# An Improved Ant Colony Optimization Algorithm Based on Pheromone Backtracking

Zhiguo Liu[1, 2], Tao Liu[1, 2], Xiue Gao[1, 2]
[1] Key Laboratory of Communications Network and Information Processing, Dalian University, China
[2]School of Information Engineering, Dalian University, China
gaoxiue510@126.com

***Abstract*—In this paper, backtracking algorithm is adopted to the pheromone updating rule to resolve the basic Ant Colony Optimization (ACO) algorithm's shortcoming of easily falling into local optima. When the pheromone accumulated to the backtracking point on the tour, pheromone will be backtracked in the improved algorithm. The improved algorithm not only solves the ACO algorithm in excessive accumulation of pheromone problems, but also has better global search ability and convergence speed, which increase the quality of the solution space by using the information of the previous iterations' ants. Finally, the improved algorithm is applied to the Traveling Salesman Problem(TSP), and the simulation results show that it is much better than basic ACO algorithm in many aspects, such as the optimal iterations, the average and the optimal solution etc.**

***Keywords-ant colony algorithm; pheromone; backtracking***

## I. INTRODUCTION

Ant Colony Optimization (ACO) algorithm[1] is firstly proposed by Marco Dorigo et al in 1991 year for solving the combinatorial optimization problem, such as Traveling Salesman Problem (TSP). ACO algorithm not only can searches intelligently and optimize globally, but also has the features of good robustness, positive feedback, distributed computation, and easily combining with other algorithms etc. Therefore, the advent of ACO algorithm provides a powerful tool for many fields to solve complex combinatorial optimization problem. In recent 20 years, ACO algorithm for solving optimization problem has developed continuously, but there still exist the shortcomings of consuming time, easy to stagnation and easily falling into local optima[2].

In the ACO algorithm, pheromone characterizes the carrier of the past information, directly affecting the ACO algorithm's global convergence and solving efficiency[3]. Thus, many scholars have done a lot of researches on it. Besides ant colony system(ACS)[4], Reference [5] proposed a dynamic transition aiming at the disadvantage of stagnation behavioral ant colony algorithm; reference [6] proposed KCC-Ants and ELU-Ants algorithm, using a new local pheromone updating rule; The order optimization was introduced into the initialization of pheromone in reference [7], and presented an ordinal updating strategy of pheromone; reference [8] proposed a adaptive ant system and LMS algorithm was introduced to update the pheromone; reference [9] presented a new pheromone updating strategy that only updates the pheromone of important tours. Although previous algorithms improved the performance of ACO algorithm through pheromone, they don't address the ACO algorithm's problem of pheromone's excessive accumulation, the algorithm would be trapped into local optima finally.

Backtracking algorithm[10] is a selecting optimization search method, which searching forwards according to selected optimal conditions with the purpose of getting the target. But when it explores to a certain step and the original choice is not optima or fails to meet goals, then step back to reselect. This paper applies the backtracking algorithm to the ACO algorithm to reduce search space and solves the problem of excessive accumulation of pheromone, which increasing the convergence speed and global searching ability.

## II. BASIC ACO ALGORITHM

Basic ACO algorithm is proposed for solving TSP, so the following model that solves the n-city TSP is used to introduce the basic ACO algorithm. TSP is an NP-hard combinatorial optimization problem in operations research and theoretical computer science. Given a list of cities and their pairwise distances, the task is to find a shortest possible tour that visits each city only once. Now let the coordinates of n cities be $C=\{c_1,c_2,...,c_n\}$, the number of ants be $m$, the total amount of residual pheromone at time $t$ in the edge connection of $(i, j)$ is $\tau_{ij}(t)$. At the initial moment, the amount of pheromone on each tour is equal to a constant $\tau_{ij}(0)=const$. Firstly the $m$ ants are putted randomly into $n$ cities. Under common circumstances, the more the number of ants is, the stronger the global search ability of the optimization is. But as the increase of the number of ants, the convergence time of the optimization will rise at an exponential rate. Then the ants select the next city in accordance with the pheromone on the edge and so on. During the walking process, the probability formula that *k-th* $(k=1, 2, ..., m)$ ant selects the next city is:

$$P^k_{ij}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}(t)]^{\beta}}{\sum_{Z\subset allowed_k}[\tau_{iz}(t)]^{\alpha}\cdot[\eta_{iz}(t)]^{\beta}} & j\not\subset tabu_k\\ 0 & otherwise\end{cases} \tag{1}$$

Where $P^{k}{}_{ij}(t)$ is the probability that *k-th* ant select the edge $(i, j)$ to move on at time *t*; $\alpha$ is pheromone parameter to show the importance of pheromone; $\eta_{ij}(t)$ is heuristic function, which equals the reciprocal of the distance between the *i-th* city and the *j-th* city, namely $\eta_{ij}(t) = 1/d_{ij}$, characterizing the weight of the edge $(i, j)$; $\beta$ is heuristic pheromone parameter, expressing the importance of heuristic information; $tabu_k\ (k = 1, 2, ..., m)$ is the taboo table for *k-th* ant, recording the city that the *k-th* ant has going through; $allowed_k = \{C - tabu_k\}$ represents the next city *k-th* is allowed to select at time *t*.

When all of the ants complete their tours on their own iteration, the pheromone on the edge which the ants have walked needs updating. First, portion of the original pheromone is evaporated. Then the pheromone of edges which ants have gone through is increased. Pheromone updated formula is as follows:

$$\tau_{ij}(t + \Delta t) = (1 - \rho)\,\tau_{ij}(t) + \tau_{ij}(\Delta t) \tag{2}$$

$$\tau_{ij}(\Delta t) = \sum_{k=1}^{m} \tau_{ij}^{k}(\Delta t) \tag{3}$$

Where $\rho\ (0 \le \rho < 1)$ is the pheromone decay parameter, $(1 - \rho)$ is the pheromone residual parameter; $\tau_{ij}(\Delta t)$ is the pheromone increment on the edge $(i, j)$ in this iteration, $\tau_{ij}(\Delta t) = 0$ initially. According to the different pheromone update strategy, Marco Dorigo produces three different algorithm model, namely ANT-density model, ANT-quantity model and ANT-cycle model. The difference between them is the numeration of $\tau_{ij}(\Delta t)$ The three models are expressed as equation (4) to (6) respectively.

ANT-density model:

$$\tau_{ij}^{k}(\Delta t) = Q \tag{4}$$

Where $Q_1$ is a constant, which represents the amount of pheromone the ant release going through each edge, increment of the amount of pheromone on edge $(i, j)$ mainly depends on the number of ants ever going through.

ANT-quantity model:

$$\tau_{ij}^{k}(\Delta t) = Q_2/d_{ij} \quad k = (1, 2, \cdots, m) \tag{5}$$

Where $Q_2$ is a constant, which represents the amount of pheromone the ant release after going through the edge $(i, j)$; $d_{ij}$ is the length of edge $(i, j)$.

ANT-cycle model:

$$\tau_{ij}^{k}(\Delta t) = Q_3/L_k \quad k = (1, 2, \cdots, m) \tag{6}$$

Where $Q_3$ is a constant, which represents the whole amount of pheromone the ant release after going through all the cities; $L_k$ is the distance that *k-th* ant has completed the tour of all the cities, the increment of pheromone on edge $(i, j)$ depends on the number of ants through the path and the length edge $(i, j)$.

The performance of ANT-cycle model is the best, so ANT-cycle model is used to be the basic ACO algorithm.

After pheromone updating is complete, set the taboo table *Tabu* to Null, and continue the above steps until the optimal solution is found or the maximum iterations are reached.

## III. IMPROVED ALGORITHM

Recently, during the iterative process of the basic ACO algorithm, it has a large difference between the concentrations of pheromone on edges, which lead ants to select certain edges, consequently reducing the global search capability. To resolve this problem, backtracking algorithm is adopted to the pheromone updating rule to get a better control of the ant colony pheromone gap.

### A. Pheromone Backtracking Algorithm

Backtracking algorithm is a search algorithm with both systematic and jumping, when the optimal value is not found, it will shield the current search space and re-search to narrow the search space to increase the search efficiency. Backtracking is generally applied to solve those with a large number of potential solutions, but numbers of preferential solutions have been examined.

Pheromone accumulates with the increase of the iterations through the ACO algorithm. It's very easy to accumulate continuously when a local optimal tour appears, leading to the pheromone much larger on this tour than others. During this time, algorithm has been trapped into a local optimum. According to the above problems, the backtracking algorithm is introduced to the colony in this paper, backtrack the pheromone, the amount will backtrack to the initial value when we found the algorithm get into the local optima. The detailed implementations are as follows:

When the algorithm completes initialization, it starts iteration. Each iteration of the optimal value was recorded as $Best(NC)$, $NC$ is the iterations of the algorithm. We update pheromone on the optimal path at this time as (7) shows, when the optimal value does not change for N times in each backtracking period.

$$\tau_{ij(t+\Delta t)} = \tau_{ij(t)} - N \bullet Q / L\_best(NC) \tag{7}$$

Where $L\_best(NC)$ is the length of current best tour when the number of iterations is $NC$; $Q$ (equals to $Q_3$) is the whole amount of pheromone the ant released after going through all the cities. When the optimal value does not changing for $M$ times, it gets back to the backtracking point, and re-initialized the pheromone as (8) shows.

$$\tau_{ij}(0) = \begin{cases} \tau_0 / N & (i, j) \subset R\_best \\ \tau_0 & otherwise \end{cases} \quad (8)$$

In equation (8), $R\_best$ is the current best tour. Such pheromone updating rule can get a quick research based on a known local optimal solution. Make use of pheromone backtracking not only prevents excessive accumulation, but also ensures a faster search speed.

### B. Pheromone Updating Rule

Pheromone updating rules in the improved algorithm using global update rules and local update rules, the local update rule adopt basic ACO algorithm are (2) and (6). Global pheromone updating rule adopt (9) to improve the convergence speed.

$$\tau_{ij}(t + \Delta t) = \tau_{ij}(t) + Q / L\_best \quad (9)$$

Where $L\_best$ is the length of current optimal tour. The pheromone updating rule can decrease bad solutions, reduce the search scope of the solution space and increase the quality of the solution space, accordingly find the optimal solution quickly, and improve the performance of the algorithm.

### C. Description of the Improved Algorithm

*1)* Given the city coordinates $C$, initializing parameters $\alpha$ $\beta$ $\rho$ $Q$, $\tau_{ij}(0)$ and the maximum iterations $NC\_max$.

*2)* Put the m ants randomly into the $n$ cities, and put the city into the ants' taboo list $Tabu$.

*3)* Ants select the next city according to the (1), and then put the selected cities to their $Tabu$ lists correspondingly to complete the n-city tour.

*4)* Calculate the shortest length of tours in current iteration and all iteration when the ants complete one iteration. If $NC \geq NC\_max$, then jump to *8)*, or jump to *5)*.

*5)* The pheromone is updated by applying the local updating rule defined by (2) and (6), applying the global updating rule defined by (9).

*6)* If the global optimum hasn't changed by $N$ times, update the pheromone by applying the updating rule defined by (7), if the global optimum hasn't changed by $M$ $(M > N)$ times, update the pheromone according to (8).

*7)* Set the taboo table $Tabu$ to Null, $NC = NC + 1$, jump to *2)*.

*8)* Output the optimal solution.

## IV. APPLY IMPROVED ALGORITHM TO TSP

We apply the novel algorithm to the 30-city TSP of Oliver30, and compare the novel algorithm with the basic algorithm, ACS and the improved algorithm of reference[5].

The running simulation environment: Intel Core (TM) 2 Quad 2.67GHz processor; 4G RAM; 6M L2 cache; 1333MHz front-side bus; Windows XP operating system; MATLAB 7.1 IDE. Set parameters: $\alpha = 1$, $\beta = 5$, $\rho = 0.1$, $Q = 100$, $\tau_{ij}(0) = 1$, $N = 10$, $M = 30$, $NC\_max = 300$. The results are shown in Table 1, Fig.1 and Fig.2.

TABLE I. COMPARING DIFFERENT ALGORITHMS IN OLIVER30 PROBLEM

| *Performance* | *Basic* | *Improved* | *ACS*[5] | *Reference[5]* |
|---|---|---|---|---|
| *Best* | 426.54 | 423.74 | 423.74 | 423.74 |
| *Worst* | 434.62 | 425.82 | 434.63 | 425.27 |
| *Average* | 430.92 | 424.86 | 426.74 | 424.19 |
| *Standard deviation* | 7.1790 | 1.74 | | |
| *Optimal iterations* | 80 | 50 | 1259 | 521 |

The values in Table 1 are obtained with the average of 20 simulations. As can be seen from table 1, the novel ACO algorithm is better than the basic one in performance of optimal solution, optimal iterations, and standard deviation and so on. Compared with the ACS and algorithm from Reference[5], the optimal iterations of novel algorithm is much less than the other 2 algorithms'.

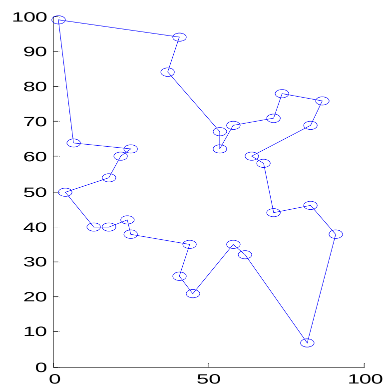

Figure 1. Optimal path.

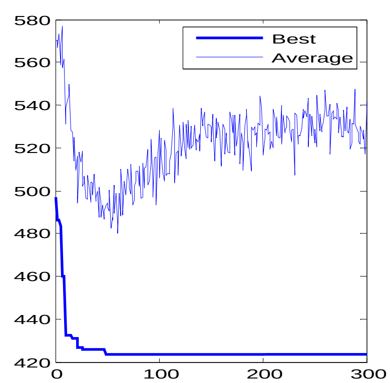


Figure 2. Evolutionary curve

Fig.1 and Fig.2 shows the algorithm of the optimal path and the evolution of curves, and we can see that the optimal tour length decreases rapidly to the current optimal value of 423.74 in the 50th iteration. Meanwhile, the average tour length declines rapidly following the optimal tour length. When the algorithm finds the optimal solution, the average tour length increases, this is the result of algorithm's global expansion search.

In summary, the novel algorithm can be seen in less iteration to find optimal solutions, and avoid falling into local optima.

## V. CONCLUSION

This paper proposes a new method to improve the ACO algorithm-pheromone backtracking. Using this method to update the pheromone, it can not only achieve the optimal solution rapidly by present information, but also can expand to global search through the pheromone backtracking when the algorithm falling into a local optimum. The simulation on TSP of Oliver30 shows that the proposed algorithm is superior.

## REFERENCES


[1] A. Colorni, M. Dorigo and V. Maniezzo et al, "Distributed optimization by ant colonies," Proceedings of the 1st European Conference on Artificial Life, 1991, pp.134-142.

[2] Guo Shao-sheng and Meng You-xin, "An improved entropy-based ant colony optimization algorithm," 2010 International Conference on Computer Application and System Modeling, Proceedings, vol.15, 2010, pp.1548-1550.

[3] M. Daniel and M. Martin, "Ant colony optimization with global pheromone evaluation for scheduling a single machine," Applied Intelligence, vol.18, 2003, pp.105-111.

[4] M. Dorigo, L M Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," IEEE Transactions on Evolutionary Computation, vol.1, no.1, 1997, pp.53-66.

[5] Zheng Song, Hou Di-bo, Zhou Ze-kui, "Ant colony algorithm with dynamic transition probability," Control and Decision, vol.23, no.2, 2008, pp.225-228.

[6] Naimi and Hossein Miar, "New robust and efficient ant colony algorithms: using new interpretation of local updating process," Expert Systems with Applications, no.36, 2009, pp. 481-488.

[7] Zhang Zhao-jun, Feng Zu-ren and Ren Zhi-gang, "Novel Ant Colony Optimization Algorithm Based on Order Optimization," Journal of Xi'An Jiaotong University, vol.44, no.2, 2010, pp.15-19; 30.

[8] A. Paul and S. Mukhopadhyay, "An Adaptive Pheromone Updation of the Ant-System using LMS Technique," International Conference on Modeling, Optimization, and Computing, vol.1298, 2010, pp.498-503.

[9] Bi Ying-zhou, Zou Peng, Zhong Zhi and Ding Li-xin, "A novel pheromone update with important solution components," 2010 6th International Conference on Natural Computation, vol.5, 2010, pp.2551-2555.

[10] Wang Li and Li Dong, "Special factor backtracking algorithm for optimizing," 2010 International Conference on Intelligent Computing and Integrated Systems, 2010, pp.27-30.