A-NEARNESS ANT COLONY SYSTEM WITH ADAPTIVE STRATEGIES FOR THE TRAVELING SALESMAN PROBLEM

Jin-qiu Lv[1], Xiao-ming You[1] and Sheng Liu[2]

[1]College of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai, 201620, China
2School of Management, Shanghai University of Engineering Science
Shanghai, 201620, China

ABSTRACT

On account of ant colony algorithm easy to fall into local optimum, this paper presents an improved ant colony optimization called α-AACS and reports its performance. At first, we provide an concise description of the original ant colony system(ACS) and introduce α-nearness based on the minimum 1-tree for ACS's disadvantage, which better reflects the chances of a given link being a member of an optimal tour. Then, we improve α-nearness by computing a lower bound and propose other adaptations for ACS. Finally, we conduct a fair competition between our algorithm and others. The results clearly show that α-AACS has a better global searching ability in finding the best solutions, which indicates that α-AACS is an effective approach for solving the traveling salesman problem.

KEYWORDS

Ant colony system, α-nearness, minimum 1-tree, lower bound, adaptive strategy.

1. INTRODUCTION

Traveling Salesman Problem(TSP)[1]is one of the most intensively studied problems in prototypical optimization problems which has achieved great improvements. Formally, the TSP can be represented by asking for the minimum path among all paths with the restrictions of visiting each vertex only once and returning to the original vertex in a weighted, complete, undirected graph.

The Ant Colony Optimization(ACO)[2] is a new cooperative intelligent algorithm, which can be applied to the TSP in a straightforward way. Ants are often able to find the shortest path between a food source and the nest. They communicate via pheromone to mark their trails in variable quantities. Artificial ants imitate the behavior of ant colonies in some extent. Although the first ACO algorithm, Ant System (Dorigo, 1992; Dorigo et al., 1991a, 1996), was found to be inferior to state-of-the-art algorithms for the TSP, it provided inspiration for a number of extensions that significantly improved performance. These extensions include elitist AS, rank-based AS, MAX–MIN AS and ACS[2].

Besides, a great deal of scholars has devoted researches and effort in ACO. For example, Ying Zhang and Lijie Li adopted Dual Nearest Insertion Procedure to initialize the pheromone, integrated reinforcement learning through computing the low bound by 1-minimum spanning tree, and combined Lin Kerninghan local search[3]. Gang Hu et al. presented binary ant colony

algorithm with controllable search bias which had a good search ability and a high convergence speed [4]. A new directed pheromone for representing the global information of searching is defined by Xiangping Meng[5]. In [6], authors presented an improved ant colony algorithm based on natural selection, which 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. Tianjun Liao and Thomas Stutzle proposed UACOR, a unified ant colony optimization (ACO) algorithm for continuous optimization, which allowed the usage of automatic algorithm configuration techniques to automatically derive new ACO algorithms[7].

In this paper, a new ant colony optimization(α-AACS) is presented, which incorporates with 3-opt local search to improve the solution quality. The paper is organized as follows. Section 2 provides a description of the proposed algorithm with a new technique. Section 3 reports experimental results and section 4 comes to a conclusion.

2. ALGORITHM DESCRIPTION

2.1. α-AACS framework

The framework of α-AACS is given as follows in pseudo code.

Input: a TSP data
Set parameters;
Initialize Pheromones trails;
Compute a lower bound ;

While(termination condition not met) **do**
  Compute Heuristic information by the minimum 1-tree
  Construct the solution by adaptive strategies
  Local search by 3-opt
  Update pheromones
End

2.2. Compute Heuristic information by the minimum 1-tree

In the original ACS, when building a tour, ant k at the current position of city *i* chooses the next city *j* to move according to the so-called pseudorandom proportional rule, given by[2]:

$$\mathrm{j} = \begin{cases} \arg\max\left\{\tau_{il}\left[\eta_{il}\right]^{\beta}\right\} l \in N_i^k, \text{ if } q \le q_0; \\ J \qquad\qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{1}$$

where q is a random variable uniformly distributed in interval [0,1], $q_0(0 \le q_0 \le 1)$ is a parameter, and J is a random variable generated according to the probability distribution given by equation (2)(with α=1).

$$\mathrm{p}_{ij}^k = \frac{\left[\tau_{ij}\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}{\sum_{l \in N_i^k}\left[\tau_{il}\right]^{\alpha}\left[\eta_{il}\right]^{\beta}}, \text{ if } j \in N_i^k \tag{2}$$

where $\eta_{ij} = 1/d_{ij}$ represents the path information of edge (i, j), α and β are two weight parameters, τ_{ij} represents the intensity of pheromone, and N_i^k represents the set of cities ant has not visited in the taboo list.

However, applying this rule may risks to prevent the optimum from discovering. If a best solution contains one link, which is not connected to the several nearest neighborhoods of its two end cities, then the algorithm will have difficulties in obtaining the optimum. Therefore, in order to better reflect the possibility of a given link being a member of an optimum, we introduce the concept of α-nearness[8].

Definition1 A 1-tree for a graph G = (N, E) is a spanning tree on the node set N\{1} combined with two edges from E incident to node 1. And a minimum 1-tree is a 1-tree of minimum length.

Definition2 Let T be a minimum 1-tree of length L(T) and let $T^+(i,j)$ denote a minimum 1-tree required to contain the edge (i,j). Then the α-nearness of an edge (i,j) is defined as the quantity

$$\alpha(i,j) = L(T^+(i,j)) - L(T).$$

When edge (i, j) is added to the minimum 1-tree, one edge must be removed from the minimum 1-tree whose length is denoted by β(i, j). Thus α(i,j) = c(i,j) - β(i,j). Then, as shown in Figure 1, if (j_1,j_2) is one of the edges belong to the minimum 1-tree, *i* is one of the remaining nodes and j_1 is on that cycle that results from adding the edge (i,j_2) to the tree, then $\beta(i,j_2)$ can be computed as the maximum of $\beta(i,j_1)$ and $c(j_1,j_2)$.

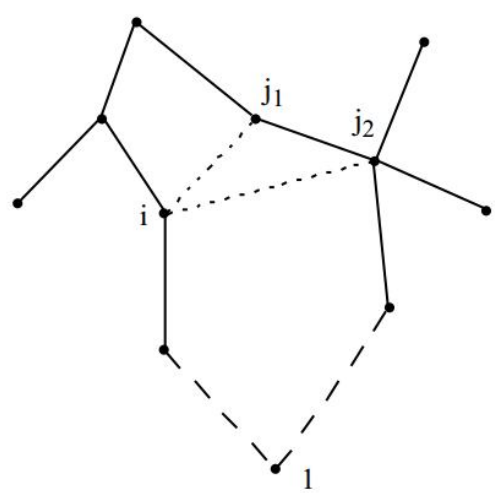


Figure.1 β(i,j2) may be computed from β(i,j1)

We use the algorithm as follows to compute α-values. Set one-dimensional auxiliary arrays *b* and *mark*, which array *b* corresponds to the β-values for a given node *i*, that is *b[j] =β(i,j)*. Array *mark* is used to indicate that *b[j]* has been computed for node *i*. The calculating of *b[j]* is conducted in two phases. First of all, the *b[j]* for all nodes *j* on the path from node *i* to the root of the tree is computed. These nodes are marked with *i*. Then, the remaining b-values are computed by the forward pass. So, wo can obtain the α-values in the inner loop.

```
for i=2:n do
   mark[i]=0;
end do
for i=2:n do
    b[i]=-∞;
```

```
k=i;
while(k!=2) do
   j=dad[k] //dad[k] denotes the father node of k.
   b[j]=max(b[k], c(k,j));
   mark[j]=i;
   k=j;
end do
for j=2:n do
    if j!=i
      if mark[i]!=i
         b[j]=max(b[dad[j]], c(j, dad[j]));
      end if
        α(i, j)=c(i, j)-b[j];
      end if
  end do
end do
```

Next, the procedure of estimating the heuristic value by minimum 1-tree is described in following steps. Here φ is a positive constant parameter.

Step1: Compute a minimum spanning tree for G'=(N\{1}, E) with Prim's algorithm;

Step2: Find a minimum 1-tree for G by adding of the two shortest edges incident to node 1 to the minimum spanning tree;

Step3: Computer the nearness α(i,j) for all edges (i,j);

Step4: $\eta_{ij} = 1/[\alpha(\mathrm{i,j}) + \varphi]$

2.3 Compute a lower bound to improve α-nearness

In the previous subsection, the α-values provide a good estimate of the edges' probability of belonging to an optimal tour. Computational tests have shown that the α-measure provides a better estimate of the likelihood of an edge being optimal than the usual c-measure[8]. Furthermore, by making a simple transformation of the original cost matrix we can improve the α-measure again. We use the transformation based on the following equation[9].

$$\mathrm{d}_{ij} = c_{ij} + \pi_i + \pi_j \qquad (3)$$

where the vector $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$. Therefore, the cost matrix C= (c_{ij}) is transformed to D = (d_{ij}), i.e., matrix C and D have the same optimal tour. The length of every tour for D is $2\sum \pi_i$ longer than the length for C . Set T_π denote a minimum 1-tree with respect D, then its length $L(T_\pi)$ is a lower bound on the length of an optimal tour for D. Therefore $\mathrm{w}(\pi) = L(T_\pi) - 2\sum \pi_i$ is lower bound on the length of an optimal tour for C. Now the work is

to find a vector $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ which maximizes the lower bound $w(\pi) = L(T_\pi) - 2\sum \pi_i$. In the situation of $w(\pi) > w(0)$, α-values of D are better estimates of edges being optimum than α-values of C.

We use an iterative method subgradient optimization[10] to maximize $w(\pi)$. The iterative equation is $\pi^{k+1} = \pi^k + t^k(0.7v^k + 0.3v^{k-1})$, where v^k is a subgradient vector which $v^{-1} = v^0$, and t^k is the step size and a positive value. The subgradient vector is computed by $v^k = d_k - 2$, where d_k is a vector whose elements are the degrees of the nodes in the current minimum 1-tree. This method makes the minimum 1-trees transform tours. Figure 2 shows the steps of a subgradient algorithm for computing the maximum of $w(\pi)$.

1. Let $k = 0, \pi^0 = 0$ and $W = -\infty$.
2. Find a minimum 1-tree, T_π^K.
3. Compute $w(\pi^k) = L(T_\pi^K) - 2\sum \pi_i$.
4. Let $W = \max(W, w(\pi^k))$.
5. Let $v^k = d_k - 2$, where d^k contains the degrees of nodes in T_π^K.
6. If $v^k = 0$ (T_π^K is an optimal tour), or a stop criterion is satisfied, then stop.
7. Choose a step size, $t^k (t^0 = 1)$.
8. Let $\pi^{k+1} = \pi^k + t^k(0.7v^k + 0.3v^{k-1})$, where $v^{-1} = v^0$.
9. Let $k = k + 1$ and go to Step 2.

Figure. 2 Subgradient optimization algorithm.

It has been proven that W will always converge to the maximum of $w(\pi)$, if $t^k \to 0$ for $k \to 0$ and $\sum t^k = \infty$ [11].

2.4 Adaptive roulette selection operator

In equation (1), the parameter q_0 determines ants whether to make the best possible move or to explore other paths by roulette selection. That is, parameter q_0 controls the algorithm's degree of exploitation and exploration. Thus, adaptive roulette selection operator presented in this paper is to set a smaller value to increase the diversity of population in the early evolution, while setting a greater value in order to accelerate the convergence in the later stage of evolution. The performance of this operator will be demonstrated by the experiments in next section.

2.5 3-opt local search

The 3-opt neighborhood consists of those tours that can be obtained from a tours by replacing at most three of its arcs, making the lengths of new tours shorter than before. The candidate set is

determined by the α-nearness in this paper and letting the length of the candidate set be 5. Table 1 shows the percent of optimal edges having a given rank among the nearest neighbors with respect to the c-measure, the α-measure, and the improved α-measure, respectively[8]. Obvious, the average rank of the optimal edges among the candidate edges is reduced to 1.7.

Table.1The percentage of optimal edges among candidate edges for the 532-city problem

rank	c	α(π=0)	improved α
1	43.7	43.7	47.0
2	24.5	31.2	39.5
3	14.4	13.0	9.7
4	7.3	6.4	2.3
5	3.3	2.5	0.9
6	2.9	1.4	0.1
7	1.1	0.4	0.3
8	0.7	0.7	0.2
9	0.7	0.2	
10	0.3	0.1	
11	0.2	0.3	
12	0.2		
13	0.3		
14	0.2	0.1	
15			
16			
17			
18			
19	0.1		
20			
21			
22	0.1		
Average ranks	2.4	2.1	1.7

The removal of three arcs results in three partial tours that can be recombined into a full tour in four different ways, as shown in Figure 3.

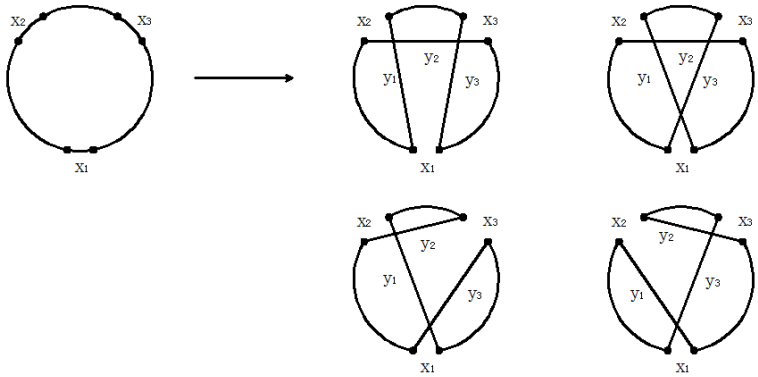


Figure.3 Four 3-opt exchanges

3. EXPERIMENT RESULTS

In this section, we carry on some computer simulations on a collection of benchmark problems from TSPLIB[12] and compare them with the known optimal solution of other algorithms. For each benchmark and each algorithm, the experiments are executed by 10 times.

Table 2 compares the proposed α-AACS(with 3-opt), ACS with 3-opt and ACS without 3-opt, using various TSP benchmark problems. Table 2 shows that: in Eil51 and Kroa150 problems, α-AACS can obtain the optimal solution; in Kroa100 Kroa200 and Pr264 problems, α-AACS can obtain the solution which is very similar to the optimal solution and the error can be approximated as 0; in the large-scale problems Lin318, the error of α-AACS is 0.37%, which has reduced by about 7% comparing with ACS.

Figure 4 compares α-AACS(with 3-opt), ACS with 3-opt and ACS without 3-opt using Kroa150 benchmark problem. The solutions generated by α-AACS and ACS with 3-opt are very close, which indicates the advantage of α-AACS is not obvious for the small scale problem. However, the situation is different obtained from figure 5. It can be observed that the α-AACS can obtain the optimum and outperform the two other algorithms. Thus, the results demonstrate our algorithm's global searching ability in finding the best solutions for large-scale problems.

Table.2 Comparison of α-AACS(with 3-opt) with other algorithms

Benchmark problem	Optimum	Algorithm	Near-optimum	Error(%)
Eil51	426	α-AACS+3opt	426.21	0
		ACS+3opt	431.17	1.21
		ACS	438.74	2.99
Kroa100	21282	α-AACS+3opt	21285.44	0.016
		ACS+3opt	21316.37	0.16
		ACS	22384.64	5.18
Kroa150	26524	α-AACS+3opt	26524.86	0
		ACS+3opt	26748.56	0.85
		ACS	28155.86	6.15
Kroa200	29368	α-AACS+3opt	29369.41	0.0048
		ACS+3opt	29834.05	1.59
		ACS	30855.32	5.06
Pr264	49135	α-AACS+3opt	49139.68	0.0095
		ACS+3opt	49729.52	1.21
		ACS	52562.55	6.97
Lin318	42029	α-AACS+3opt	42185.91	0.37

	ACS+3opt	42600.72	1.36
	ACS	45164.35	7.46

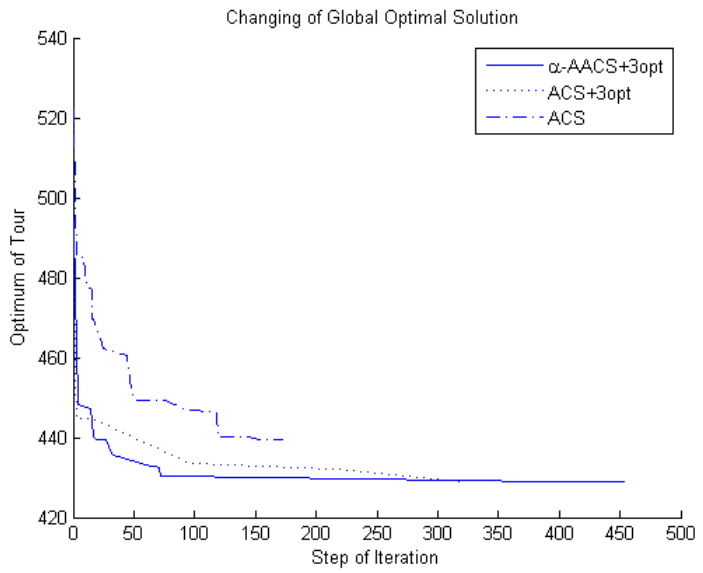


Figure. 4 Comparison of α-AACS(with 3-opt) with other algorithms using Eil51 benchmark problem

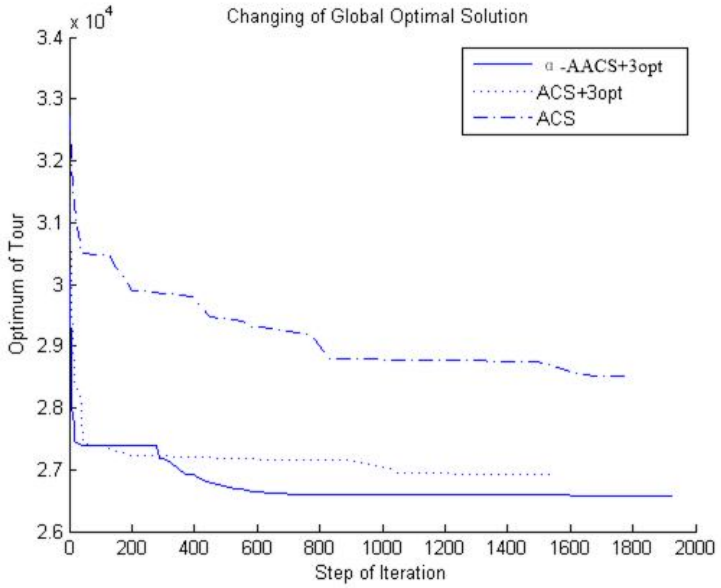


Figure. 5 Comparison of α-AACS(with 3-opt) with other algorithms using Kroa150 benchmark problem

Table 3 compares the average length of tours of our algorithm using various q_0 applied to different benchmark problems. As can be seen, when α-AACS uses the third q_0, the average lengths are all better than other two situations for three benchmark problems. According to it,

setting q_{0_1}=0.3, q_{0_2}=0.9 will yield better solutions to the TSP. What's more, setting q_{0_1}=0.3, q_{0_2}=0.9 has the stability in finding the optimum. Figure 6 shows an example of comparison of α-AACS using different q_0 for Kora200 benchmark problem. Hence, the use of q_{0_1}=0.3, q_{0_2}=0.9 with our algorithm is recommended to solve the TSP.

Table. 3 Comparison of the average tours of α-AACS using different q_0

Benchmark problem	**Set1(q_{0_1}=0.5, q_{0_2}=0.7)**	**Set2(q_{0_1}=0.4, q_{0_2}=0.8)**	**Set3(q_{0_1}=0.3, q_{0_2}=0.9)**
Eil51	431.25	430.95	429.16
Kroa100	21621.33	21342.56	21294.73
Kroa200	31589.33	29435.35	29377.54

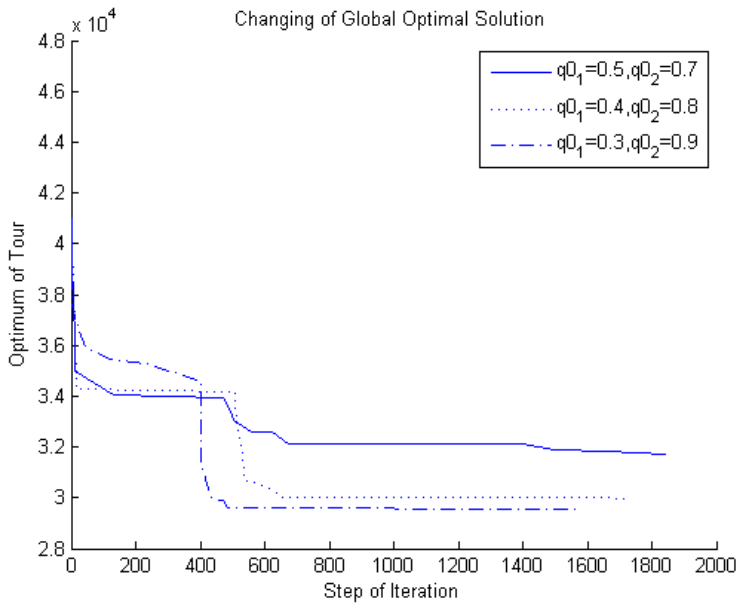


Figure. 6 Comparison of α-AACS using different q_0 in Kroa200 benchmark problem

4. CONCLUSION

This paper presents a new ant colony optimization called α-AACS algorithm for solving the TSP. We introduce the minimum 1-tree's concept, a method for computing the lower bound to improve α-nearness and the adaptive roulette selection operator which can improve the quality of solution. The experiment results show that the proposed algorithm can yield global minimum or near global minimum to the traveling salesman problem. Hence, it is an effective algorithm for the TSP. So, when q_{0_1}=0.3 and q_{0_2}=0.9, the algorithm in this paper has the best effect in solving the TSP. In the future, we will focus on different applications of ACO, such as robots path planning problem and resource constrained project scheduling problem. The former is our research core which has broad application prospects.

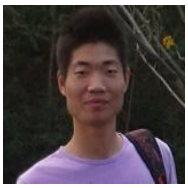
ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of Innovation Program of Shanghai Municipal Education Commission (Grant No.12ZZ185), Natural Science Foundation of China (Grant No.61075115), Foundation of No. XKCZ1212. Xiao-Ming You is corresponding author.

REFERENCES

[1] T.Stutzle and H. Hoos, MAX-MIN Ant System and Local Search for the Traveling Problem – Proceeding of the IEEE International Conference on Evolutionary Computation, 1997, 309-315.
[2] M.Dorigo and T. Stutzle, Ant Colony Optimization, MIT PRESS, 2007.
[3] Ying Zhang and Lijie Li, MST Ant Colony Optimization with Lin-Kerninghan Local Search for the Traveling Salesman Problem – ISCID, Vol.166, 2008, 344-347.
[4] Gang Hu et al., Binary ant colony algorithm with controllable search bias – Control Theory & Applications, Vol.28, 2011, No.8, 1071-1080.
[5] Xiangping Meng et al., Ant algorithm based on direction-coordinating – Control and Decision, Vol.28, 2013, No.5, 782-786.
[6] Hua-feng Wu et al., Improved ant colony algorithm based on natural selection strategy for solving TSP problem – Journal on Communications, Vol.34, 2013,No.4, 165-170.
[7] Tiaojun Liao, Thomas Stutzle, Marco A. Montes de Oca and Marco Dorigo, A unified ant colony optimization algorithm for continuous optimization – European Journal of Operational Research, Vol.234, 2014, 597-609.
[8] Keld Helsgaun, An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic – European Journal of Operational Research, Vol.126, 2000,No.1, 106-130.
[9] M.Held and R. M. Karp, The Traveling-Salesman Problem and Minimum Spanning Trees – Oper. Res., Vol.18, 1970, 1138-1162.
[10] M.Held and R. M. Karp, The Traveling-Salesman Problem and Minimum Spanning Trees: Part II – Math. Programming, Vol.1, 1971, 16-25.
[11] B.T.Poljak, A general method of solving extremum problems – Soviet Math. Dokl., Vol.8, 1967, 593-597.
[12] University of Heidelberg. TSPLIB website [EB/OL]. http://www. iwr. Uni-heidelberg. de/groups/comopt/software/TSPLIB95/tsp.

Authors



Jin-Qiu Lv was born in ZhenJiang, JiangSu, China in 1991. He received the Bachelor degree (2013) in Jiangsu University of science and technology, JiangSu, China. Now he is MSc student in the field of Intelligent Algorithms optimization in Shanghai University of Engineering Science. His current research interests include intelligent information processing and embedded systems.

Xiao-Ming You was born in 1963. She is corresponding author. She received her M.S. degree in computer science from Wuhan University in 1993, and her Ph.D. degree in computer science from East China University of Science and Technology in 2007. Her research interests include swarm intelligent systems, distributed parallel processing and evolutionary computing. She now works in Shanghai University of Engineering Science as a professor.