Available online at www.sciencedirect.com
ScienceDirect
Progress in Natural Science 18 (2008) 1417–1422
Progress in Natural Science
www.elsevier.com/locate/pnsc

An ant colony optimization method for generalized TSP problem

Jinhui Yang [a], Xiaohu Shi [a], Maurizio Marchese [b], Yanchun Liang [a,*]

[a] College of Computer Science and Technology, Jilin University, Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education, Changchun 130012, China
[b] Department of Information Engineering and Computer Science, University of Trento, Via Sommarive 14, 38050 Povo (TN), Italy

Received 5 February 2008; received in revised form 3 March 2008; accepted 3 March 2008

Abstract

Focused on a variation of the euclidean traveling salesman problem (TSP), namely, the generalized traveling salesman problem (GTSP), this paper extends the ant colony optimization method from TSP to this field. By considering the group influence, an improved method is further improved. To avoid locking into local minima, a mutation process and a local searching technique are also introduced into this method. Numerical results show that the proposed method can deal with the GTSP problems fairly well, and the developed mutation process and local search technique are effective.
© 2008 National Natural Science Foundation of China and Chinese Academy of Sciences. Published by Elsevier Limited and Science in China Press. All rights reserved.

Keywords: Generalized traveling salesman problem; Ant colony optimization; Mutation; 2-OPT

1. Introduction

The traveling salesman problem (TSP) is a deceptively simple combinatorial problem. It can be stated very simply: a salesman spends his time visiting n cities (or nodes) cyclically. In one tour he visits each city just once, and finishes up where he started. The question is: in what order should he visit the cities to minimize the distance traveled? TSP became a benchmark for many new approaches in combinatorial optimization [1–6]. A large number of publications have been dedicated to study the TSP problem, together with some of its variations. The generalized TSP (GTSP) is a very simple but practical extension of TSP. In the GTSP problem, the set of nodes is the union of m clusters, which may or may not be intersected. Each feasible solution of GTSP, called a g-tour, is a closed path that includes at least one node from each cluster, and the objective is to find a g-tour with the minimum cost. In a special case of GTSP, called E-GTSP, each cluster is visited exactly once.

GTSP was introduced by Henry-Labordere [7], Saksena [8] and Srivastava [9] in the context of computer record balancing and of visit sequencing through welfare agencies since the 1960s. A wide variety of real world problems can be modeled as GTSP. In fact, TSP could be considered as a specific form of GTSP. Therefore, many researchers have focused on this interesting model and developed many efficient methods. The former methods were mainly based on dynamic programming [7–10], which could transfer GTSP into TSP. In 1993, Fischetti [11] developed a branch-and-cut algorithm for the GTSP problems. Unfortunately, these methods could only cope with small GTSP problems for their low efficiencies. Recently, some heuristic algorithms have been proposed to solve the GTSP problems. Snyder and Daskin proposed a hybrid GTSP solving algorithm based on random-key GA and local search method (HRKGA) [12]. By designing a generalized chromosome, Wu et al. have developed a generalized

* Corresponding author. Tel.: +86 431 85153829; fax: +86 431 85168752.
E-mail address: ycliang@jlu.edu.cn (Y. Liang).

1002-0071/$ - see front matter © 2008 National Natural Science Foundation of China and Chinese Academy of Sciences. Published by Elsevier Limited and Science in China Press. All rights reserved.
doi:10.1016/j.pnsc.2008.03.028

chromosome-based genetic algorithm [13]. The generalized chromosome could unify the GTSP and TSP problems into one uniform mode by introducing "super vertex". Tasgetiren et al. [14] and Shi et al. [15] have proposed two discrete PSO-based algorithms for GTSP, respectively. More recently, a generalized chromosome genetic algorithm is analyzed and applied to consistently solve the GTSP and the classical TSP [16].

Inspired by the foraging behavior of ant colonies, Dorigo et al. developed the ant colony optimization (ACO) which is applied to the TSP [17]. From that many advanced ACO algorithms have been proposed. Typical of these are Ant System with elitist strategy and ranking (ASrank) [18], Ant Colony System (ACS) [19], and MAX-MIN Ant System (MMAS) [20]. And the application fields of ACO have been extended from TSP to the quadratic assignment problem, scheduling problem, and vehicle routing problem (see [21] for an introduction and overview).

Focusing on the GTSP problem, this paper proposes an ACO-based schedule and extended ACO application to this practical field. For brevity of the presentation, only E-GTSP will be considered in this paper. A mutation process and a local searching technique are also introduced into this method. To test the effectiveness of the proposed method, 20 GTSP problems are examined as benchmarks. Numerical results show that the proposed method can deal with the GTSP problems fairly well, and the developed mutation process and the local search technique are effective.

2. Generalized traveling salesman problem

The generalized traveling salesman problem (GTSP) has been introduced by Henry-Labordere, Saksena and Srivastava in the context of computer record balancing and of visit sequencing through welfare agencies since the 1960s. The GTSP represents a kind of combinatorial optimization problem. It can be described as the problem of seeking a special Hamiltonian cycle with the lowest cost in a complete weighted graph. Let $G = (X, E, W)$ be a complete weighted graph, where

$$X = (x_1, x_2, \ldots, x_n) \quad (n \geq 3)$$

$$E = \{e_{ij} | x_i, x_j \in X\}$$

and

$$W = \{w_{ij} | w_{ij} \geq 0 \quad \text{and} \quad w_{ii} = 0, \forall i, j \in \{1, 2, \ldots, n\}$$

are vertex set, edge set and cost set, respectively. The vertex set X is partitioned into m possibly intersecting groups X_1, X_2, …, X_m, with $|X_i| \geqslant 1$ and $X = \bigcup_{j=1}^{m} X_j$. The special Hamiltonian cycle is required to pass through all of the groups, but not all of the vertices differing from that of the TSP. For convenience, we also call W as the cost matrix and take it as $W = (w_{ij})_{n \times n}$. There are two different kinds of GTSP under the abovementioned framework of the special Hamiltonian cycle: (1) the cycle passes exactly one vertex in each group and (2) the cycle passes at least one vertex in each group. In this paper we only discuss the GTSP for the first case, namely, the E-GTSP.

3. Ant colony optimization for TSP

The ACO is developed according to the observation that real ants are capable of finding the shortest path from a food source to the nest without using visual cues. To illustrate how the "real" ant colony searches for the shortest path, an example from [22] will be introduced for better comprehension. In Fig. 1(a), suppose A is the food source and E is the nest. The goal of the ants is to bring the food back to their nest. Obviously, the shorter paths have advantage compared with the longer ones. Suppose that at time $t = 0$ there are 30 ants at point B (and 30 at point D). And at this moment there is no pheromone trail on any segments. So the ants randomly choose their path with equal probability. Therefore, on the average 15 ants from each node will go toward H and 15 toward C (Fig. 1(b)). At $t = 1$ the 30 new ants that come to B from A find a trail of intensity, 15 on the path that leads to H, laid by the 15 ants that went that way from B, and a trail of intensity 30 on the path to C, obtained as the sum of the trail laid by the 15 ants that went that way from B and by the 15 ants that reached B coming from D via C (Fig. 1(c)). The probability of choosing a path is therefore biased, so that the expected number of ants going toward C will be double of those going toward H: 20 versus 10, respectively. The same is true for the new 30 ants in D which come from E. This process continues until all of the ants will eventually choose the shortest path.

Given an n-city TSP with distances d_{ij}, the artificial ants are distributed to these n cities randomly. Each ant will choose the next to visit according to the pheromone trail remained on the paths just as mentioned in the above example. However, there are two main differences between artificial ants and real ants: (1) the artificial ants have "memory"; they can remember the cities they have visited and therefore they would not select those cities again. (2) The artificial ants are not completely "blind"; they know the distances between two cities and prefer to choose the

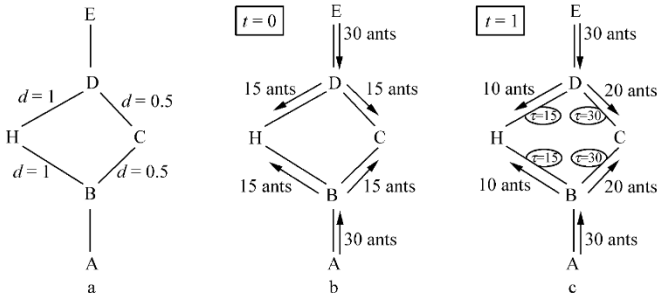

Fig. 1. An example with artificial ants [21]. (a) The initial graph with distances. (b) At time $t = 0$ there is no trail on the graph edges; therefore, ants choose whether to turn right or left with equal probability. (c) At time $t = 1$ trail is stronger on shorter edges, which are therefore, in the average, preferred by ants.

nearby cities from their positions. Therefore, the probability that city j is selected by ant k to be visited after city i could be written as follows:

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}\cdot[\eta_{ij}]^{\beta}}{\sum_{s\in allowed_k}[\tau_{is}]^{\alpha}\cdot[\eta_{is}]^{\beta}} & j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where τ_{ij} is the intensity of pheromone trail between cities i and j, α the parameter to regulate the influence of τ_{ij}, η_{ij} the visibility of city j from city i, which is always set as $1/d_{ij}$ (d_{ij} is the distance between city i and j), β the parameter to regulate the influence of η_{ij} and $allowed_k$ the set of cities that have not been visited yet, respectively.

At the beginning, l ants are placed to the n cities randomly. Then each ant decides the next city to be visited according to the probability p_{ij}^{k} given by Eq. (1). After n iterations of this process, every ant completes a tour. Obviously, the ants with shorter tours should leave more pheromone than those with longer tours. Therefore, the trail levels are updated as on a tour each ant leaves pheromone quantity given by Q/L_k, where Q is a constant and L_k the length of its tour, respectively. On the other hand, the pheromone will evaporate as the time goes by. Then the updating rule of τ_{ij} could be written as follows:

$$\tau_{ij}(t+1) = \rho\cdot\tau_{ij}(t) + \Delta\tau_{ij} \quad (2)$$

$$\Delta\tau_{ij} = \sum_{k=1}^{l}\Delta\tau_{ij}^{k} \quad (3)$$

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{if ant } k \text{ travels on edge } (i,j) \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where t is the iteration counter, $\rho \in [0,1]$ the parameter to regulate the reduction of τ_{ij}, $\Delta\tau_{ij}$ the total increase of trail level on edge (i,j) and $\Delta\tau_{ij}^{k}$ the increase of trail level on edge (i,j) caused by ant k, respectively.

After the pheromone trail updating process, the next iteration $t+1$ will start. Fig 2 is the pseudo-code of the algorithm.

4. Ant colony optimization method for GTSP

4.1. Extended ACO method for GTSP

In GTSP problem, the n cities are divided into m groups. Each group should be visited by exactly one city. Similarly as ACO for GTSP, the probability that city j is selected by ant k to be visited after city i is computed according to two factors, namely, that the pheromone trail quantity distributed on the paths and the visibility of city j from city i. But the ants must be able to identify the cities in the same classes of the ones having been visited. A simple idea is that the probability should also be computed by Eq. (1), where the set $allowed_k$ has different meaning. It is the set of cities which are in the classes having not been visited by ant k. And the updating rule of τ_{ij} is also followed by Eqs. (2)–(4). Denote $visited_k$ and $tabu_k$ as the set of visited cities and the set of groups having been visited by ant k, respectively. Then

$$allowed_k = \{x | x \in X \text{ and } x \notin G, \forall G \in tabu_k\}$$

The formally extended ACO algorithm for GTSP is

1. Initialize:
   Set *time*:=0 {*time* is time counter}
   For every edge (i,j) set an initial $\tau_{ij} = c$ for trail density and $\Delta\tau_{ij} = 0$.
2. Set s:=0 {s is travel step counter}
   For k:=1 to l do
   Place ant k on a city randomly. Place the city in $visited_k$.
   Place the group of the city in $tabu_k$.
3. Repeat until $s \leqslant m$
   Set s:=$s+1$
   For k:=1 to l do
   Choose the next city to be visited according to the probability p_{ij}^{k} given by Eq. (1).
   Move the ant k to the selected city.
   Insert the selected city in $visited_k$.
   Insert the group of selected city in $tabu_k$.
4. For k:=1 to l do
   Move the ant k from $visited_k(n)$ to $visited_k(1)$.
   Compute the tour length L_k traveled by ant k.
   Update the shortest tour found.
   For every edge (i,j) do
   For k:=1 to l do
   Update the pheromone trail density τ_{ij} according to Eqs. (2)–(4).
   time:=*time* + 1
5. If (time<TIME_MAX) then
   Empty all $visited_k$ and $tabu_k$
   Goto Step 2.
   Else
   Print the shortest tour.
   Stop

```
Initialize
For t=1 to iteration number do
  For k=1 to l do
    Repeat until ant k has completed a tour
      Select the city j to be visited next
      With probability p_ij given by Eq. (1)
    Calculate L_k
  Update the trail levels according to Eqs. (2-4).
End
```

Fig. 2. Pseudo-code of ACO for TSP.

4.2. ACO for GTSP considered group influence

The above algorithm could extend ACO from TSP to GTSP simply with a minor modification of the construction for the set $allowed_k$. But the previous algorithm does not consider the influence of group when selecting the next city to be visited. To our intuition, when two cities are neighbors in the tour, their groups always have a short

"distance". So another factor should be considered when selecting the next city, namely, that the cities in those groups near to the current city are preferred to be selected. For a city j, denote $C(j)$ as the group that contains city j. Then the probability that city j is selected by ant k to be visited after city i could be re-written as follows:

$$p_{ij}^k = \begin{cases} \dfrac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta \cdot q_{iC(j)}^k}{\sum_{s \in allowed_k} [\tau_{is}]^\alpha \cdot [\eta_{is}]^\beta \cdot q_{iC(s)}^k} & j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

where

$$q_{it}^k = \begin{cases} \dfrac{\sum_{r \in G_t} [\tau_{ir}]^\alpha \cdot [\eta_{ir}]^\beta}{\sum_{s \in allowed_k} [\tau_{is}]^\alpha \cdot [\eta_{is}]^\beta} & G_t \notin tabu_k \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

is the factor of group influence. Obviously, q_{it}^k has the larger value when the tth group G_t is "nearer" to the group contained city i. Therefore, this method is all the same with the above basic idea method except for that p_{ij}^k should be computed by Eqs. (5), (6) instead of Eq. (1).

4.3. Mutation process

In order to avoid to be locked into local minima, the mutation idea is introduced from the Genetic Algorithm. After an ant completes its tour, it will perform the mutation process according to the given mutation probability p_{mute}. In the process, a city is randomly removed from the tour, replacing the city with another city randomly chosen from the same group, and finally the randomly chosen city is inserted into the $m-1$ slots. The shortest tour of all the potential ones, including the original tour, is the new solution of the concerned ant. Denote $N(i,j)$ as the jth city of the ith group. The pseudo-code of the mutation process can be shown by Fig. 3.

4.4. 2-OPT local search

To speed up the convergence, a local searching technique called "2-OPT Local search" [23] is used in the proposed method. The function of the process could be considered as to delete the crossover of traveling lines. The 2-OPT local search basically removes two edges from the tour, and reconnects the two paths created. There is only one way to reconnect the two paths so that we still have a valid tour (Fig. 4). We do this only if the new tour will be shorter. It means that a crossover point is deleted in the original tour. The pseudo-code of the 2-OPT process is shown as Fig. 4 and the schematic diagram is shown by Fig. 5, respectively.

```
Randomly select t, let 0≤t≤m
Randomly select s, let 0≤s≤|X_{C(visited(t))}|
For i:=1 to m-1 do
    Insert the node N(visited(t), s) after the ith
        node, compute the length L̃_i
Find the shortest L̃_shortest
If ( L̃_shortest < L_origin)
    Update the tour with the shortest one
End
```

Fig. 3. Pseudo-code of the mutation process.

```
For i:=1 to i<m-3 do
  For j:=i+2 to j<m do
    If (d_{i,i+1}+d_{j,j+1}>d_{ij}+d_{i+1,j+1})
      For k:=0 to k<(j-i)/2 do
        swap( x_{j-k} , x_{i+k+1}).
```

Fig. 4. The pseudo-code of the 2-OPT process.

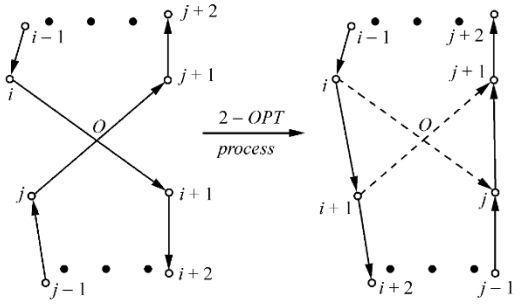

Fig. 5. Schematic diagram of the 2-OPT process.

5. Numerical simulation

To verify the validity of our proposed methods, we calculate 20 instances on a PC with 1.8 GHz processor and 1G memory. These instances can be obtained from TSPLIB library [24] and were originally generated for testing standard TSP algorithms. To test GTSP algorithms, Fischetti et al. [11] provided a partition algorithm to convert the instances used in TSP to those which could be used in GTSP. Because the partition algorithm can generate the same results at different experiments provided that the data order are the same, the partition algorithm can be used to generate test data for different algorithms.

In the experiments, basic extended ACO method, ACO considered group influence, ACO considered group influence plus mutation process, ACO considered group influence plus 2-OPT and (5) ACO considered group influence plus both mutation process and 2-OPT were all performed five times. By trial and error, the mutation probability p_{mute} was selected as 0.05 in all experiments. The results of comparison are shown in Table 1. The first column represents the names of the test instances, the second column represents the exact optimal tour length for each problem given in Ref. [12], and the third to the twelfth columns represent the minimum and average lengths obtained by the above five methods, respectively. From Table 1 it can be noticed

Table 1
Comparison results of the five proposed methods for benchmark test problems

Problem	Opt	Method 1		Method 2		Method 3		Method 4		Method 5	
		Min	Ave	Min	Ave	Min	Ave	Min	Ave	Min	Ave
11EIL51	174	176	178.8	176	177.8	176	176.8	176	177.8	176	**176**
14ST70	316	315	320.2	316	318.8	315	317.4	315	316.6	315	**315.2**
16EIL76	209	214	216	214	216.6	214	**214.4**	214	216.4	214	215.2
16PR76	64825	65078	65985.4	65078	65994.6	65157	65880.6	65253	66056	65059	**65125**
20KROA100	9711	9765	9892.8	9753	9834.2	9753	9822	9753	**9759.8**	9758	9777.6
20KROC100	9554	9859	9956.8	9579	9769.8	9569	9646	9569	9706.8	9569	**9634.4**
20KROD100	9450	9651	9736.2	9532	9682.4	9493	**9517**	9493	9648.4	9481	9555
20KROE100	9523	9648	9732.6	9651	9726.2	9560	9650.6	9560	9685.4	9535	**9616.2**
20RAT99	497	503	507.8	503	507	503	**506.2**	503	507.6	503	507.6
20RD100	3650	3712	3749.6	3712	**3727**	3705	3741.6	3718	3749.4	3705	3744
21EIL101	249	259	**263.6**	259	264.6	259	266	259	264.6	259	266.8
21LIN105	8213	8290	8314	8253	8293	8213	8255.2	8213	8267	8213	**8235.4**
22PR107	27898	27901	27940.6	27901	27937.8	27901	27934.8	27901	27955.4	27898	**27933**
25PR124	36605	37272	37730	37047	37500.4	36903	37232.2	36903	**37104.2**	36903	37183.8
29PR144	45886	46700	47148.8	46216	46828.2	45989	46478.4	45989	**46034.4**	45989	46084.2
30KROA150	11018	11470	11750.4	11594	11674.8	11470	11618	11568	11618.2	11470	**11540.8**
30KROB150	12196	12698	**12879.8**	12763	12990.8	12737	12904.8	12887	13055.2	12763	13156
31PR152	51576	52465	52809	51602	**52151.2**	52441	52791.6	51690	52376.8	51602	52221
32U159	22664	22897	**23137.8**	22897	23144.2	22909	23236.6	22964	23328.6	22897	23391.2
40D198	10557	10681	10818.6	10687	10843.6	10681	10794	10681	10842.4	10681	**10793**

that in all the 20 instances, the fifth method obtains nine best average results, while those of the first, the second, the third and the fourth are 3, 2, 3, and 3, respectively. It is also easy to see that the last method has the best performance among the five methods for these 20 instances. Comparing the results of the first and the second method, we could find that the second method possesses 14 better results while that of the first method is only 6. Similarly, the third method possesses 16 better results than the second method and the other four results are reversed. The fourth method possesses 11 better results than the second method and seven results are reversed, of the other two results the two methods obtain the same results. These comparisons indicate that the consideration of group influence in GTSP problem could improve the performance of the original ACO method, as well as that the introduction of both mutation process and 2-OPT local search could get better results.

Table 2 shows the results of the fifth method. The first column represents the names of the test instances, the second column represents the exact optimal tour length for each problem given in Ref. [12], the third column represents the best result obtained, the fourth to the eighth columns represent the results in each run time, the ninth column represents the average result and the tenth column

Table 2
Results of ACO considered group influence plus both mutation process and 2-OPT

Problem	Opt	Best result	Run 1	Run 2	Run 3	Run 4	Run 5	Average	Err (%)
11EIL51	174	176	176	176	176	176	176	176	1.15
14ST70	316	315	315	315	316	315	315	315.2	0
16EIL76	209	214	214	216	214	216	216	215.2	2.97
16PR76	64825	65059	65253	65059	65078	65157	65078	65125	0.46
20KROA100	9711	9758	9758	9838	9758	9769	9765	9777.6	0.69
20KROC100	9554	9569	9624	9780	9579	9569	9620	9634.4	0.84
20KROD100	9450	9481	9520	9761	9481	9493	9520	9555	1.11
20KROE100	9577	9535	9651	9684	9535	9651	9560	9616.2	0.41
20RAT99	497	503	514	503	506	503	512	507.6	2.13
20RD100	3650	3705	3823	3756	3718	3718	3705	3744	2.58
21EIL101	249	259	263	274	269	259	269	266.8	7.15
21LIN105	8213	8213	8271	8213	8267	8213	8213	8235.4	0.27
22PR107	27898	27898	27970	27995	27901	27898	27901	27933	0.13
25PR124	36605	36903	37036	37272	36903	37354	37354	37183.8	1.58
29PR144	45886	45989	46238	45989	45989	45989	46216	46084.2	0.43
30KROA150	11018	11470	11585	11585	11470	11490	11574	11540.8	4.74
30KROB150	12196	12763	12958	13157	13207	12763	13695	13156	7.87
31PR152	51576	51602	51602	53025	51851	53025	51602	52221	1.25
32U159	22664	22897	24026	23424	22897	23424	23185	23391.2	3.21
40D198	10557	10681	10863	10863	10695	10863	10681	10793	2.24

represents the relative error (Err), respectively, where the relative error is calculated as

$$\mathrm{Err} = \frac{\mathrm{Ave} - \mathrm{Opt}}{\mathrm{Opt}} \times 100\% \quad (7)$$

From Table 2 it can be seen that in all the simulations among the 20 test problems, the minimum relative error is 0, the maximum relative error is 7.87%, and the average relative error is 2.06%. There are some differences in the optimal results for different instances. It might be caused by the complexity of the path.

6. Conclusions

Focused on the generalized traveling salesman problem, this paper extends the ant colony optimization method from TSP to this field. Based on the basic extended ACO method, we developed an improved method by considering the group influence. To avoid locking into local minima, a mutation process is also introduced into this method. Moreover, a local searching technique, namely, 2-OPT search is applied.

The 20 benchmark instances from TSPLIB library [24] are used to test the effectiveness of our proposed methods. The numerical results show that the consideration of group influence in GTSP problem improves the performance of the basic extended ACO method. Furthermore, when the mutation process and/or 2-OPT local search are used, the results are all better than before, especially when the two processes are both used. Our proposed methods could get fairly good results when the problem scale is less than 200 cities.

Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant Nos. 60673023, 60433020, 60703025, 10501017) and the European Commission for TH/Asia Link/010 (111084).

References

[1] Bellmore M, Nemhauser GL. The traveling salesman problem: a survey. Oper Res 1968;16:538–58.
[2] Goldberg DE. Messy genetic algorithms: motivation, analysis, and first results. Complex Syst 1989;3:493–530.
[3] Huang L, Zhou CG, Wang KP. Solving constrained traveling salesman problems by genetic algorithms. Prog Nat Sci 2003;13:295–9.
[4] Liang YC, Ge HW, Zhou CG, et al. Solving traveling salesman problems by genetic algorithms. Prog Nat Sci 2003;13:135–41.
[5] Wu CG, Liang YC, Lee HP, et al. Hybrid ant colony algorithm for traveling salesman problem. Prog Nat Sci 2004;14:631–7.
[6] Chatterjee S, Carrera C, Lynch LA. Genetic algorithms and traveling salesman problems. Eur J Oper Res 1996;93(3):490–510.
[7] Henry-Labordere AL. The record balancing problem: a dynamic programming solution of a generalized traveling salesman problem. RIRO B 1969;2:43–9.
[8] Saskena JP. Mathematical model for scheduling clients through welfare agencies. CORS J 1970;8:185–200.
[9] Srivastava SS. Generalized traveling salesman problem through *n* sets of nodes. CORS J 1969;7:97–101.
[10] Chentsov AG, Korotayeva LN. The dynamic programming method in the generalized traveling salesman problem. Math Comput Model 1997;25:93–105.
[11] Fischetti M, Salazar JJ, Toth P. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. Oper Res 1997;45:378–94.
[12] Snyder LV, Daskin MS. A random-key genetic algorithm for the generalized traveling salesman problem. Eur J Oper Res 2006;174:38–53.
[13] Wu CG, Liang YC, Lee HP, et al. A generalized chromosome genetic algorithm for generalized traveling salesman problems and its applications for machining. Phys Rev E 2004;70:016701.
[14] Tasgetiren MF, Suganthan PN, Pan QK. A discrete particle swarm optimization algorithm for the generalized traveling salesman problem. In: Proceedings of the ninth annual conference on genetic and evolutionary computation; 2007. p. 158–67.
[15] Shi XH, Liang YC, Lee HP, et al. Particle swarm optimization-based algorithms for TSP and generalized TSP. Inf Process Lett 2007;103:169–76.
[16] Yang JH, Wu CG, Lee HP, et al. Solving traveling salesman problems using generalized chromosome genetic algorithm. Prog Nat Sci 2008;18(7):887–92.
[17] Dorigo M. Optimization, learning and natural algorithms [in Italian]. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy; 1992. p. 140.
[18] Dorigo M, Maniezzo V, Colorni A. The ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern B 1996;26(1):1–13.
[19] Dorigo M, Ganbardella L. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evol Comput 1997;1(1):53–66.
[20] Stutzle T, Hoos HH. The maxmin ant system and the local search for the traveling salesman problem. In: IEEE international conference on evolutionary computing, Piscataway, NJ; 1997. p. 309–15.
[21] Dorigo M, Caro GDi. The ant colony optimization meta-heuristic. In: New ideas in optimization. London: McGraw-Hill; 1999. p. 11–32.
[22] Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern 1996;26:29–41.
[23] Voudouris C, Tsang E. Guided local search and its application to the traveling salesman problem. Eur J Oper Res 1999;113:469–99.
[24] Reinelt G. TSPLIB – a traveling salesman problem library. ORSA J Comput 1991;3:376–84.