Approximate Backbone Subsection Optimization Algorithm for the Traveling Salesman Problem

Feipeng Wang[1](✉), Hu Peng[1], Changshou Deng[1](✉), Xujie Tan[1], and Likun Zheng[2]

[1] School of Information Science and Technology, Jiujiang University, Jiujiang, Jiangxi, China
373779432@qq.com, csdeng@jju.edu.cn
[2] School of Computer and Information Engineering, Haerbin Commerce University, Haerbin, Heilongjiang, China

Abstract. Approximate backbone subsection optimization algorithm is proposed to solve the traveling salesman problem, for the precision accuracy of the basic ant colony algorithm for solving the larger traveling salesman problem is low. First, traveling salesman problem approximate backbone is obtained by the ant colony algorithm, and then the original traveling salesman problem is sectioned based on the approximate backbone. Then the ant colony optimization algorithm is applied to solve the subsections to improve the precision accuracy of the global optimal solution. The experimental results show that the algorithm is more precision accuracy than the basic ant colony algorithms in the solution of the typical traveling salesman problem.

Keywords: Traveling salesman problem · Approximate backbone · Subsection · Ant colony optimization

1 Introduction

Traveling Salesman Problem (TSP) is a classical NP-hard combinatorial optimization problem [1]. Its model is simple but difficult to solve. Ant Colony Optimization (ACO) is another heuristic search algorithm applied to combinatorial optimization problems after the meta-heuristic search algorithms such as simulated annealing algorithm, genetic algorithm, Tabu Search algorithm and artificial neural network algorithm.

Dorigo et al. applied ant colony algorithm to classical optimization problems such as TSP and the Quadratic Assignment Problem, and got good results. But using ant colony algorithm to solve the large scale TSP directly, the efficiency is low, and the quality of the solution is not high [2, 3].

Backbone refers to the intersection of all global optimal solutions of a problem instance [4]. He et al. [4] divides the backbone application in heuristic algorithm design into two types: probability type and deterministic type. Probabilistic backbone algorithm [5–7] mainly has three stages: first, the local optimal solution solving stage;

secondly, the approximate backbone probability computing stage; finally, the probability-oriented solution stage, mainly using the approximate backbone probability for the generation of initial solution, instance transformation or local search in the neighborhood determination [6]. The deterministic backbone algorithm can be further divided into space constrained and instance reduction [6]. Schneider [8, 9], Qi et al. [10], Fischer et al. [11], Dong et al. [12] improve the efficiency of solving TSP instances by conventions; Zou et al. [13] on the basis of conventions of TSP, through the analysis of the experimental results of the instances, concluded that about 80% of the local optimal solutions are the edges of the global optimal solution, and this generalization. The rate is not related to the scale of the problem. Based on this conclusion, an approximate backbone subsection of ACO (ABSACO) algorithm for TSP is proposed.

ABSACO algorithm is based on the set of local optimal solutions obtained by ant colony algorithm, through the statistics of the edges in the set of local optimal solutions, and according to a certain weight proportion from the statistical results, select a certain number of edges to form the approximate backbone of the optimal solution; then based on the approximate backbone, the current optimal solution obtained by ant colony algorithm is sectioned. Finally, the subsections are optimized by ant colony optimization algorithm. The strategy based on approximate backbone subsection effectively decomposes the original TSP, reduces the size of the solution, so that the basic ant colony algorithm can improve the efficiency of solving large-scale TSP and improve the quality of the solution.

2 Ant Colony Algorithm

Given a group of cities N, TSP can be described as finding a closed loop of the shortest length that passes through each city only once. Let d_{ij} be the length of the path from the city i to city j. In this paper, the Euclidean distance $(d_{ij} = [(x_i - x_j)^2 + (y_i - y_j)^2]^{1/2})$ is given. Then, a TSP is a known graph G(N, E), N represents a group of cities, E represents the edges between the groups of cities, and solves the problem of a shortest path through each city in the graph G. In this paper, we solve the symmetric TSP, that is, $d_{ij} = d_{ji}$.

2.1 Ant System

Ant colony algorithm is initially applied to solve TSP. A basic ant colony algorithm-ant system (AS) can be simply described as follows: initialization parameters, m ants randomly placed in one of the corresponding n cities, the construction of each ant's City taboo table and accessible table; then, each ant according to formula (1) in accordance with the probability of moving from city i to city j, all the ants complete a round trip after the N cycle; calculate the path length of each ant in a round trip, update the pheromone according to formula (2); record the shortest round trip length of m ants. Repeat this process until the maximum number of iterations is reached.

The formula (1) is the probability formula for ant k moving from city i to city j.

$$p_{ij}^{k}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha} \cdot \left[\eta_{ij}\right]^{\beta}}{\sum_{l \in N_i^k} \left[\tau_{il}(t)\right]^{\alpha} \cdot \left[\eta_{il}\right]^{\beta}}, \quad if\ j \in N_i^k \tag{1}$$

Among, N_i^k is the collection of ant k to transfer cities. The $\tau_{ij}(t)$ indicates the pheromone concentration on the connection path between city i and city j at t time. The heuristic function $\eta_{ij} = 1/d_{ij}$ denotes the expected degree of the ant's transfer from the city i to the city j. The parameters alpha and beta are the parameters controlling the concentration of the pheromone and the expected degree of the transfer, respectively.

The formula (2) is the formula for updating pheromones.

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \tag{2}$$

Among them, ρ is a parameter; $1-\rho$ represents the evaporation of pheromone on the path between time t and time t + 1. The $\Delta\tau_{ij}^{k}(t)$ is the increment of pheromone released by the ant k on the edge e(i, j) from time t to time t + 1 per unit length. The $\Delta\tau_{ij}^{k}(t)$ is calculated by formula (3).

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L^k}(t) & \text{If the edge e(i, j) is used} \\ & \text{by the ant k at t time} \\ 0 & \text{Otherwise} \end{cases} \tag{3}$$

Q is a constant. L^k is the path length traveled by ant k.

2.2 Ant System with Elitist Strategy

The ant system with elitist strategy (AS_{elite}) mentioned in reference [14] is an improved pheromone concentration updating method based on the basic ant colony algorithm AS. The improved pheromone concentration updating formula is as follows:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) + \Delta\tau_{ij}^{*} \tag{4}$$

Among them, the $\Delta\tau_{ij}^{*}$ is the increment of pheromone on path ij caused by elite ants, which is calculated by formula (5).

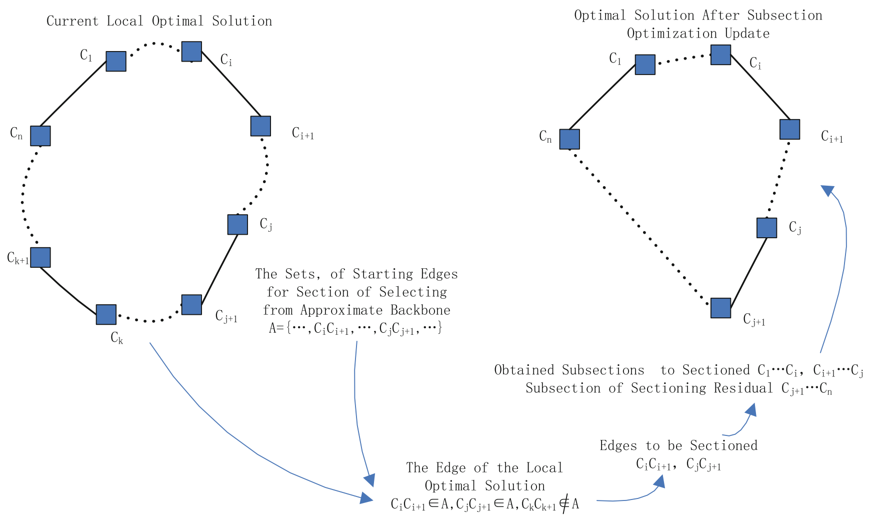
$$\Delta\tau_{ij}^{*} = \begin{cases} \rho \cdot \frac{Q}{L^*} & \text{If the edge e(i, j) is part of the} \\ & \text{optimal solution found} \\ 0 & \text{Otherwise} \end{cases} \tag{5}$$

In formula (5), ρ is the number of elite ants, and L^* is the optimal solution path length.

3 Approximate Backbone Subsection of ACO

3.1 Approximate Backbone Subsection Strategy

There are a lot of random experiments on two typical examples in TSPLIB, in reference [13]. The experiments show that approximately 80% of the edges in the local optimal solution are the edges of the global optimal solution. Approximate Backbone Subsection (ABS) strategy is based on this discovery. The edges of the local optimal solution are counted and the approximate backbone of the optimal solution is obtained. From the approximate backbone, it is the starting edge of the subsection that we choose a part of the statistical results which appear more frequently. Then the current local optimal solution is sectioned. Sectioned subsections are then optimized separately. If there is a better result, then the atomic path subsection is better instead of. The final solution is the optimal solution. The main idea of approximate backbone subsection optimization strategy is shown in Fig. 1.



Current Local Optimal Solution
C1
Ci
Ci+1
Cn
Cj
Ck+1
Cj+1
Ck
The Sets, of Starting Edges for Section of Selecting from Approximate Backbone A={⋯,CiCi+1,⋯,CjCj+1,⋯}
Optimal Solution After Subsection Optimization Update
C1
Ci
Cn
Ci+1
Cj
Cj+1
Obtained Subsections to Sectioned C1⋯Ci, Ci+1⋯Cj Subsection of Sectioning Residual Cj+1⋯Cn
Edges to be Sectioned CiCi+1, CjCj+1
The Edge of the Local Optimal Solution CiCi+1∈A,CjCj+1∈A,CkCk+1∉A

Fig. 1. Approximate backbone subsection optimization strategy main idea

For solving the traveling salesman problem in n cities, the current local optimal solution path is $C_1 \ldots C_i C_{i+1} \ldots C_j C_{j+1} \ldots C_k C_{k+1} \ldots C_n$. The Sets, of Starting Edges for Section of Selecting from Approximate Backbone, is $A = \{\ldots, C_i C_{i+1}, \ldots, C_j C_{j+1}, \ldots\}$. Thus, the $C_i C_{i+1}$ is one of edges of current local optimal solution, it belongs to set A, and the $C_j C_{j+1}$ is too. The $C_k C_{k+1}$ is one of edges of current local optimal solution, but it does not belong to set A. Based on this, there are the subsection $C_1 \ldots C_i, C_{i+1} \ldots C_j$ and $C_{j+1} \ldots C_n$ from the current local optimal solution. Finally, the subsection $C_1 \ldots C_i, C_{i+1} \ldots C_j$ and $C_{j+1} \ldots C_n$ carries on the optimization separately, the solution path after the subsection optimization renewal is the optimal solution path, and the obtained solution is the optimal solution.

3.2 Approximate Backbone Subsection of ACO Algorithm

Approximate backbone subsection colony optimization algorithm flow chart is shown in Fig. 2.

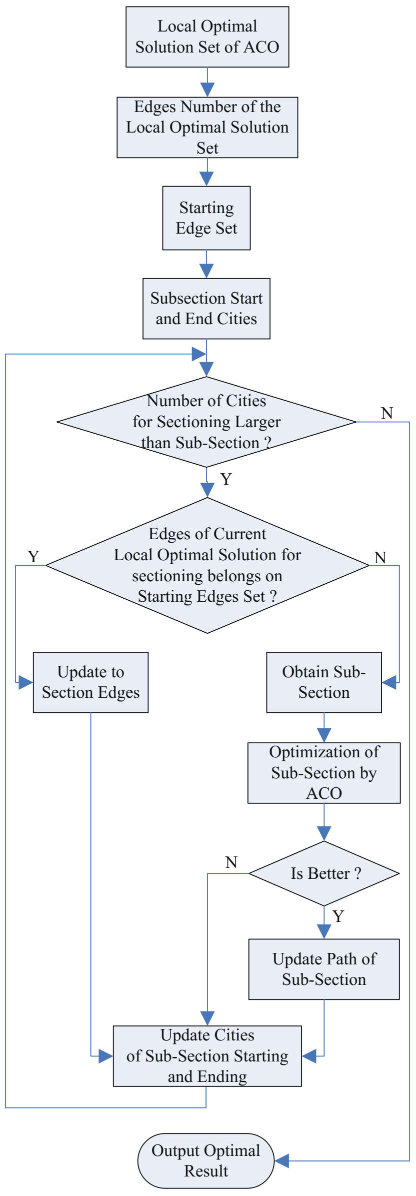

Local Optimal Solution Set of ACO
Edges Number of the Local Optimal Solution Set
Starting Edge Set
Subsection Start and End Cities
Number of Cities for Sectioning Larger than Sub-Section ?
N
Y
Edges of Current Local Optimal Solution for sectioning belongs on Starting Edges Set ?
Y
N
Update to Section Edges
Obtain Sub-Section
Optimization of Sub-Section by ACO
N
Is Better ?
Y
Update Path of Sub-Section
Update Cities of Sub-Section Starting and Ending
Output Optimal Result

Fig. 2. Approximate backbone subsection ant colony optimization algorithm flow chart

The main steps of the algorithm are described below.

Step 1: ant colony algorithm iterates the maximum $NC_{\max}$ times to generate $NC_{\max}$ local optimal solutions.

Step 2: The number matrix *BorderNums* is obtained by counting the edges of $NC_{\max}$ local optimal solutions.

Step 3: According to the weight ratio ω, the starting edges Set SSSB is obtained, whose number of edge is greater than or equal to $\omega \cdot NC_{\max}$, from the matrix BorderNums of times for each edge occurrence.

Step 4: The $\mathrm{R}_{\mathrm{ngb}}$ is a path of optimal solution to $NC_{\max}$ local optimal solution. It is sectioned to obtain the optimal solution, whose path is R_{ugb}.

Step5: Output optimal solution.

4 Experimental Results and Analysis

4.1 Experimental Environment and Parameter Description

Nine examples of universal TSPLIB (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/) were selected for the experiment. Among them, examples Oliver 30, Att48, Eil51, Berlin 52, St70, Eil76, Eil101, Pr107, Ts225 in TSPLIB provide the optimal values are 423.7406, 33523.7085, 426(429.9833), 7542(7544.3659), 675(678.5975), 538(558.7119), 629 (642.3095), 678.44303, 126643. Among them, the value in parentheses of the optimal value is the result of the optimal path provided by TSPLIB. Deviation is the result that the difference between the optimal value minus the known optimal value divide the known optimum value. In Step 1 and Step 4, ant system with elitist($\mathrm{AS}_{\mathrm{elite}}$) is used. The values of parameters alpha and beta in ant colony algorithm are set to 1 and 5. The pheromone decay factor p is 0.1 and local pheromone adjustment factor Q is 1. The $\mathrm{NC}_{\max}$ of the maximum number of iterations in Step 1 is 300 and the number of ants is 32. In Step 4, the value of $\mathrm{InnerNC}_{\max}$, maximum iterations number, is 200. The $\mathrm{m}_{\mathrm{inner}}$, the number of ants, is 8. Proportional weight Omega is set to 0.8.

4.2 Experimental Results and Analysis

1. Comparison with ACO

Table 1 shows the comparison results of 9 examples using ant colony algorithm and ABSACO algorithm. As can be seen from Table 1, the overall quality of the solution based on approximate backbone subsection is higher than that of the solution based on ant colony algorithm, and the accuracy of the results obtained by Att48, Eil51, Berlin 52, St70, Eil76, Eil101, Pr107 based on approximate backbone subsection is up to or higher than that of TSPLIB. The accuracy of the results of the optimal path is provided.

Figure 3 is the optimal solution path diagrams obtained by ABSACO algorithm for some instances.

Table 1. TSP instance test results of ACO and ABSACO

TSP instances	Optimal ACO/ABSACO	Average ACO/ABSACO	Deviation ACO/ABSACO
Oliver30	423.7406/423.7406	424.62568/423.7406	0.00000/0.00000
Att48	35221.8906/33523.7085	36061.26433/34122.03316	0.05066/0.00000
Eil51	449.2723/428.9816	460.58299/432.30439	0.05463/0.00700
Berlin52	7548.9927/7544.3659	7796.615208/7573.139225	0.00093/0.00031
St70	703.4685/677.1096	739.70106/701.960295	0.04218/0.00313
Eil76	572.4761/551.6946	588.080035/569.14083	0.06408/0.02545
Eil101	692.5415/641.0211	723.31598/680.26375	0.10102/0.01911
Pr107	46740.4675/44301.6837	47475.26795/44481.13528	0.05502/-0.00003
Ts225	131429.9493/128758.896	135826.9446/132692.7332	0.03780/0.01671

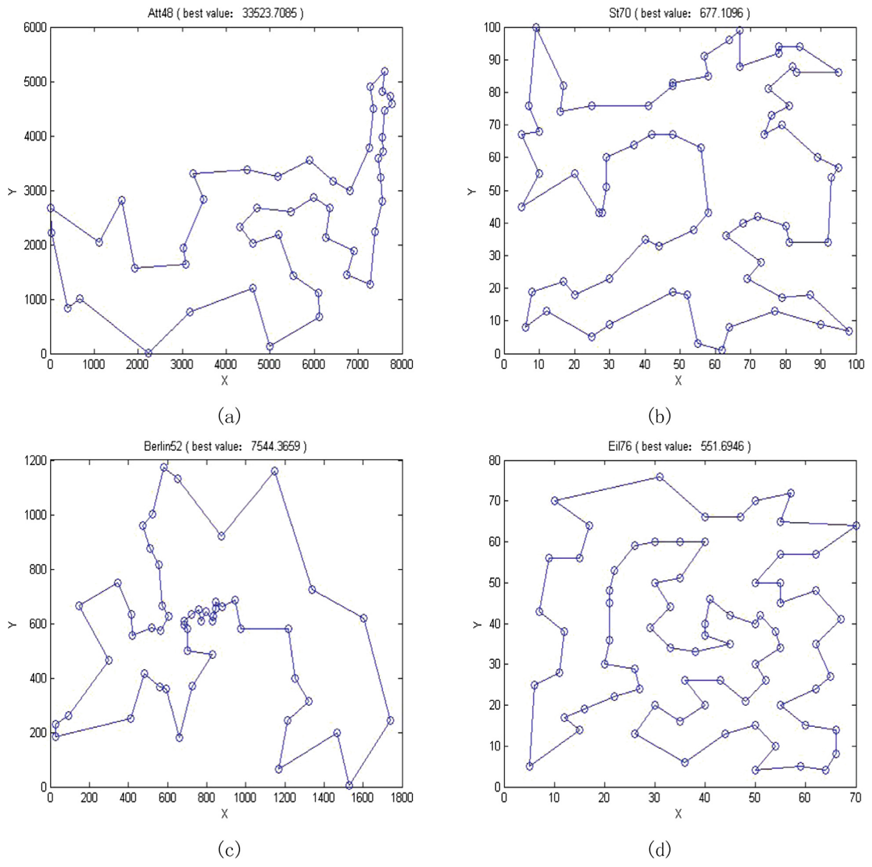


Fig. 3. Optimal solution path diagrams obtained by ABSACO algorithm

2. Comparison with IPDULACO and NACA

Compared with the existing IPDULACO [15] and NACA [16] algorithms, the quality of ABSACO algorithm is verified. As shown in Table 2, the accuracy of the solution obtained by ABSACO algorithm is higher than that obtained by IPDULACO algorithm for five instances, Eil51, St70, Eil76, Eil101, Ts225, etc. In these five instances, the quality of the solution obtained by ABSACO algorithm is improved by 0.5, 21, 1, 16.71 and 2196.58 respectively, and the relative improvement rate is 0.1164%, 3.0086%, 0.1808%, 2.5406%, 1.6773%. IPDULACO and ABSACO have the same precision for example Oliver 30, but the average value of ABSACO algorithm is 423.74. The algorithm only needs one time to get the global optimal solution; the two algorithms solve the example Oliver 30, Eil51, St70, and the average value of the optimal solution of ABSACO algorithm relative to IPDULACO algorithm is extracted respectively, to 0.78, 8.37 and 6 higher. The ABSACO algorithm is more stable.

Table 2. Comparison of solution quality between IPDULACO and ABSACO

TSP instances	Optimal IPDULACO/ABSACO	Average IPDULACO/ABSACO	Deviation IPDULACO/ABSACO
Oliver30	423.74/423.74	424.52/423.74	0.00000/0.00000
Eil51	429.48/428.98	440.67/432.30	0.00817/0.00700
St70	698/677	708/702	0.03407/0.00296
Eil76	553/552	563/569	0.02788/0.02602
Eil101	657.73/641.02	675.23/680.26	0.04568/0.01911
Ts225	130955.48/128758.90	131974.80/132692.73	0.03405/0.01671

As can show in Table 3, the ABSACO algorithm is better than the NACA algorithm in solving the example Eil51, Berlin 52, St70, Eil76, Pr107. The precision of the optimal value is improved, which is 11, 58, 35, 23, 2338 less than the NACA algorithm, and the quality of the solution is improved compared with the NACA algorithm 2.5000%, 0.7630%, 4.9157%, 4.0000%, 5.0129%. The average value of the optimal value obtained by ABSACO algorithm is very close to the optimal value obtained by ABSACO algorithm, which shows that ABSACO algorithm performs stably. The deviation is calculated by the known optimal solution. From the deviation point of view, the result obtained by ABSACO algorithm is closer to the known optimal solution, especially the result obtained by ABSACO algorithm for example Pr107. The result is preferable to the known optimal solution.

Table 3. Comparison of solution quality between NACA and ABSACO

TSP instances	Optimal NACA/ABSACO	Average NACA/ABSACO	Deviation NACA/ABSACO
Eil51	440/429	458/432	0.03286/0.00704
Berlin52	7602/7544	7892/7573	0.00796/0.00027
St70	712/677	768/702	0.05481/0.00296
Eil76	575/552	620/569	0.06877/0.02602
Pr107	46640/44302	48890/44481	0.05275/-0.00002

5 Summary

Aiming at the problem that the precision of basic ant colony algorithm for solving traveling salesman problem is not high, an optimization algorithm is proposed, which gets the approximate backbone by statistical local optimal solution edge and solves the traveling salesman problem piecewise on the basis of approximate backbone. The experiment shows that the algorithm based on approximate backbone subsection has high accuracy in solving large-scale traveling salesman problem, and can obtain the optimal solution in an acceptable time. On the basis of a certain amount of local optimal solution, the approximate backbone is obtained by statistics, and a certain proportion of the edges in the approximate backbone are taken as the sectioned edges. The original problem is sectioned to reduce the scale of the problem and improve the quality of the solution. The next step is to reduce the statistics and the number of subsections to improve the efficiency of solving the problem. The optimization strategy based on approximate backbone subsection is applied to other combinatorial optimization algorithms to solve other large-scale traveling salesman problems.

Acknowledgement. This work was supported by the National Natural Science Foundation of China (No. 61763019), the Natural Science Foundation of Heilongjiang Province (General Program:F2017019), the "Thirteenth Five-Year Plan" of Education Science in Jiangxi Province for 2017 (No.17YB211), and the Science and Technology Plan Projects of Jiangxi Provincial Education Department (No. GJJ161072, No. GJJ161076, No. GJJ170 953).

References

1. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, San Francisco (1979)
2. Colorni, A., Dorigo, M.: Heuristics from nature for hard combinatorial optimization problems. Int. Trans. Oper. Res. **3**(1), 1–21 (1996)
3. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. **1**(1), 53–66 (1997)
4. Jiang, H., Qiu, T., Hu, Y., Li, M.-C., Luo, Z.-X.: Backbone analysis and applications in heuristic algorithm design. Acta Autom. Sin. **37**(3), 257–269 (2011)
5. Helsgaun, K.: An effective implementation of the Lin-Kernighan traveling salesman heuristic. Eur. J. Oper. Res. **126**(1), 106–130 (2000)
6. Zhang, W.X., Looks, M.: A novel local search algorithm for the traveling salesman problem that exploits backbones. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence, pp. 343–348. Morgan Kaufmann Publishers, San Francisco (2005)
7. Helsgaun, K.: General *k*-opt submoves for the Lin-Kernighan TSP heuristic. Math. Program. Comput. **1**(2–3), 119–163 (2009)
8. Schneider, J., Froschhammer, C., Morgenstern, I., Husslein, T., Singer, J.M.: Searching for backbones-an efficient parallel algorithm for the traveling salesman problem. Comput. Phys. Commun. **96**(2–3), 173–188 (1996)
9. Schneider, J.: Searching for backbones - a high-performance parallel algorithm for solving combinatorial optimization problems. Future Gener. Comput. Syst. **19**(1), 121–131 (2003)
10. Qi, Y.-T., Liu, F., Jiao, L.-C.: Immune algorithm with self-adaptive reduction for large-scale TSP. J. Softw. **19**(6), 1265–1273 (2008)

11. Fischer, T., Merz, P.: Reducing the size of traveling salesman problem instances by fixing edges. In: Cotta, C., van Hemert, J. (eds.) EvoCOP 2007. LNCS, vol. 4446, pp. 72–83. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71615-0_7
12. Dong, C., Jäger, G., Richter, D., Molitor, P.: Effective tour searching for TSP by contraction of pseudo backbone edges. In: Goldberg, A.V., Zhou, Y. (eds.) AAIM 2009. LNCS, vol. 5564, pp. 175–187. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02158-9_16
13. Zou, P., Zhou, Z., Chen, G.L., Gu, J.: A multilevel reduction algorithm to TSP. J. Softw. **14**(1), 35–42 (2003). http://www.jos.org.cn/1000-9825/14/35.pdf. (in Chinese with English abstract)
14. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern.-Part B **26**(1), 29–41 (1996)
15. Xu, K., Lu, H., Cheng, B., Huang, Y.: Ant colony optimization algorithm based on improved pheromones double updating and local optimization for solving TSP. J. Comput. Appl. **37**(6), 1686–1691 (2017)
16. Zhang, C., Tu, L., Wang, J.: Application of self-adaptive ant colony optimization in TSP. J. Central South Univ. (Sci. Technol.) **46**(8), 2944–2949 (2015)