cities, if we designate a smaller d value, then the convergence speed of the algorithm will be enhanced and the precision of the algorithm can be increased.

3.1.3. Crossover Operator Choosing Strategy

Because the ACS algorithm is easy to fall into the local best result and cause the phenomenon of being bogged down, we designed a strategy to conduct crossover operation of better results, thus solving the problem of rapidly converging the algorithm into a certain result, diversifying the space of results, and increasing the possibility of seeking for the best result.

The crossover operator is mainly used for the evolution of algorithm. This study adopts the greedy crossover operator to conduct crossover operation.

In the present algorithm, first of all, at the end of each round of cycle, the length of the path covered by each ant is computed, and the probability of the kth ant's being chosen as the father generation will be computed in accordance with equation (12). Then, the probability of each ant's being chosen as the father generation will be computed, and then two of them will be chosen by way of roulette to be used as the two father generations of the crossover operator so as to realize the crossover operation. Finally, the path lengths of the two newly born generations will be computed, and if the result is better than the optimal result of the round concerned, then the present result will be used to replace the optimal result of the round concerned, and equation (10) will be used to update the amount of the pheromone on the path; if the result is not better than the optimal result of the round concerned, then the processing of the descendants will be abandoned.

$$p^k_{parent} = \frac{\frac{1}{L_k}}{\sum_{i=1}^{m}\frac{1}{L_i}}, (k = 1, 2, \cdots, m) \qquad (12)$$

3.2. The COACO Algorithm

Now we will describe the COACO algorithm proposed in our study.

Step 1: Initiation. According to the coordinate files of the n cities, we set up a nearest node matrix B, designate the nearest nodes to be queried as d (d≤n), set the counter to NC=MAX, set the initial amount of pheromone on each path as a constant, distribute m ants evenly into n cities and guarantee that each city has at least one ant, and set the city to which the ant is allocated into the tabu table of the ant.

Step 2: For each ant, with the present city i as the center, according to the nearest node choosing rules, we will sort out the d cities that ants have not travelled through and compute their transfer probability, and if the number of the cities that have not be travelled through is less than d, then all will be computed; after that, we will select a city j whose transfer probability is larger to conduct the transfer. In the d cities, the node j will be selected according to equation (13).

$$\begin{cases} j = \arg\max_{j \notin tabu_k} \{[\tau_{ij}(t)][\eta_{ij}(t)]^{\beta}\}, if\ q \le q_0 \\ p^k_{ij} = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{k \notin tabu_k}[\tau_{ik}(t)]^{\alpha}[\eta_{ik}]^{\beta}}, j \notin tabu_k, Otherwise \end{cases} \qquad (13)$$

In equation (13), q is a random number (0<q≤1); when $q > q_0$, the transfer probability p^k_{ij} of d cities will be computed, the city j will be selected according to the roulette rules, and city j will be added into this ant's tabu table $tabu_k$.

Step 3: Local updating of pheromone. This step is identical with that of the local updating of pheromone in the ACS algorithm.

Step 4: Computing of the optimal path. When m ants have finished travelling through all the cities, we will compute the length of the path that each ant has covered, find out the optimal path among the paths that have been covered by these m ants, and save its travelling path and its length of the path.

Step 5: Global updating of pheromone. When all the ants have travelled through all the cities, first in accordance with equation (10) we will update the pheromone on the optimal path found out in Step 4, and then we will conduct the crossover operation and update the optimal path length and the global pheromone.

Step 6: If the designated search number is not attained, then empty the tabu table and repeat the above steps.

4. COMPARISON OF THE EXPERIMENT RESULTS

In order to verify the validity of the algorithm proposed in this study, we downloaded the well-known TSP example eil76 from TSPLIB, conducted experiments, and compared our results with those of the ACS algorithm and those of the algorithm proposed in reference [9] in the two aspects of algorithm convergence and experiment results. Because the parameter selection has no existing theory to guide, we had to determine the necessary parameters by way of experiments. The parameters set by the present algorithm are as follows: in equation (1), α=1, β=4, ρ=0.6, Q=20000, q_0=0.5, d=15; in equation (10), α=0.5.

4.1. Experimental Comparison of the Convergence Trait

Fig. (**1**) shows the convergence comparison when we obtained the optimal result by conducting ten rounds of experiments to solve the eil76 TSP problem by means of different methods. In Fig. (**1**), the horizontal coordinate stands for the algorithm's iteration number of times, and the vertical coordinate stands for the obtained optimal path length. The solid line stands for the algorithm's convergence trait, the dot-and-dash line stands for the ACS algorithm's

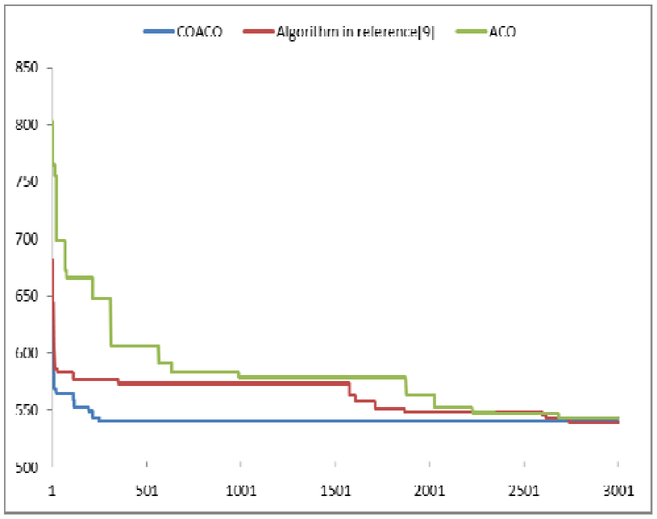

Fig. (1). Comparison of the convergence trait of three ACO's in solving the eil76 TSP problem.

convergence trait, and the continuous dashed line stands for the convergence trait of the algorithm in reference [9]. Our experiments show that the present algorithm can indeed increase the convergence speed of the algorithm.

4.2. Comparison of the Results of Optimizing the TSP Problem

Table **1** presents the comparison of the better results obtained from solving the eil76 TSP problem. From Table **1** it can be seen that although the optimal result obtained from using the present algorithm is short of the currently published optimal result, it can been seen from Fig. (**1**) that its convergence speed is obviously much faster than those of the other two algorithms.

Table 1 Comparison of the better results of the eil76 TSP problem.

Algorithm	Result
ACS	543.578023
Algorithm in reference [9]	539.741359
Algorithm in proposed in the present study	540.48414
Result published by website [10]	538

CONCLUSION

Our experiments have shown that the method employed by the present study is effective to increase the convergence speed of the algorithm. However, the ACO algorithm is not like the genetic algorithm which has a solid mathematical basis and a systematic analyzing methodology. Besides, the parameters involved in our experiments have no theory to guide and the research findings obtained so far are mostly based on experiments. Nevertheless, we can believe that the ACO algorithm, like any other intelligent algorithm, will be applied more and more widely in more fields and gradually form its own theoretical basis.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

Declared none.

REFERENCES

[1] Q. B. Zhu, and Z. J. Yang, "An ant colony optimization algorithm based on mutation and dynamic pheromone updating", *J. Soft.*, vol. 15, Issue 2, pp.185-192, 2004.

[2] L. Li, F. Qiao, and Q.D. Wu, "ACO-based multi-objective scheduling of parallel batch processing machines with advanced process control constraints", *Int. J. Adv. Manuf. Technol.*, vol. 44, pp. 985-994, 2009.

[3] M. Mavrovouniotis, and S. Yang, "A memetic ant colony optimization algorithm for the dynamic travelling salesman problem", *Soft Comput.*, vol. 15, pp.1405-1425, 2011.

[4] J. Rivero, D. Cuadra, J. Calle and P. Isasi, "Using the ACO algorithm for path searches in social networks", *Appl. Intell.*, vol.36, pp. 899-917, 2012.

[5] C. Rizk and J. Arnaout, "ACO for the surgical cases assignment problem", *J. Med. Syst.*, vol. 36, pp.1891-1899, 2012.

[6] V.P. Vinay and R. Sridharan, "Taguchi method for parameter design in ACO algorithm for distribution–allocation in a two-stage supply chain", *Int. J. Adv. Manuf. Technol.*, vol. 64, pp.1333-1343, 2013.

[7] X. Chen, Y. Kong, and X. Fang, "A fast two-stage ACO algorithm for robotic path planning", *Neural Comput. & Applic.*, vol. 22, pp. 313-319, 2013.

[8] J.E. Guo, S.T. Wang and H.L. Xu, "Amino acid sequence alignment algorithm based on ant colony optimization genetic algorithm", *Comput. Applic.*, vol. 27, Issue 6, pp. 1434-1437, 2007.

[9] M. Dorigo, and G. D. Caro, "Ant colony optimization: A new meta-heuristic", In: *Proc. of the 1999 Congress on Evolutionary Computation*, vol 2, Washington: IEEE Press, pp.1470-1477, 1999.

[10] http://elib.zib.de/pub/Packages/mp-testdata/tsp/tsplib/tsp/index.html

Received: November 11, 2013 Revised: February 11, 2014 Accepted: March 3, 2014