Article

Chinese Science Bulletin

Artifical Intelligence

May 2012 Vol.57 No.15: 1886–1892
doi: 10.1007/s11434-012-5162-7

A novel method for solving the multiple traveling salesmen problem with multiple depots

HOU MengShu* & LIU DaiBo

School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China

Received December 15, 2011; accepted March 23, 2012

Multi-traveling salesman problem (MTSP) is an extension of traveling salesman problem, which is a famous NP hard problem, and can be used to solve many real world problems, such as railway transportation, routing and pipeline laying. In this paper, we analyze the general properties of MTSP, and find that the multiple depots and closed paths in the graph is a big issue for MTSP. Thus, a novel method is presented to solve it. We transform a complicated graph into a simplified one firstly, then an effective algorithm is proposed to solve the MTSP based on the simplified results. In addition, we also propose a method to optimize the general results by using 2-OPT. Simulation results show that our method can find the global solution for MTSP efficiently.

combinatorial optimization, traveling salesman problem, MTSP, algorithm

Citation: Hou M S, Liu D B. A novel method for solving the multiple traveling salesmen problem with multiple depots. Chin Sci Bull, 2012, 57: 1886–1892, doi: 10.1007/s11434-012-5162-7

The traveling salesman problem (TSP) is a typical combinatorial optimization problem. A generalization of the TSP is the multiple traveling salesmen problem (MTSP), which determines a set of routes enabling multiple salesmen to start at and return to depots.

The TSP consists of finding the shortest closed route to visit all cities. Several methods based on heuristics have been proposed to solve it, including classical search maps [1], simulated annealing [2], artificial neural networks (NNs) (Kohonen-type self-organizing maps [3], Hopfield-type NNs [4]), genetic algorithms (GAs) [5], evolutionary programming [6], ant colony optimization [7], tabu search [8], fine-tuned learning [9], etc.

Although the TSP has received a great deal of attention, research on the MTSP is limited. Bektas [10] introduced comprehensive formulations and solution procedures for the MTSP, and indicated that exact algorithms [11,12] could always obtain degenerated results when solving the MTSP. Heuristic algorithms, neural network-based methods, and ant systems have all been proposed to solve the MTSP. Heuristic algorithms are the preferred method, while neural network-based methods are widely used to solve path planning [13], robotic systems [14] and authentication [15]. Ryan et al. [16] used tabu search to solve the MTSP, while Qu et al. [17] used a competition-based neural network to solve a minmax MTSP. Thus far, GAs have been applied to a wide range of application areas, including solving the MTSP. Liaw et al. [18] proposed a hybrid genetic algorithm, which is based on tabu search, to solve the MTSP. Carter et al. [19] researched chromosome representation and related genetic operators to find an applicable method for solving the MTSP. Additionally, the ant system, which was proved by [7], is a perfectly acceptable meta-heuristic for a number of NP-hard problems. In [20], an ant system is applied to the MTSP.

The MTSP seems to be more appropriate than the TSP for practical applications and can be used to simulate many everyday applications such as transportation logistics, job planning, vehicle scheduling, and so on. Some reported applications are presented in [10]. The main applications include print press scheduling [21], crew scheduling [22], school bus routing [23], mission planning [24], and the de-

*Corresponding author (email: mshou@uestc.edu.cn)

© The Author(s) 2012. This article is published with open access at Springerlink.com

sign of global navigation satellite surveying system networks [25]. Moreover, the MTSP can be used to solve the problem of multiple traveling robots [26,27], and can be considered as a relaxation of the vehicle routing problem (VRP) [28] with the capacity restrictions removed. This means that all the formulations and solution approaches proposed for the VRP are also valid and applicable to the MTSP, by assigning sufficiently large capacities to the salesmen.

The MTSP can be extended to many variations [10]. As far as the number of depots and the target paths are concerned, it includes a single depot and multiple depots, as well as closed and open paths. A closed path starts and ends at the same depot, whereas an open path does not require returning to the original depot. This paper presents a novel method for solving a heterogeneous MTSP which allows salesmen to start from different depots and end their tours at the original depots.

1 Definition of the MTSP

Given a set of nodes and M salesmen located at each depot, the MTSP aims to find M routes for each salesman starting from a set of depots, and ending at the original depots, so that each intermediate node is visited exactly once and the total cost is minimized.

Let $G=(V, E, W)$ be a connected graph, where $V=\{v_1, v_2,...,v_n\}$ is a set of cities, and $E=\{< v_i, v_j > | v_i, v_j \in V, i \neq j\}$ is an edge set with a non-negative cost matrix $W=\{w_{ij}|$ the weight of $<v_i, v_j>\}$. The graph is said to be symmetric if any $<v_i, v_j> \in E$ satisfies $w_{ij}=w_{ji}$. In this paper, we only consider symmetric graphs that satisfy the triangle inequality.

Definition 1: $w(v_i, v_j)$ is the distance or side length between v_i and v_j, denoted by w_{ij}.

Definition 2: $d(i)$ and subD(i) denote the number of edges connecting to v_i in a given graph.

Definition 3: A path denotes a route between two endpoint nodes with degree 1.

Definition 4: A tour denotes a route that starts at one node and ends at the same node.

Definition 5: The edges connected to home depots in the final result are called primal edges.

2 Simple model

A simple model, referred to as the SModel, is used to simplify an initial graph, G. The detailed operations and related restrictive conditions are described below.

All edges belonging to E are sorted in descending order according to weight and stored in the edge set, SortEdgeArray. Then, all sorted edges are checked by eq. (1) once only. If $<v_i, v_j>$ satisfies

$$\begin{cases} d(i) > 2, \\ d(j) > 2, \end{cases} \tag{1}$$

it is deleted from SortEdgeArray, and $d(i)$ and $d(j)$ are reduced by 1. Once all edges have been checked, the left edges together with all nodes constitute one or more sub-graphs. The statistical data for the degree of each node is given in Table 1. The first column lists all possible degrees, while the second column gives the percentage of nodes with each degree. The number of nodes with degree greater than 2 accounts for about 16.5% of all the nodes. It is worth noting that the computational complexity of generating an SModel is $O(eln(e))$, where e is the number of edges.

3 New solution for the MTSP

The SModel is proposed to implement the new method, MDCP (multiple depots and closed paths). After it has been generated, the subsequent workings of the MDCP are based on the model.

3.1 Generating an SModel and deleting redundant edges

The MDCP first generates an SModel, and then deletes all redundant edges in the model and reorganizes isolated paths. According to the given statistics, about 16.5% of the nodes have a degree greater than 2 in the SModel, which means that there must be redundant edges in the model. We refer to those edges connected to nodes with a degree greater than 2 as redundant candidate edges. For each redundant candidate edge $<v_i, v_j>$, if the condition

$$d(i) > 2 \quad \text{or} \quad d(j) > 2 \tag{2}$$

is satisfied, it is deleted from SortEdgeArray. All redundant candidate edges are checked in descending order according to weight. Once all of these have been checked by eq. (2), no degree of a node is greater than 2. We denote the resulting graph as G_0.

3.2 Testing paths

S_1 and S_2 are two node sets used to record the endpoint nodes of paths, rings, and isolated nodes in G_0. If MDCP discovers a node with degree 0, the node is stored in an entry of S_2. If MDCP discovers a ring, the maximal weighted

Table 1 Statistical data for the degree of all nodes after the selection action

Degree	Percentage
2	83.5%
3	11%
4	4%
≥5	≤1.5%

edge of the ring is deleted and the two endpoint nodes are stored in an entry of S_1. The endpoints of left paths are also stored in S_1.

3.3 Generating *m* routes

Based on the elements of S_1 and S_2, the number of paths can be determined and the following operations, corresponding to the relation between the numbers of salesmen and paths, would differ.

(i) Examining the number of paths. The number of paths is, in fact, the sum of the number of entries in S_1 and S_2. We denote the number of paths as *Ps* and the number of salesmen as *M*.

$Ps < M$, if *Ps* is less than *M*, MDCP selects some paths and divides each of them into multiple paths by deleting the maximum weighted edges of each selected path until *Ps* is no less than *M*, and then continues to examine the number of paths as explained below.

$Ps = M$, if *Ps* is equal to *M*, MDCP generates *M* tours by linking the two endpoint nodes of each path. If two endpoint nodes of a path are not connected to each other, the operation proceeds to that given in section 3.4. Additionally, if *M* tours are generated successfully, MDCP optimizes the final result using the method presented in section 3.5.

$Ps > M$, if *Ps* is greater than *M*, the subsequent operation is described as follows.

(ii) Path connections. MDCP generates a sub-graph from all the endpoint nodes, denoted as $G_{sub}=(V_{sub}, E_{sub})$. *V*sub includes all the endpoint nodes stored in S_1 and S_2, while E_{sub} is an edge set whose elements are connected to the nodes stored in V_{sub}. sub$D(i)$ denotes the degree of v_i in G_{sub}. MDCP continues to check each edge stored in E_{sub}. For each $<v_i, v_j>$, if any one of the following three conditions is satisfied, $<v_i, v_j>$ is deleted from E_{sub}, and sub$D(i)$ and sub$D(j)$ are decreased by 1.

(a) $d(i)=1$, $d(j)=1$, sub$D(i)>1$ and sub$D(j)>1$;

(b) $d(i)=0$, $d(j)=1$, sub$D(i)>2$ and sub$D(j)>1$;

(c) $d(i)=1$, $d(j)=0$, sub$D(i)>1$ and sub$D(j)>2$.

According to the above three conditions, it is guaranteed that there is at least one edge connected to each node in S_1 and at least two edges connected to each node in S_2. Furthermore, the left edges of *E*sub are checked by the following three conditions. If $<v_i, v_j>$ satisfies any one of these, it is deleted from E_{sub}, and sub$D(i)$ and sub$D(j)$ are decreased by 1.

(a) $d(i)=1$, $d(j)=1$, sub$D(i)>1$; or $d(i)=1$, $d(j)=1$, sub$D(j)>1$;

(b) $d(i)=0$, $d(j)=1$, sub$D(i)>2$; or $d(i)=0$, $d(j)=1$, sub$D(j)>1$;

(c) $d(i)=1$, $d(j)=0$, sub$D(i)>1$; or $d(i)=1$, $d(j)=0$, sub$D(j)>2$.

Irrespective of whether the *M* tours are generated successfully by the operations described above, MDCP converges. The number of iterations is no greater than the sum of the number of entries in S_1 and S_2.

In other words, if the left edge connects two paths that are stored in S_1 and S_2, respectively, MDCP joins the node stored in S_2 to the path stored in S_1 and removes the corresponding entry in S_2. If the left edge connects two nodes in S_2, MDCP links the two nodes together, stores the new generated path in S_1, and removes the related entries stored in S_2. Similarly, if the left edge connects two nodes in S_1, MDCP merges the two paths into one and deletes the entry that is not used to represent the new path stored in S_1. The number of entries is thus always reduced. So, the number of iterations is no more than the sum of the number of entries in S_1 and S_2. If *G* is not a complete graph, the *M* tours may not be generated successfully by the method described above, and then the operation proceeds to that described in Section 3.4.

3.4 Related remedial methods

Definition 6. If *G* is a simple graph, *N* is the number of nodes, and δ denotes the minimum connectivity which is the minimum degree of all nodes divided by *N*.

If the number of paths whose endpoint nodes are not connected to each other is θ and the number of endpoint nodes is $\bar{\theta}$, then the number of isolated nodes is $\bar{\theta}-2\theta$. The largest possible number of edges connected to endpoint nodes is $\bar{\theta}(\bar{\theta}-1)/2$, and the maximum possibility, κ, that all the endpoint nodes are not connected to each othe is $(1-\delta)^{\gamma}$. If δ is 0.5 and $\bar{\theta}$ is 5, the probability is less than 0.001.

Two remedial methods are proposed for merging paths or converting a path into a tour.

(i) Merging two paths into one. This method is used to link two paths. Two models are proposed to implement it.

Model 1: The first model for merging two paths into one is shown in Figure 1(a). Nodes v_a, v_n, v_i, and v_k are endpoint nodes that are not connected to each other, and v_b is adjacent with v_c. The MDCP tries to connect v_k to v_c, and then connects a neighbor of v_b to v_n which is on the other side of v_c. It is worth noting that the dotted lines denote the newly added edges, while the dotted line with a slash through it denotes the edge that will be deleted from the model.

The infeasibility probability of this method, denoted as ρ_1, is $(1-2\delta(1-(1-\delta)^{\bar{\theta}-2}))^{N-\bar{\theta}-\theta_1}(1-\delta(1-\delta(1-\delta)^{\bar{\theta}-2}))\theta_1$, where θ_1 is the number of paths containing only 3 nodes.

Model 2: The second model for merging two paths into one is shown in Figure 1(b). v_i and v_k are two endpoint nodes of a path, while two adjacent nodes, v_c and v_s, are intermediate nodes of the other path. If v_i and v_k are connected to v_c and v_s, respectively, the two paths can be merged into one. Similarly, the infeasibility probability of this method, denoted as ρ_2, is $((1-2\delta^2)^{\theta-1}(1-\delta^2)^{\bar{\theta}-2\theta})^{N-\bar{\theta}-\theta}$.

The objective of the two models is the same. The probability that any two of the paths can be merged into one is no less than $1-\kappa\rho_1\rho_2$.

(ii) Converting a path into a tour. The model for con-

verting a path into a tour is shown in Figure 1(c). As before, v_b is adjacent to v_c. v_a and v_k are two endpoint nodes on the path. If v_a is connected to v_c and v_k is connected to v_b, the path can be transformed into a tour by deleting $<v_b, v_c>$ and adding $<v_a, v_c>$ and $<v_b, v_k>$. The feasibility probability that a path with N nodes can be transformed into a tour is $1-(1-\delta)(1-2\delta^2)^{N-4}(1-\delta^2)^2$.

3.5 Optimization

MDCP optimizes each of the generated tours by replacing two edges of a tour with a better strategy. Differing from 2-OPT [29], MDCP adds a serial number to each node of the tour and implements the optimization method by checking the serial numbers. The rules are given below:

(i) Any two adjacent nodes are assigned adjacent serial numbers.

(ii) Serial numbers are assigned to nodes in ascending order.

(iii) Four endpoint nodes of two paths can be linked together by two strategies satisfying theorem 1.

(iv) If the edge connecting the node with the first serial number to the node with the last serial number is fixed, theorem 2 is used to change the topology of the previous tour.

Theorem 1: For two new edges e_{ij} and e_{uv}, if the sum of the serial numbers of v_i and v_j is not equal to that of v_u and v_v, the two fixed edges can be replaced by the new edges.

Theorem 2: Let the serial number of v_i be 0 and that of v_j be n-1 and $<v_u, v_v>$ be one of the two selected edges. If the serial number of v_u is greater than the serial number of v_v, $<v_i, v_j>$ and $<v_u, v_v>$ can be replaced by $<v_i, v_u>$ and $<v_j, v_v>$.

4 Experiments and computational results

Since there are no open-source benchmarks for testing the algorithms of the MTSP, we computed some instances presented in TSPLIB [30], which is the standard public library for the TSP. Although the MTSP is different to the TSP, typical instances and optimal results of these can reflect the performance of MDCP to a great extent.

The instances tested were Euclidean, two-dimensional symmetric problems with different node-scales. The relation between TSP and MDCP is as follows: if the total number of nodes is n, the TSP solution is a connected graph with n edges and the degree of all nodes equal to 2, while the MDCP solution consists of M (M is the number of salesmen) connected graphs with n edges and the degree of all nodes equal to 2.

To the best of our knowledge, no other method has previously been proposed for solving the same problem. To evaluate the performance of MDCP, we compared the computational results with the optimal results presented in TSPLIB. Although these results cannot be compared per se, the optimal results can confirm the level of performance of MDCP to a large extent. Additionally, all computational results were optimized by the method proposed in section 3.5.

4.1 Computational results

An analysis of the general results of MDCP is presented in Table 2. The column Instance gives the names of the tested instances; column Num lists the node-scale of the tested instances; and column OPT denotes the optimal result presented in TSPLIB for each tested instance. MDCP solves all the given instances with the number of salesmen varying from 2 to 10. The average time cost for each run is listed in column T in ms. Irrespective of the number of salesmen used by MDCP, the number of edges for MDCP is equal to that of the TSP for a certain instance. As shown in Table 2, it is obvious that the results obtained differ for different numbers of salesmen. Although the calculated results are affected by the number of salesmen, there is no obvious rule to determine how many salesmen should be used to obtain the best result. For example, the result for eil51 tested using 6 salesmen is the best of all the results for this instance, and likewise, the result for st70 using 10 salesmen is the best of all the results for this instance. It is, however, a certainty that the topology of the fixed instance affects the computational result with different numbers of salesmen. The advantage of MDCP is that it converges quickly.

The values for the difference between the calculated results and the optimal results are plotted in Figure 2. For each result, the difference is computed by the following formula:

$$\text{diff} = \frac{\text{general result} - \text{OPTIMAL}}{\text{OPTIMAL}}. \quad (3)$$

The bold line denotes the pivotal line, and markers linked by other lines denote the difference of different instances tested by eq. (3). Obviously, the majority of the tested results

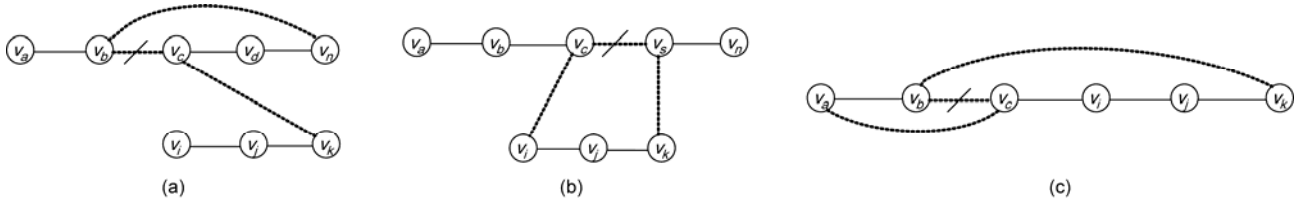

Figure 1 The two remedial models (a) and (b) are used to merge two paths into one, while (c) is used to generate a tour from a path.

Table 2 The general results of MDCP. Num is the node-scale of each instance, OPT is the optimal result presented in TSPLIB, M is the number of salesmen, T is the average time cost for each instance with different numbers of salesmen

Instance	Num	OPT	M=2	M=3	M=4	M=5	M=6	M=7	M=8	M=9	M=10	T(ms)
Eil51	51	426	446	459	456	454	444	459	471	493	490	1.3
St70	70	675	758	745	794	808	812	776	753	723	702	3.5
Eil76	76	538	573	558	569	568	580	580	601	637	642	4.5
Rat99	99	1211	1687	1543	1421	1365	1312	1290	1274	1254	1396	7.3
Kroa100	100	21282	24934	24180	25353	25294	24555	23870	24049	23399	23379	13.2
Krob100	100	22141	24752	24829	26042	25149	24273	23943	24211	24705	24165	13.0
Eil101	101	629	685	689	668	648	743	733	726	717	703	12.6
Pr107	107	44303	47428	45242	43365	41509	39690	38360	36622	34884	33512	18.9
Krob150	150	26130	30051	28704	28976	30028	30337	30665	30028	29718	28944	25.4
Kroa200	200	29368	51875	51281	50372	49413	50959	50127	49877	49498	50605	60.6
Tsp225	225	3916	4505	4432	4275	4355	4501	4574	4460	4351	4656	82.3
A280	280	2579	3014	3106	3239	3274	3225	3158	3117	3144	3174	139.5
Lin318	318	42029	53790	52380	51193	49821	49515	49839	50178	49220	49558	154.6

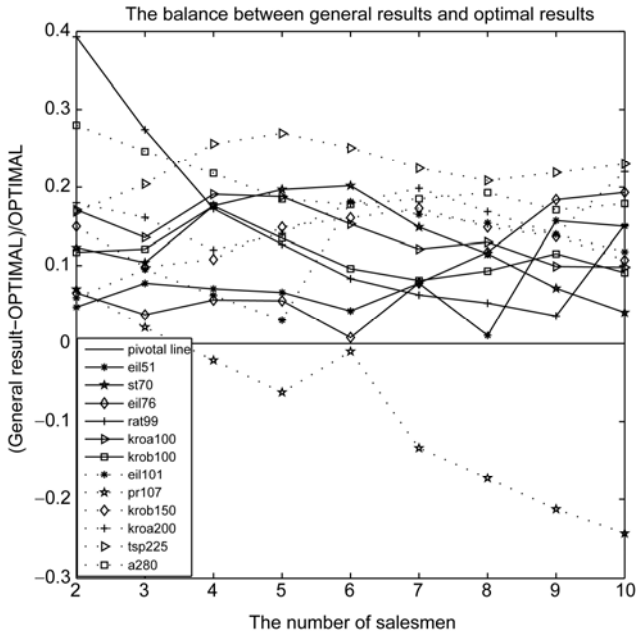

Figure 2 The differences between the general results and optimal results. The bold line denotes the pivotal line.

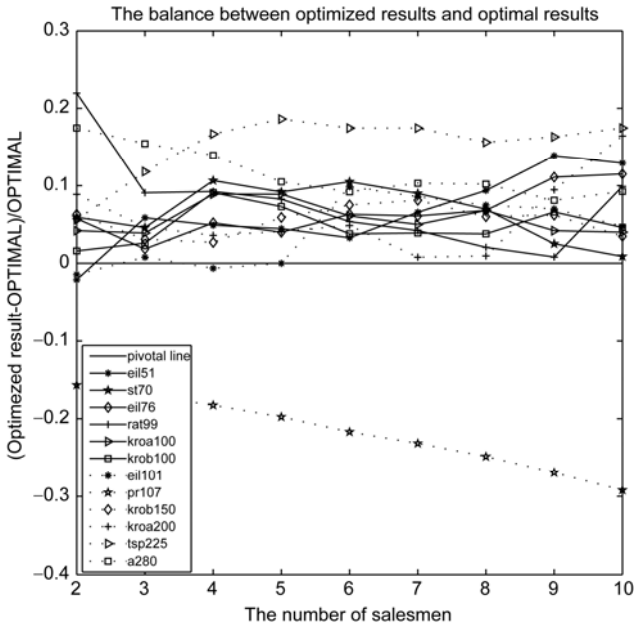

Figure 3 The differences between the optimized results and optimal results. The bold line denotes the pivotal line.

are worse than the optimal results, but the results for pr107 are better.

4.2 Optimized results

The optimization proposed in section 3.5 was used to optimize all the computational results presented in Table 2. Compared to the corresponding results in Table 2, the optimized results are much better. The values for the difference between the optimized results and the optimal results are plotted in Figure 3. Almost all differences, calculated by eq. (4) are below 0.2, with the majority between 0 and 0.1.

$$\text{diff} = \frac{\text{optimized result} - \text{OPTIMAL}}{\text{OPTIMAL}}. \tag{4}$$

However, optimizing the computational results increases the time cost. The calculation formula for the increment in time cost is expressed as

$$\text{time cost increment} = \frac{\text{cost1} - \text{cost2}}{\text{cost2}} \times 100\%, \tag{5}$$

where cost1 is the time cost of the optimization, and cost2 is

Table 3 Comparison of total cost for general results of MDCP and optimized results. The total cost decrease denotes the percentage difference in cost between the general results and the optimized results, and time cost increase denotes the percentage increase in time cost

Instance	Num	OPT	The total cost decrease (%)									Time cost increase (%)
			M=2	*M*=3	*M*=4	*M*=5	*M*=6	*M*=7	*M*=8	*M*=9	*M*=10	
Eil51	51	426	6.8	1.9	2.1	2.1	0.9	1.2	1.2	1.9	2.1	69.2
St70	70	675	6.4	14.3	6.9	10.5	9.8	5.9	4.6	4.6	3.1	60
Eil76	76	538	0.7	1.8	0.6	1.5	1.5	1.6	4.8	7.2	7.8	40
Rat99	99	1211	17.3	10.1	8.1	4.4	2.9	2.3	3.1	2.7	5.4	287
Kroa100	100	21282	12.9	9.6	10.2	9.9	9.2	7.1	6.2	5.7	5.6	92.4
Krob100	100	22141	10.2	9.5	8.5	6.2	5.8	4.1	5.5	4.9	4.5	120.7
Eil101	101	629	10.3	8.7	6.8	3.0	8.2	7.9	7.9	7.0	7.0	188
Pr107	107	44303	22.7	19.1	16.1	13.4	11.3	9.8	9.6	5.6	4.8	81.4
Krob150	150	26130	8.7	6.7	8.1	8.9	8.6	9.2	8.8	7.4	6.5	125
Kroa200	200	29368	16.1	10.6	7.3	6.7	6.6	5.3	5.1	4.4	5.5	180
Tsp225	225	3916	8.9	11.1	7.7	9.0	9.7	9.9	9.7	4.8	5.4	67
A280	280	2579	11.3	8.6	8.9	8.7	7.6	6.8	4.6	4.5	4.9	195.6
Lin318	318	42029	10.5	9.1	7.8	8.0	8.5	8.3	9.1	8.9	8.7	178

the time cost of generating the general results listed in Table 2. All the time cost increments are listed in Table 3.

4.3 Differences in results

To visually compare the general results listed in Table 2 with the optimized results, we calculated the percentage difference in total cost of the general results and optimized results for each tested instance. The larger the difference is, the better is the optimal degree. As shown in Table 3, the column labeled total cost decrease denotes the optimal degree. Most of the results listed in Table 2 decrease by 2%–10%. On the whole, the optimized results are much better than the general results. The calculation formula for the percentage difference is expressed as:

$$\text{diff} = \frac{\text{general result} - \text{optimized result}}{\text{OPTIMAL}} \times 100\%. \qquad (6)$$

5 Conclusions

This paper proposed a new method known as the MDCP to solve a heterogeneous MTSP with multiple depots and closed paths. A simple model was introduced to implement MDCP. The model can transform a complicated graph into a simplified one. Based on the model, the subsequent workings of MDCP involve merely linking paths together. Based on SModel, the greatest advantage of MDCP is that it can find a global solution efficiently.

An optimization method that is similar to 2-OPT was used to optimize the general results. Serial numbers were used to label nodes. It was experimentally verified that the optimization method can decrease the total cost of MDCP to a large extent.

This work was supported by the National Natural Science Foundation of China (61073177).

1 Gu J, Huang X. Efficient local search with search space smoothing: A case study of the traveling salesman problem (TSP). IEEE Trans Syst Man Cybern, 1994, 24: 728–735
2 Chen Y W, Lu Y Z, Chen P. Optimization with extremal dynamics for the traveling salesman problem. Physica A, 2007, 385: 115–123
3 Tarzjan S, Khademi M, Akbarzadeh T M, et al. A novel constructive-optimizer neural network for the traveling salesman problem. IEEE Trans Syst Man Cybern Part B, 2007, 37: 754–770
4 Qu H, Yi Z, Tang H J. Improving local minima of columnar competitive model for TSPs. IEEE Trans Circuits Syst Part I, 2006, 53: 1353–1362
5 Katayama K, Narihisa H. An efficient hybrid genetic algorithm for the traveling salesman problem. Electr Commun Jpn, 2001, 84: 76–83
6 Fogel D B. Applying evolutionary programming to selected traveling salesman problems. Cybern Syst, 1993, 24: 27–36
7 Dorigo M, Maniezzo V, Colorni A. Ant system: Optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B, 1996, 26: 29–41
8 Misevicius A, Smolinskas J, Tomkevicius A. Using iterated tabu search for the traveling salesman problem. Inf Technol Control, 2004, 3: 29–40
9 Coy S P, Golden B L, Runger G C, et al. See the forest before the trees: Fine-tuned learning and its application to the traveling salesman problem. IEEE Trans Syst Man Cybern Part A, 1998, 28: 454–464
10 Bektas T. The multiple traveling salesman problem: An overview of formulations and solution procedures. Omega, 2006, 34: 209–219
11 Laporte G, Nobert Y. A cutting planes algorithm for the m-salesmen problem. J Oper Res Soc, 1980, 31: 1017–1023
12 Ali A I, Kennington J L. The asymmetric m-traveling salesmen problem: A duality based branch-and-bound algorithm. Discrete Appl Math, 1986, 13: 259–276
13 Howard L, Simon X Y, Yevgen B. Neural-network-based path planning for a multirobot system with moving obstacles. IEEE Trans Syst

Man Cybern Part C, 2009, 39: 410–419
14 Barreto G A, Araujo A F, Ducker C, et al. A distributed robotic control system based on a temporal self-organizing neural network. IEEE Trans Syst Man Cybern Part C, 2002, 32: 347–357
15 Wang S H, Wang H. Password authentication using hopfield neural networks. IEEE Trans Syst Man Cybern Part C, 2008, 38: 265–268
16 Ryan J L, Bailey T G, Moore J T, et al. Reactive tabu search in unmanned aerial reconnaissance simulations. In: Proceeding of the 30th IEEE Conference on Winter Simulation. California: IEEE Press, 1998, 1: 873–882
17 Qu H, Yi Z, Tang H J. A columnar competitive model for solving multi-traveling salesman problem. Chaos Soliton Fract, 2007, 31: 1009–1029
18 Liaw C F. A hybrid genetic algorithm for the open shop scheduling problem. Eur J Oper Res, 2000, 124: 28–42
19 Carter E, Ragsdale C T. A new approach to solving the multiple traveling salesperson problem using genetic algorithms. Eur J Oper Res, 2006, 175: 246–257
20 Pan J J, Wang D W. An ant colony optimization algorithm for the multiple traveling salesmen problem. In: Proceeding of 1st IEEE Conference on Innovative Computing, Information and Control. Washington: IEEE Press, 2006, 1: 210–213
21 Gorenstein S. Printing press scheduling for multi-edition periodicals. Manage Sci, 1970, 16: 373–383
22 Svestka J A, Huckfeldt V E. Computational experience with an msalesman traveling salesman algorithm. Manage Sci, 1973, 19: 790–798
23 Angel R D, Caudle W L, Noonan R, et al. Computer assisted school bus scheduling. Manage Sci, 1972, 18: 279–288
24 Brummit B, Stentz A. Dynamic mission planning for multiple mobile robots. In: Proceeding of the IEEE International Conoference on Robotics and Automation. IEEE Press, 1996, 3: 2396–2401
25 Saleh H A, Chelouah R. The design of the global navigation satellite system surveying networks using genetic algorithms. Eng Appl Artif Intel, 2004, 17: 111–122
26 Talay S S, Erdogan D R, Dept N. Multiple traveling robot problem: A solution based on dynamic task selection and robust execution. IEEE/ASME Trans Mechatronics, 2009, 14: 198–206
27 Qu H, Yang S X, Willms A R, et al. Real-time robot path planning based on a modified pulse-coupled neural network model. IEEE Trans Neural Networks, 2009, 20: 1724–1739
28 Lau H C, Chan T M, Tsui W T, et al. Application of genetic algorithms to solve the multidepot vehicle routing problem. IEEE Trans Autom Sci Eng, 2010, 7: 383–392
29 Croes G A. A method for solving traveling salesman problems. Oper Res, 1958, 6: 791–812
30 Reinelt G, TSPLIB: A traveling salesman problem library. ORSA J Comput, 1991, 3: 376–384

Open Access This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.