A Novel Ant Colony Algorithm with Mutation Operations of Genetic Algorithm for TSP

Bencan Gong[1,a], Peng Chen[1,b]

[1]College of Computer and Information Technology, China Three Gorges University, Yichang, China

[a]gongbc@ctgu.edu.cn, [b]chenpeng@ctgu.edu.cn

Keywords: Ant colony algorithm; Mutation operations; Genetic algorithm; TSP

Abstract. Ant colony algorithm (ACA) is a very effective way for traveling salesman problem (TSP) due to the positive feedback and the distributed parallel computing mechanism. But it has many drawbacks, for example, long search time, precocity and stagnation. In order to speed up convergence rate and ensure the global search ability, a novel ant colony algorithm was proposed. After every round of search, it uses mutation operations of genetic algorithm to optimize tours and quicken the convergence by comparing path length before and after variation. Experiments show that the improved ant colony algorithm can overcome prematurity and obtain the higher solution accuracy than ACA.

Introduction

Traveling Salesman Problem is a classical combinatorial optimization problem and is a NP-hard problem. When *n* cities and its coordinate positions are given, it needs to get the shortest Hamilton circuit, where each city is passed once and only once. TSP has been widely applied to lots of fields, such as large-scale integrated circuit wiring, VLSI chip design, network routing, robot path planning, etc. But the exact solution can't be found for large TSP in polynomial time, because the problem space will increase exponentially with the growth in the number of cities. Many researchers use a variety of heuristic algorithms to solve the problem, for example, ant colony algorithm [1], genetic algorithms (GA), simulated annealing (SA), tabu search (TS), neural networks (NN), particle swarm optimization (PSO), immune algorithm (IA), and so on.

In these algorithms, ant colony algorithm has already obtained promising effect on TSP due to its high efficiency, robustness, positive feedback, distributed computing, simple and easy to combine with other algorithms; but it also has some drawbacks, such as slow convergence speed, easy to falls into local optimal solution. So researchers proposed many improved ant colony algorithms. Ref. [2] proposed an ant colony optimization algorithm of adaptive premium penalty. It was based on the fact that a better path had much better sections and a worse path had much worse sections, and it carried out discriminative premium penalty pheromone renewal strategy to increase the speed of pheromone release by comparing the path sections of the best and worst ants. The better sections would have more pheromone and the worse sections would have less pheromone. The discriminative pheromone release could guide the search of the optimal solution, reduce the search of useless areas, and accelerate the convergence of solutions, thus the search efficiency was enhanced.

Ref. [3] proposed the novel method of updating the whole and local pheromone to avoid early stagnancy. It used the constraint satisfaction techniques to solve the problems of slow convergence by reducing the search space, accelerating search rate and increasing efficiency. Ref. [4] proposed a method based on uniform design to choose appropriate parameters of genetic algorithm and ant colony algorithm. Ref. [5] proposed an improved ant colony algorithm. It combined with the idea of iterative local search algorithm to strengthen the local optimal solution and improve the accuracy of the ant colony algorithm. Each ant would select the next node as short as possible by the 2-opt neighborhood search algorithm. Ref. [6] used adaptive adjustment mechanism to adjust pheromone evaporation factor, and mutated solutions according to the characteristics of ant colony searching for path in each round of iteration. An improved ant colony algorithm is proposed in this paper. Ref. [7]

studied a kind of improved ant colony algorithm with crossover operator among better results after each round of iteration. Ref. [8] proposed an improved ant colony algorithm based on natural selection. It employed evolution strategy of survival the fittest in natural selection to enhance pheromones in paths whose random evolution factor was bigger than threshold of evolution drift factor in each process of iteration. Ref. [9] proposed a novel ACO algorithm based on particle swarm optimization (PSO) algorithm. The new pheromone update method is presented, which combines the global asynchronous feature and elitist strategy. Moreover, the iteration number of ACO algorithm invoked by PSO algorithm is reduced significantly by large amounts of statistical experiments. Ref. [10] proposed an improved Quantum Ant Colony Algorithm based on Bloch coordinates by combining Quantum Evolutionary Algorithm with Ant Colony Algorithm. In this algorithm, the current position information of ants is represented by the Bloch spherical coordinates. Position update, position variation and random behavior of ants are all achieved with quantum rotation gate.

Basic ant colony algorithm

State transition rule. In node *i*, ant *k* chooses next node *j* by the state transition rule in Eq. 1

$$P^k(i,j) = \begin{cases} \dfrac{[\tau(i,j)]^\alpha \cdot [\eta(i,j)]^\beta}{\sum_{s \notin tabu_k} [\tau(i,s)]^\alpha \cdot [\eta(i,s)]^\beta}, & if\ j \notin tabu_k \\ 0, & otherwise \end{cases}. \tag{1}$$

Where α is the relative importance of the pheromone; β is the relative importance of heuristic information; $\tau(i,j)$ is the residual pheromone concentration on edge *(i, j)*; $\eta(i,j) = 1/d(i,j)$ is the heuristic information; $d(i,j)$ is the length of edge *(i, j)*; $tabu_k$ is the set of the next cities that ant *k* could visit.

Pheromone updating rule. After each ant has visited all *n* cities, pheromone is updated on all edges according to Eq. 2.

$$\begin{aligned} &\tau(i,j) = \rho \cdot \tau(i,j) + \sum_{k=1}^{m} \Delta\tau^k(i,j) \\ &where\quad \Delta\tau^k(i,j) = \begin{cases} Q / L^k & if\ edge(i,j) \in path\ done\ by\ ant\ k \\ 0 & otherwise \end{cases} \end{aligned}. \tag{2}$$

Where ρ *(0< ρ <1)* is the reserved proportion of pheromone on edge *(i, j)*; $\Delta\tau(i,j)$ is the increment of pheromone; L^k is the length of the path by ant *k*; *Q* is a constant; *m* is the number of ants.

Improved ant colony algorithm

The single list is used to represent the paths found by ants, where each node includes *city* number and *next* pointer in Fig. 1.

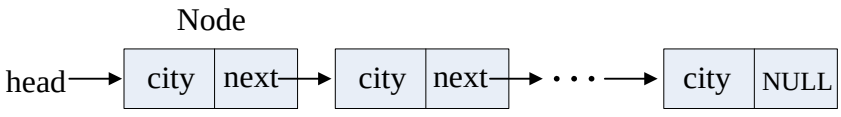

Fig. 1 Representation of the path

The pseudo-code of mutation operations is shown in Fig. 2.

```
Node *P1,*P2,*P3,*P4,*P5;
P1=Head;
P2=P1->next; P3=P2->next;
while(P3->next->next!=NULL)
        P4=P3->next; P5=P4->next;
        while(P5!=NULL)
                d1=d(P2->city,P3->city)+d(P4->city,P5->city);     //d is the distance
                d2=d(P2->city,P4->city)+d(P3->city,P5->city);
                if(d1>d2)
                        P2->next=P4; P3->next=P5; P4=P3;          //exchange nodes
                End if
                P4=P4->next; P5=P5->next;
        End while
        P4=Head; P5=P4->next;
        while(P5!=P2)
                d1=d(P1->city,P2->city)+d(P4->city,P5->city);
                d2=d(P1->city,P4->city)+d(P2->city,P5->city);
                if(d1>d2)
                        P4->next=P1; P5->next=P2; P5=P1;
                End if
                P4=P4->next; P5=P5->next;
        End while
        P1=P1->next; P2=P1->next; P3=P2->next;
End while
P1=Head;
P2=P1->next; P3=P2->next;
while(P3->next->next!=NULL)
        P4=P3->next; P5=P4->next;
        while(P5!=NULL)
                d1=d(P2->city,P3->city)+d(P4->city,P5->city);
                d2=d(P2->city,P4->city)+d(P3->city,P5->city);
                if(d1>d2)
                        P2->next=P4; P3->next=P5; P4=P3;
                End if
                P4=P4->next; P5=P5->next;
        End while
        P4=Head; P5=P4->next;
        while(P5!=P2)
                d1=d(P1->city,P2->city)+d(P4->city,P5->city);
                d2=d(P1->city,P4->city)+d(P2->city,P5->city);
                if(d1>d2)
                        P4->next=P1; P5->next=P2; P5=P1;
                End if
                P4=P4->next; P5=P5->next;
        End while
        P1=P1->next; P2=P1->next; P3=P2->next;
End while
```

Fig. 2 Pseudo-code of mutation operations

Simulation

Experimental parameters are set as follows: m=50, α=1, β=3, ρ=0.9, τ_0=0.1, NC=1000, n=100 and 318, where NC is the number of iterations, m is the number of ants, and n is the number of cities. Experimental results show that improved ant colony algorithm is significantly better than ant colony algorithm in Table 1.

Table 1 Results of experiments

Problem name	Kroa100	Lin318
Known optimal value	21282	42029
Optimal value of improved ant colony algorithm	21282	42091
Optimal value of ant colony algorithm	21389	48416

Conclusions

In order to overcome the slow convergence of ant colony algorithm and easy falling into local optimal solution, we use mutation operations of genetic algorithm to improve ant colony algorithm after each round of iteration. Simulation results show that the improved ant colony algorithm has a significant effect for solving TSP.

Acknowledgements

This work was financially supported by the National Natural Science Foundation of China (61272236).

References

[1] M. Dorigo, L. M., Gambardella: IEEE Transactions on Evolutionary Computation, Vol. 1 (1997) No.1, p. 53.
[2] Li Xinchao, He Qianhua, Li Yanxiong, Li Changbin and Wang Zhingfeng: International Conference on Machine Learning and Cybernetics, Vol. 1 (2014), p. 463.
[3] Luo Yabo, Zhang Shikun and Feng Zhang: Lecture Notes in Computer Science, Vol. 8351 (2014), p. 432.
[4] Dong Wenyong and Dong Xueshi: Lecture Notes in Computer Science, Vol. 8588 (2014), p. 184.
[5] Pang Huanli, Li Yonghe and Song Xianmin: International Conference on Information Technology and Computer Application Engineering (2013), p. 41.
[6] Shen Xuan-Jing and Shen Xuan-Jing: Journal of Chemical and Pharmaceutical Research, Vol. 6 (2014) No.8, p. 508.
[7] Guo Junen and Diao Wenguang: Open Mechanical Engineering Journal, Vol. 8 (2014) No.1, p. 96.
[8] Wu Hua-Feng, Chen Xin-Qiang, Mao Qi-Huang, Zhang Qian-Nan and Zhang Shou-Chun: Journal on Communications, Vol. 34 (2013) No.4, p. 165.
[9] Li Qing, Zhang Chao, Chen Peng and Yin Yi-Xin: Control and Decision, Vol. 28 (2013) No.6, p. 873.
[10] Chen Xiaofeng, Xia Xingyou and Yu Ruiyun: Journal of Computers (Finland), Vol. 8 (2013) No.6, p. 1536.