International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.7 (2014), pp.395-402
http://dx.doi.org/10.14257/ijmue.2014.9.7.33

Application of Improved Ant Colony Algorithm in Solving TSP

Dan Liu, Lijuan Zheng and Jianmin Wang

School of Information Science and Technology
Shijiazhuang Tiedao University
Shijiazhuang 050043, China
E-mail: liudanld@126.com

Abstract

Using ant colony algorithm to solve TSP (traveling salesman problem) has some disadvantages as easily plunging into local minimum, slow convergence speed and so on. In order to find the optimal path accurately and rapidly, an improved ant colony algorithm is proposed. Experimental results show that the improved ant colony algorithm has better effectiveness for TSP problems solutions.

Keywords: *Ant Colony Algorithm, TSP, Optimal Path*

1. Introduction

TSP (Traveling salesman problem) was firstly put forward by an Irish mathematician William Roman Hamilton at the beginning of the 19th century. It is usually described like this: with urban traffic network preset, how to select a path for promoters and make sure the total journey, which starts from a certain point (station), passes each node and returns back to the starting point, is the shortest [1]. TSP is one of the world-known non-deterministic polynomial problems and one of the classical issues in the circle of combination optimization and computer science [3].

Since TSP is an abstract form of complex engineering optimization in various fields, it has been extensively applied in such fields as VLSI chip design, network route and automobile route selection, and has provided different algorithms in discrete optimization with a platform full of methods and thoughts [2, 4]. In addition, as a typical problem in NPC, TSP and other NPC issues are equivalent. If breakthroughs are made in the solution of TSP, a large number of NPC issues can be readily solved. That's why studies on TSP solutions have great practical value and theoretical significance. Although the absolutely effective algorithm of TSP hasn't been found yet, human beings have been continuously exploring and accumulating numerous algorithms for a long time. The existing main algorithms can be divided into traditional optimization algorithm and modern intelligent optimization algorithm. The former can be mainly divided into two directions, namely complete algorithm and approximation algorithm [6]. The latter mainly include genetic algorithm, simulated annealing algorithm, ant colony algorithm, taboo search algorithm, greedy algorithm and neural network algorithm.

Ant colony algorithm is a kind of heuristic algorithm put forward by M. Dorigo to solve the famous travelling salesman problem and also a kind of intellectual evolutionary algorithm simulating ant colony's foraging behavior in the Nature. In the searching process of ant colony algorithm, ants firstly search the surrounding area of their nest randomly. Once an ant finds food source, it will evaluate the number and quality of food, and carry some food to the nest; it will also leave a kind of secretion that is regarded as pheromone on the ground on its way back. The number of such pheromone it leaves depends on the amount and quality of food [10-15]. As to certain path, the larger the number of ants selecting this path is, the more

ISSN: 1975-0080 IJMUE

intensive pheromone left by ants along this path becomes. In this way, more ants will be attracted to select this path, thus forming a kind of favorable positive feedback. Through positive feedback mechanism, ants can finally find the shortest paths from food to their nest one by one [8-9].

It is shown from the above that the mode ants cooperate in foraging is, in essence, a. the thicker pheromone trace in the path is, the larger probability it will be chosen, also known as path probability selection mechanism; b. the shorter the path is, the faster pheromone trace in it grows, also known as pheromone update mechanism; c. ants communicate via pheromone, also known as collaboration mechanism. It is the exact feature of the living creature-ant colony that has been applied in ant colony algorithm for solution. Since the process in which ants forage is extremely similar with the solution of TSP, the earliest application of ant colony algorithm was on TSP solution [7,16-19]. However, there was a larger possibility of being stuck in locally optimal solution, which means after the research reaches certain time period or degree in which the results acquired from all individuals are almost the same, which blocks the way of further exploring better results and ends up with failure in finding the globally optimal solution [5, 20-23]. Therefore, an improved ant colony algorithm which can solve the problem is put forward in this dissertation.

2. Traditional Ant Colony Algorithm in Solving TSP

2.1. Quantization of Ants' Behaviors in Solving TSP with Ant Colony Algorithm

Take n as the scale of TSP, here referred to as the number of cities, m as the number of artificial ants, τ_{ij} as pheromone concentration of edge (i, j) at t moment. After ants finish one cycle, pheromone concentration on the corresponding moment edge is formula (1), which means the addition of residual pheromone concentration on the last time range and the newly added pheromone concentration on the current time range. Take ρ as a range of constant from 0 to 1. Apparently, 1-ρ means the volatilization intensity factor of pheromone concentration from t to t+1,

$$\tau_{ij}(t+1) = \rho\tau_{ij} + \Delta\tau_{ij} \qquad (1)$$

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \qquad (2)$$

Here, (2) is the added pheromone concentration at (i j) from t to t+1.

As to the three computation models of $\Delta\tau_{ij}$ suggested by M. Dorigo: ant-cycle system、ant-quantity system and ant-density system, the latter two models utilize local information while the former one utilizes global information. According to experiments and comparisons, ant-cycle system shows better performance in TSP solution. So, it is taken as the basic model in this dissertation.

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k}, \text{if the kth ant passes edge(i,j) between (t,t+1)} \\ 0, \text{ else} \end{cases} \qquad (3)$$

Here, Q is a constant used to show the total quantity of released pheromone after the ant finishes a complete path search. L_k shows the total path cost for the kth ant, and it is the aggregation of all costs kth ant need to pass each path. Apparently, ant will not release any pheromone in the path it doesn't pass.

Define the kth ant at point V_i at the present moment. Next, choose point V_j, which means the probability of selection path (i, j) is (4):

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{k\tau_{is}^{\alpha}\eta_{is}^{\beta}(t)}, s\in allowed, j\in allowed_k \\ 0, otherwise\end{cases} \quad (4)$$

$$\eta_{ij}=\frac{1}{C_{ij}} \quad (5)$$

C_{ij} is the cost ant needs in the path (i, j). α and β are used to control the relative importance degree of pheromone concentration and the length of path. $allowed_k$ is the collection of optional paths in the next step on the premise that kth ant meets the three conditions of artificial ants' behaviors. The detailed procedure of Ant Colony Algorithm is shown in Figure 1.

2.2 Ant Colony Algorithm Description in Solving TSP

Based on the above analysis, data structure visited-cityk which controls the feature of "not selecting paths already taken in the present cycle as the next step" is introduced, and it represents a collection of cities the kth ant visits, $1\le k\le m$. At the initialized stage, m artificial ants are placed in different cities, and the initial pheromone concentration of each edge is set as a constant $\tau_{ij}(0)=C$. The first value of element visited-cityk of each ant is set as the city it starts. After ants finish a complete path searching (starting off from a certain city, passing all cities once and returning to the starting point), calculate the value of $\Delta\tau_{ij}^{k}$ and update the pheromone concentration on each edge. Then, starts a new cycle till iterations reaches the preset maximum number of cycles-NC or all ants select the same path. That's the end of the process.

Algorithm of Ant-Cycle System is described as below:

(1) Initialization: $t=0, nc=0, \Delta\tau_{ij}=0$, place m ants in n cities, nc represents the number of iterative cycle; t represents a "fake" time control variable.

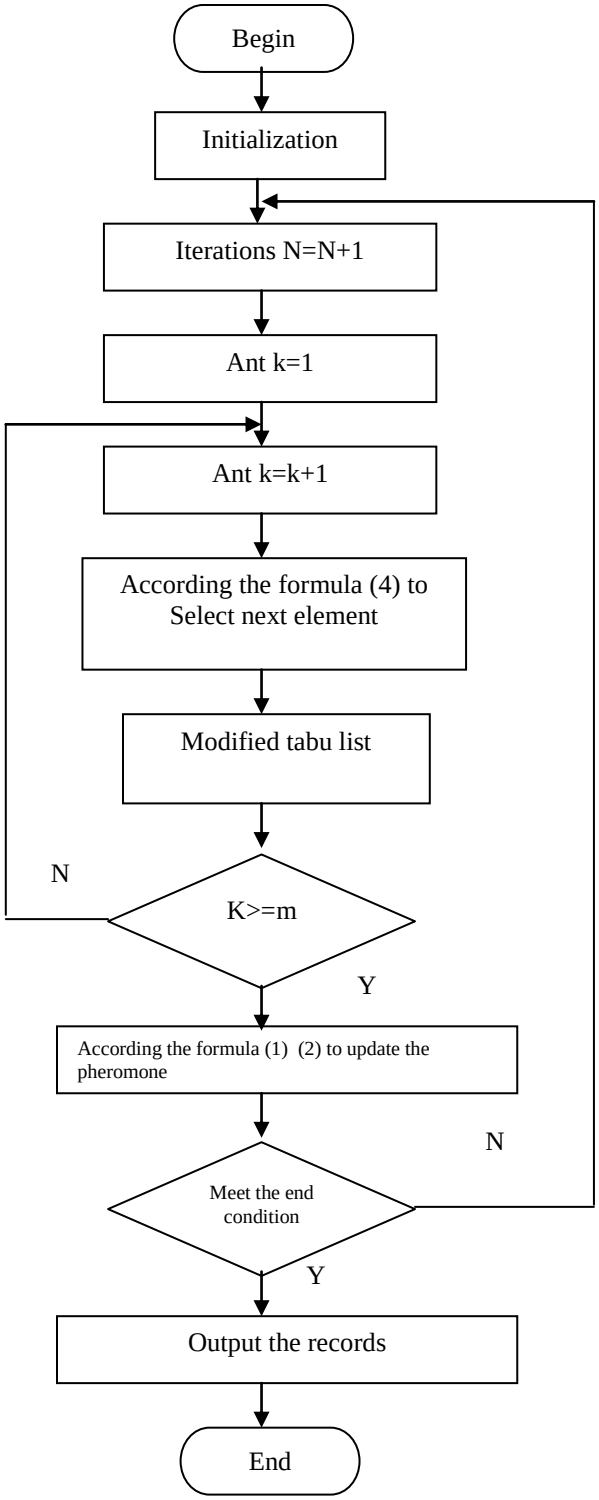

Figure 1. The Detailed Procedure of Ant Colony Algorithm

(2) Place the initial starting points of all ants in the current disaggregation visited-cityk; move each ant k to the next point j at probability p_{ij}^{k} and place j in the current disaggregation visited-cityk.

(3) Calculate the objective function value L_k of each ant; record the current best solution.

(4) Update the pheromone concentration of each edge based on rules of adjusting pheromone.

(5) Set $\Delta\tau_{ij} = 0, nc = nc + 1$ on each edge (i, j).

(6) If nc $<$ NC and there's no degeneration (all solutions found are identical), proceed with step (2). Otherwise, finish the process and output the current best solution.

3. Improved Ant Colony Algorithm in Solving TSP

3.1. Improved Ant Colony Algorithm Strategies

The difference between improved ant colony algorithm and basic ant colony algorithm lies in the addition of disturbance strategy, volatilization factor adjustment strategy and rewards and punishments strategy. In order to improve the global searching ability of the algorithm and avoid being stuck in local optimum, disturbance strategy is put forward here, because the basic ant colony algorithm will be easily stuck in local optimum. When the iterations of this cycle is the integral multiple(s) of the number of cities, start disturbance of pheromone concentration and decrease higher values of pheromone concentration, so as to prevent the algorithm from being stuck in local optimum. This dissertation targets at the first 20% path with the highest pheromone concentration. The specific method goes like this: adjust the pheromone concentration of the first 20% path with the highest pheromone concentration to an average value while keeping the remaining pheromone concentration. In this way, research can be continued based on the existing better value in the next iteration, thus avoiding local optimum. The value of pheromone volatilization coefficient ρ is directly related with the global searching ability and rate of convergence of ant colony algorithm. The expression (6) below shows the update treatment this algorithm applies on residual pheromone.

$$\tau_{ij}(t) = \tau_{ij}(t-1) \times (1-\rho) \qquad (6)$$

Here, ρ is pheromone volatilization velocity coefficient. The method it is computed also depends on computation models. In order to avoid being stuck in local optimum, the smaller volatilization coefficient is applied at the initial stage of searching. And the bigger volatilization coefficient is applied at the later stage to improve the rate of convergence. In this algorithm, dynamic setting of ρ is applied. Based on different scales of cities, ρ is smaller at the initial stage of iteration, which can expand the range of searching and avoid being stuck in local optimum. At the later stage of iteration, ρ value gradually increases, which makes it possible to achieve faster rate of convergence. Pick up better solutions found in each cycle and offer rewards and worse solutions for punishment, so as to accelerate the differences of pheromone between better paths and common paths. The specific method goes like this: offer extra pheromone rewards to paths ants pass and achieve optimal values and less pheromone rewards to paths ants pass and achieve approximate optimal values, and decrease pheromone in paths ants pass and achieve worse values.

3.2. Improved Ant Colony Algorithm Flow

At the initial stage, acquire a city distance table from city information and place all ants in the starting cities. Set pheromone concentration on each edge as a constant. After ants finish a complete path selection, calculate the value of τ_{ij}^{k} and update the pheromone concentration on each edge. Then, start a new cycle of iteration. When iterations reaches the preset constant, put an end to the process.

Basic steps of ant colony algorithm for TSP are shown below:

(1) Initialization: finish initialization and assignment of corresponding parameters;

(2) Read data of all cities and calculate the distance between two cities. Initialize pheromone concentration on each edge;

(3) Before starting a new cycle of iteration, initialize taboo table of all ants and rules of all ants' state transition probability. Select the next destination city under the restriction of taboo table and update the current taboo table till a legal path is finally formed;

(4) Pick up better solutions found in the cycles and offer rewards, otherwise, punishments to worse solutions;

(5) Dynamically set volatilization coefficient ρ based on iterations and city scales;

(6) If iterations in the cycle is integral multiple (s) of the number of cities, activate disturbance strategy and update stimulating information sheet;

(7) After iteration in each cycle finishes, calculate the length of path generated by all ants. The length of path is the aggregation of all edges ants pass. Here, the shortest path is recorded. Update pheromone on each edge based on the rewards strategy for the shortest path. Update volatilization information on the global stimulating information table;

(8) If the value found in this cycle of iteration is better than the previous optimal value, set it as the current optimal value and output it;

(9) If conditions for termination are not reached, start a new cycle of iteration; otherwise, put an end to the process and output the optimal value.

4. Experiment Result

Take Oliver 30 in the internationally universal TSP test library TSPLIB as an example and compare the improved algorithm put forward in this dissertation, the basic ant colony algorithm and ant colony system algorithm to examine the performance of algorithms. Programming language is C++. Compiling environment is Microsoft Visual C++ 6.0. In this experiment, parameter settings are $\alpha=1$,$\beta=3$, $Q=100$, respectively. Test each algorithm for 40 times. The result of tests is shown in Table 1 below.

5. Conclusion

In conclusion, bringing in disturbance strategy and dynamic volatilization factor adjusting strategy can effectively avoid being stuck in local optimum too early; adopting rewards strategy can enhance the efficiency of searching. The improved algorithm in this dissertation can substantially improve the rate of convergence and stability when comparing with traditional any colony algorithm. Thus, it is demonstrated to be feasible enough here.

Table 1. The Result of Experiment

Algorithm	Basic Ant Colony Algorithm	Improved Ant Colony Algorithm

Average Iterations	140	78
Average Solution	430.238	428.334
Optimal Solution	425.82	423.741
Total Time of 40 Tests	46.648	11.687

Acknowledgements

The authors are extremely grateful to the anonymous reviewers for their constructive and valuable comments, which have contributed a lot to the improved presentation of this paper. The work was partially supported by scientific research project of Hebei education department (no. Z2012151).

References

[1] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem", IEEE Transaction on Evolutionary Computation**, (1997)**, pp. 53-66.
[2] M. Dorigo, V. Maniezzo and A. Coloni, "The Ant system: Optimization by a colony of cooperating agents", IEEE Trans Syst Man, Cybernetics- Part B, vol. 26, no. 1, **(1996)**, pp. 29-41.
[3] B. Bullnheimer, R. F. Hartl and C. Strauss, "A new rank based version of the ant system-a computational study", Central European Journal for Operations Research and Economics, vol. 7, **(1999)**, pp. 25-38.
[4] M. Dorigo and L. M. Gambardella, "Ant colonies for the traveling salesman problem", BioSystems, vol. 43, **(1997)**, pp. 73-81.
[5] G. Gutin and D. Karapetyan, "A memetic algorithm for the generalized traveling salesman problem", Natural Computing, vol. 9, **(2010),** pp. 47-60.
[6] B. Bontoux, C. Artigues and D. Feillet, "A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem", Computers and Operations Research, vol. 37, no. 12, **(2010),** pp. 1844-1852.
[7] M. Albayrak and N. Allahverdi, "Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms", Expert Systems with Applications, vol. 38, no. 12, **(2011),** pp. 1313-1320.
[8] M. Bhattacharyya and A. K. Bandyopadhyay, "Comparative study of some solution methods for traveling salesman problem using genetic algorithms", Cybernetics and Systems, vol. 40, no. 12, **(2009),** pp. 1-24.
[9] T. A. S. Masutti and L. N. Castro de, "A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem", Information Sciences, vol. 179, **(2009),** pp. 1454-1468.
[10] Wang, Yun, Huang, Dechai, Yu, Youhong, "Quantum computation and quantum algorithm research", Zhejiang University of Technology, Institute of Computer, Application of computer system, vol. 20, no. 6, **(2011),** pp. 228-231.
[11] Y. W. Leung and Y. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization", IEEE Transactions on Evolutionary Computation, vol. 5, no. 1, **(2010),** pp. 41-53.
[12] Fu, Jiaqi, Ye, Chunming, "Application study on hybrid quantum algorithm in traveling salesman problem", Journal of University of Shanghai for Science and Technology, vol. 31, no. 2, (**2009),** pp. 160-164.
[13] Zhen, Wei, Fang, Zhenge, Yang, Lu, Li, Xiangli, Yi, Xianyang, "Chaotic ant swarm for the traveling salesman problem ", Nonlinear Dyn, vol. 65, **(2011),** pp. 271-281.
[14] X. Shi, Y. Liang, H. P. Lee, C. Lu and Q. Wang, "Particle swarm optimization-based algorithms for TSP and generalized TSP", Inf. Process. Lett., vol. 103, **(2007),** pp. 169-176.
[15] T. Hendtlass, "Preserving diversity in particle swarm optimization", In:Lecture Notes in Computer Science, vol. 2718, **(2003),** pp. 4104-4108, Springer, Berlin.
[16] A. Huang, K. Qin and P. N. Suganthan, "Self-adaptivedifferential evolution algorithm for constrained real-parameteroptimization", in Proc. IEEE Congr. Evol Comput., **(2006),** pp. 17-24.
[17] Brest, S. Greiner, B. Boskovic, M. Mernik and V. Zumer, "Self adapting control parameters in differential evolution, A comparative study on numerical benchmark problems", IEEE Trans. Evol. Comput., vol .10, no. 6, **(2006),** pp. 64-65.

[18] Zhang, Jingqiao and A. C. Sanderson, "JADE: Adaptive Differential Evolution with Optional External Archive", Evolutionary Computation, IEEE Transactions, vol. 4, **(2009),** pp. 945 – 958.
[19] B. Bontoux, C. Artigues and D. Feillet, "A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem", Computers and Operations Research, vol. 37, **(2010),** pp. 1844-1852.
[20] L. Snyder and M. Daskin, "A random-key genetic algorithm for the generalized traveling salesman problem", European Journal of Operational Research, vol. 1, no. 74, **(2006),** pp. 38-53.
[21] M. Dorigo and L. M. Gambardella, "A study of some properties of Ant- Q", Proceedings of PPSN IV-Fourth International Conference on Parallel Problem Solving From Nature, Berlin, Germany, Berlin:Springer - Verlag, **(1996)**, pp. 656-665.
[22] E. Lawer, J. Lenstra and K. A. Ronnooy, *et a1*. "The traveling salesman problem", New York: Wiley-International Publication, **(1985)**.
[23] K. Price, R. Storn and J. Lampinen, "Differential Evolution: A Practical Approach to Global Optimization", Berlin: Springer-Verlag, **(2005)**.
[24] L. M. Gambardella and M. Dorigo, "Ant-Q:A reinforcement learning approach to the traveling salesman problem", Tahoe City, Morgan Kaufmann: Proceedings of ML-95, Twelfth International Conference on Machine Learning**, (1995)**, pp. 252-260.
[25] C. M. Qi, "An ant colony system hybridized with randomized algorithm for TSP", Proc. of the 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, **(2007)**.

Authors

Dan Liu, She received her M.Sc. in education technology from Shijiazhuang Tiedao University, in 2011. Currently, she is a lecturer at Shijiazhuang Tiedao University. Her interests are in software engineering.

Lijuan Zheng, She received her M.Sc in computer application from North China Electronic Power University, Baoding, China, in 2003 and PhD in information security from Beijing Jiao Tong University, Beijing, China, in 2014. Her research interests include security protocol analysis and design, trusted computing.

Jianmin Wang, He received his M.Sc. degree from Yanshan University, China, in 2008. He is currently a associate professor at School of Information Science and Technology Shijiazhuang Tiedao University, China. Her research interests include soft engineering

