A hybrid minimum spanning tree method for traveling salesman problem

Hehua Li[1*], Wei Xiong[1], Yong Wang[2]

[1]*Institute of Information Security Technology, Chongqing College of Electronic Engineering, Chongqing 401331, China*

[2]*North China Electric Power University, Beijing 102206, China*

Received 9 July 2014, www.cmnt.lv

Abstract

Traveling salesman problem (TSP) has a wide range of applications in communication, transportation, manufacturing etc. However, it is proven to be NP-complete in mathematics. An approximate method based on the Minimum Spanning Tree (MST) with an optimal four-vertex path is presented for the triangle TSP. We first compute the MST with a complete weighted graph. Then an Eulerian graph is generated by doubling the edges of the MST from an initial vertex. The optimal four-vertex path is used to simplify the Eulerian graph into a Hamiltonian cycle. Different from the common MST heuristics for TSP, all the generated four-vertex paths in the Hamiltonian cycle are the optimal four-vertex paths. Therefore, the approximation computed with the hybrid MST method is generally shorter than that produced with the common MST heuristics. The experiments for the Euclidean TSP examples also give the same conclusion.

Keywords: traveling salesman problem, minimum spanning tree, optimal four-vertex path, approximation

1 Introduction

Traveling salesman problem (TSP) is one well-known combinatorial optimization problem. Due to its NP-completeness, it is believed that there is no exact algorithms unless P=NP [1]. In practice, it has a wide range of applications in the complex systems of communication, transportation, manufacturing etc. TSP is extensively studied in the fields of computer science, operation research and discrete mathematics in order to find an efficient algorithm for it [2].

In general, TSP is represented as a weighted graph with n cities, where n is the scale of the problem. The cities and routes are mapped into the vertices and edges, respectively. The aim of TSP is to find a Hamiltonian cycle with the minimum weight, i.e., the optimal Hamiltonian cycle (OHC). The problem is easy to describe but hard to resolve. For a symmetrical TSP with n cities, the number of Hamiltonian cycles is $(n\text{-}1)!/2$. It is impossible to find the best solution through evaluating all of the Hamiltonian cycles on computers. Last century, the dynamic programming method is adopted by Held and Karp [3] and Bellman [4] to tackle TSP. They proposed the algorithm which can resolve TSP within a $O(n^2 2^n)$ computation time. Since then, TSP is considered as a hard problem with the exponential computation time [5]. It mentions that the Concorde package based on the improved cutting plane method contributed by Applegate, Bixby, Chvatal and Cook is able to resolve the TSP with less than 1000 cities in seconds or minutes [6]. It is reported that a VLSI with 85,900 cities has been resolved with the Concorde package on a networked computer system with 128 machines in 14 months. But for larger TSP, the necessary computation time and memory space cannot be evaluated.

Owing to the difficulty to design an exact algorithm, many scholars turn to pursuit the approximate algorithms. These approximate algorithms generally find an approximation near to the OHC in a polynomial computation time. Given an approximate algorithm A, it configures out the worst approximation whose weight is denoted as c^A. Let c^O denote the weight of the optimal solution. The performance of the approximate algorithm A is evaluated with the ratio which is computed as c^A/c^O. For minimized TSP, the ratio is bigger than or equal to 1. The approximate algorithms can be classified into two groups. The first group is the tour improvement methods, such as the 2-opt [7] and 3-opt [8] methods. They are efficient to generate an approximation. The representation of the tour improvement methods is the Lin-Kernighan heuristics (LKH) [9] which is derived from 2-opt and 3-opt methods. It has been tested with TSP with millions of cities and produces a satisfactory approximation quickly [9]. The tour construction methods also can generate the approximations in a short time. For example, the time complexity of the nearest neighbor algorithms [10], nearest insertion algorithm, double minimum spanning tree (DMST) [11] is only $O(n^2)$ and that of the Christofides heuristics [12] is $O(n^3)$. But the approximations usually deviate from the OHC. Hence, the improvement of these methods is always concerned by many researchers.

Besides the above exact and approximate algorithms for TSP, the meta-heuristic algorithms are extensively studied to resolve all kinds of TSP [13]. Different from the approximate algorithms, the meta-heuristic algorithms find

[*]***Corresponding author's*** e-mail: lihehuacqcet@yeah.net

the OHC or approximate solutions using the defined evolutionary rules. These evolutionary rules are believed to explore the complex landscapes more efficiently. They are derived from the evolution of natural life or physical process and they are simple to execute on computers. The representative algorithms include the heuristic Neural Network algorithm [14], Genetic Algorithm [15], Simulated Annealing algorithm [16], Ant Colony Optimization algorithm [17,18] and Particle Swarm Optimization [19] etc. Some local heuristics are combined with these meta-heuristic optimization algorithms to improve their performance [20,21]. The computation time and solution accuracy of the meta-heuristic algorithms are acceptable for some TSP instances. However, they are apt to find the local minima. Therefore, these algorithms are always being improved.

With the approximate methods of tour construction, the constraints of weights play an important role to compute a good approximation. However, the number of constraints also increases exponentially in proportion to the number of vertices. If the weights satisfy the triangle inequality, i.e., $w_{ik} \leq w_{ij}+w_{jk}$ holds for arbitrary three vertices *i*, *j* and *k*, we call this kind of TSP as the triangle TSP, such as the Euclidean TSP. Where, w_{ij} is the weight between two vertices *i* and *j*. The approximations of 2 and 3/2 times of the OHC are proven with the common MST heuristics and Christofides' heuristics, respectively. The triangle TSP has a broad background of applications and many rational approximate algorithms are designed considering the triangle inequality [22]. On the other hand, the four-vertex-and-three-line inequality [23, 24] is seldom considered for triangle TSP. For two paths with the same four vertices and two end vertices P1=(h, i, j, k) and P2=(h, j, i, k), one must be shorter than the other. If P1 is shorter than P2, we call P1 as the optimal four-vertex path. The optimal four-vertex path is computed with the four-vertex-and-three-line inequality and they are able to construct the better approximations.

Among the approximate algorithms of tour construction, we are interested in the MST heuristics. It has a small computation complexity O(n^2). Most importantly, the approximations less than 2 times of the OHC are guaranteed for the triangle TSP. It is often taken as the base by many researchers to design more complex algorithms. In this research, we improve the MST heuristics with an optimal four-vertex path. At first, a MST is computed with a complete weighted graph. Secondly, an Eulerian graph is generated by doubling the edges of the MST from an initial vertex. In the third step, the optimal four-vertex path is adopted to simplify the Eulerian graph into a Hamiltonian cycle composed of the optimal four-vertex paths. The optimal four-vertex path is computed with the four-vertex-and-three-line inequality which does not increase the computation complexity of the common MST heuristics. An optimal four-vertex path is shorter than a non-optimal four-vertex path with the same vertices and two end vertices. The Hamiltonian cycle with the optimal four-vertex paths is generally shorter than that without them. Therefore, the hybrid MST heuristics will generally compute a better approximation than the common MST heuristics does.

The rest of the paper is organized as follows. TSP based on weighted graph (WG) is briefly introduced in section 2. The function of the optimal four-vertex paths is given in section 3. The process of the hybrid MST heuristics is introduced in section 4. In section 5, the hybrid MST method is verified with tens of Euclidean TSP examples and the results are analyzed. In the last section, the merits and shortcomings of the method are summarized and the possibilities of the future research are given.

2 Description of TSP based on WG

In general, TSP is represented as a WG. For an ordinary graph including n vertices, it can be represented as $G=(V,E)$, where $V=(v_1,v_2,\cdots,v_n)$ are the vertices sets and $E=(e_{12},e_{13},\cdots,e_{(n-1)n})$ are the edges sets. $v_i(1\leq i\leq n)$ is the vertex and $e_{ij}(1\leq i\leq n, 1\leq j\leq n)$ is the edge linking the two vertices v_i and v_j in the graph G.

For a graph G with n vertices, the edges can be represented as an adjacent matrix and the values of e_{ij} is given as:

$$e_{ij}=\begin{cases}1, & \text{if } (v_i,v_j)\in E(G)\\ 0, & \text{otherwise}\end{cases}. \tag{1}$$

The two vertices v_i and v_j are called two adjacent vertices if $e_{ij}=1$. Otherwise, they are non-adjacent.

Given the weights $W=(w_{12},w_{13},\cdots,w_{(n-1)n})$ are assigned to the corresponding edges $E=(e_{12},e_{13},\cdots,e_{(n-1)n})$.

The graph G becomes one WG. The weight is often taken as distance, cost, etc in the TSP.

A Hamiltonian cycle (HC) is a circuit that visits each vertex once and exactly once. For TSP, the objective is to find the HC with the minimal weight, i.e. OHC. Given w_{ij} is the weight between two adjacent vertices v_i and v_j in the HC, the primary mathematical model of TSP is given as:

$$\left.\begin{aligned}&\min(w(HC))=\sum_{i,j=1}^{n} w_{ij}\\ &\text{s. t. } v_i\neq v_j \text{ and } e_{ij}\in E(HC)\end{aligned}\right\}, \tag{2}$$

where w(HC) is the weights of the HC and $e_{ij}(1\leq i, j\leq n)$ is the edge linking the two adjacent vertices v_i and v_j in the HC. For the symmetrical TSP with *n* vertices, the number of the HCs is (*n*-1)!/2. To decrease the complexity, more constraints [22] are added to Equation (1) to reduce the search space of the OHC.

3 The function of the optimal four-vertex path

3.1 THE OPTIMAL K-VERTEX PATH

Given a symmetrical TSP with n vertices, a Hamiltonian cycle is represented as HC=(v_1, v_2, v_3,…, v_i,…, v_n, v_1) and its two end vertices are identical, where v_i represents the i-th vertex. It includes n k-vertex paths and the ith path is noted as P_i=(v_i, v_{i+1}, v_{i+2},…, v_{i+k-1}), where i+k-1 is less than n. The number of k-vertex paths is computed as formula (3), where the superscript k represents the number of vertices in the path P^k.

$$N_{P^k} = \frac{k!}{2} C_n^k . \tag{3}$$

The OHC is the Hamiltonian cycle with the minimum weight. The OHC also includes n k-vertex paths which are different from the other paths. The relative positions of the vertices in the OHC are determined. Given an arbitrary k-vertex path in the OHC, its two end vertices are concluded and it is shorter than the other paths with the same vertices and two end vertices. We name this kind of k-vertex paths as the optimal k-vertex paths. Based on the description of optimal k-vertex paths, the number of optimal k-vertex paths is computed as Equation (4) [23]. It is obviously that the number of optimal k-vertex paths is much smaller than that of the whole k-vertex paths.

$$N_{OP^k} = \frac{k \times (k-1)}{2} C_n^k . \tag{4}$$

The k-vertex paths are divided into two kinds which are the optimal k-vertex paths and the non-optimal k-vertex paths. The number of the non-optimal k-vertex paths is the error between Equations (3) and (4). When k is bigger than 4, the number of non-optimal k-vertex paths becomes bigger than that of the optimal k-vertex paths. It notes that none of the non-optimal k-vertex paths belong to the OHC. When we search the OHC with the k-vertex paths, the non-optimal k-vertex paths can be neglected to reduce the computation time and memory space.

In the following, we give a simple example with 10 vertices to show the reduction of the search space of the OHC with optimal k-vertex paths.

The 10 vertices and their coordinates of the example are illustrated in Table 1. It is considered as one symmetrical plain TSP.

TABLE 1 The 10 vertices and their coordinates

Vertices No.	0	1	2	3	4
Coordinates	(82;7)	(91;38)	(83;46)	(71;44)	(64;60)
Vertices No.	5	6	7	8	9
Coordinates	(68;58)	(83;69)	(87;76)	(74;78)	(71;71)

Due to the simplicity of the example, we compute all of the P^is and OP^is ($4 \le i \le 10$). The number of the P^is and OP^is ($4 \le i \le 10$) are computed and the results are shown in Table 2.

TABLE 2 Comparison of number of OPis and Pis

Number of vertices i	3	4	5	6
Number of OP^is	360	1260	2520	3150
Number of P^is	360	2520	15120	75600
Number of vertices i	7	8	9	10
Number of OP^is	2520	1260	360	45
Number of P^is	302400	907200	1814400	181440

Through the results of the simple example, we know the number of Ps is much bigger than that of the OPs according to the number of vertices.

3.2 THE COMPUTATION OF OPTIMAL FOUR-VERTEX PATH

Although the OHC is composed of n optimal k-vertex paths, it is hard to compute the optimal k-vertex paths if k is big. On the other hand, the optimal k-vertex paths will be generated in a polynomial computation time when k is small. In an extreme case, the optimal four-vertex paths can be computed with the four-vertex-and-three-line inequality [24]. For the other optimal k-vertex paths with more vertices, more number of inequalities of weights will be necessary.

Given arbitrary four vertices h, i, j and k, they will form 4!/2 four-vertex paths for symmetrical TSP. The 12 paths P^4 are shown in Figure 1 and each path is noted with an integer in front of it. For example, P4=(h, i, k, j) represents the 3th path. The 12 P4s are arranged in two columns. For the two P4s in the same line, such as the 7th and 8th P4s, their two end vertices are the same whereas their two middle vertices are exchanged. One P4 will be shorter than the other P4. For example, if the 7th is shorter than the 8th path, the inequality (5) holds. This is the four-vertex-and-three-line inequality. It is used to compute the optimal four-vertex paths and these optimal paths are used to construct the approximations. In Figure 1, there are total 6 such shorter paths comparing with the other 6 paths in their same lines.

$$c_{ih} + c_{hk} + c_{kj} \le c_{ik} + c_{kh} + c_{hj} . \tag{5}$$

When the approximations are composed of the optimal four-vertex paths, the OHC of a WG with n vertices can be taken as the union of n pairs of optimal four-vertex paths. The optimization model of the TSP can be changed into Equation (6) with respect to the Equation (2).

$$\left.\begin{aligned} \min L(\mathrm{HC}) = \frac{1}{3}\min\left(\sum_{i=1}^{n} w_i\left(\mathrm{OP}_i^4\right)\right) \\ \text{s.t.}\ \ w_i\left(\mathrm{OP}_i^4\right) \le w_i'\left(\mathrm{P}_i^4\right) \end{aligned}\right\}, \tag{6}$$

where w_i represents the i-th OP_i^4 in the approximation, which is smaller than that of the P_i^4 with the same vertices and two end vertices as OP_i^4.

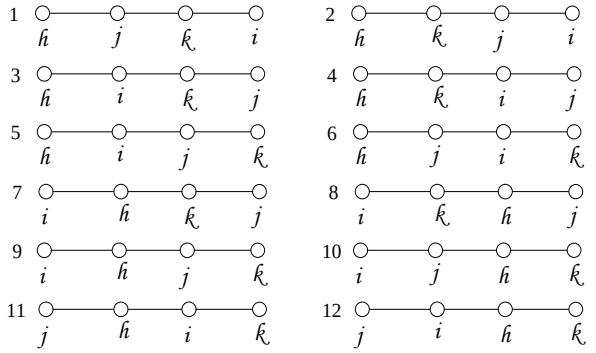

FIGURE 1 12 paths formed with 4 vertices

3.3 THE REDUCTION OF THE SEARCH SPACE WITH OPTIMAL FOUR-VERTEX PATH

It notes that the number of Hamiltonian cycles composed of the optimal four-vertex paths will be smaller than that of Hamiltonian cycles combined with the total four-vertex paths. The reason is that the number of optimal four-vertex paths is smaller than that of the total four-vertex paths. The combinations of the vertices under the four-vertex-and-three-line inequality will be less than those without considering it. For example, we want to compute the 5-vertex paths with the optimal four-vertex paths for a TSP with n vertices. Given the initial vertex is i0, the number of the 5-vertex paths composed of the optimal four-vertex paths is approximately evaluated as Equation (7), where m_5 represents the average number of the optimal four-vertex paths which can combine with the former (n-1)×(n-2)×(n-3)/2 optimal four-vertex paths. It is clearly that m5 is smaller than n-4.

$$N_5 = \frac{(n-1)\times(n-2)\times(n-3)}{2} \times m_5 . \tag{7}$$

In the previous research [25], the authors compute all of the Hamiltonian paths composed with the optimal four-vertex paths for the example in Table 1. The total number of this kind of Hamiltonian paths is no more than 30000 whereas the total number of HCs is 1814400. Hence, the search space of the OHC with the optimal four-vertex paths will be reduced a lot with the optimal four-vertex paths.

We use the optimal four-vertex paths to simplify the Eulerian graph into an approximation. The optimal four-vertex paths are shorter than the non-optimal four-vertex paths with the same vertices and two end vertices. A better approximation will be generated than that computed with the common MST heuristics. In addition, the search space of the OHC will be reduced if only the optimal four-vertex paths are used to constitute the OHC rather than the four-vertex paths.

4 The process of the hybrid MST heuristics

In the 3rd section, we have given the foundation of the optimal four-vertex paths for TSP. In this section, we use it to improve the performace of the common MST heuristics. The hybrid MST heuristics is the combination of the common MST heuristics and the optimal four-vertex paths. The process of the hybrid MST heuristics is given in Table 3. It includes four main steps.

TABLE 3 The hybrid MST heuristics

Step	The pseudo codes of the hybrid MST heuristics
1	Compute the MST using Prime algorithm for a complete weighted graph with n vertices.
2	Given an initial vertex v0, generate an Eulerian graph by doubling the edges of the MST.
3	Convert the Eulerian graph into a Hamiltonian cycle using the optimal four-vertex paths.
4	Check the four-vertex paths in the approximation. If one path is not optimal, change it into an optimal four-vertex path.

In the first step, the Prime algorithm is used to compute the MST of a weighted graph. The computation complexity is $O(n^2)$. For example, the MST of a TSP with 8 vertices is shown in Figure 2. The alphabets in Figure 2 represent the vertices of the graph. The next step is to compute an Eulerian graph by doubling the edges of the MST. We appoint vertex a as the root vertex of the Eulerian graph. Then the depth-first algorithm is used to compute the Eulerian graph as E=(a, b, d, b, e, b, a, f, g, h, g, i, g, f, a). Obviously, the length of E is 2 times of the MST, which is smaller than 2 times of the length of the OHC. For triangle TSP, we can delete the repeated vertices in E and the rest route will not become longer. In the end, we will obtain a Hamiltonian cycle of no more than 2 times of the OHC. This is the common MST heuristics. The approximation computed with the common MST heuristics is generally unsatisfactory because most of the paths in the approximation are not optimal. In the third step after the second step, we use the optimal four-vertex paths to convert the Eulerian graph into a Hamiltonian cycle, which is distinctive from the common MST heuristics. We first select a segmental route with four different vertices from the Eulerian graph and change it into a path. For example, (*a*, *b*, *d*, *b*, *e*) in *E* is selected as the first segmental route. A repeated vertex *b* will be deleted to generate a path. We can delete the vertex *b* ahead of vertex *d* or behind vertex *d* and we obtain two paths P_1=(*a*, *d*, *b*, *e*) and P_2=(*a*, *b*, *d*, *e*). The two paths will be evaluated and the optimal four-vertex path is maintained. With the Eulerian graph, the process to generate the optimal four-vertex paths is executed iteratively until a Hamiltonian cycle is obtained.

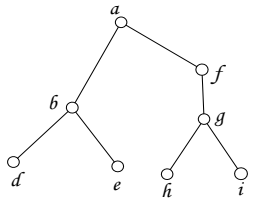

FIGURE 2 MST of a complete weighted graph with 8 vertices

The four-vertex paths in the approximation may not be optimal because some reasons. The first reason is that the appointed initial vertex may not be suitable. The second reason is that the following optimal four-vertex path may ruin the previous optimal four vertex path. To generate an approximation composed of the optimal four-vertex path,

we adjust the non-optimal four-vertex paths in the approximation into the optimal four-vertex paths.

It mentions that we compute two approximations from two sides of the Eulerian graph. For example, the first Hamiltonian cycle is computed from left to right with the Eulerian graph E whereas the second Hamiltonian cycle is generated from right to left. If the optimal four-vertex path is not used, the approximation is the same. However, the two approximations will be different when the optimal four-vertex paths are considered. In the following experiments, the shorter approximation is maintained. With the common MST heuristics, some of the generated four-vertex paths will not be the optimal four-vertex paths. With the hybrid MST heuristics, all the four-vertex paths are altered into the optimal four-vertex paths and a better approximation will be generated.

5 Experiments and discussions

Euclidean TSP is a special kind of triangle TSP. The Euclidean TSP examples are used to test the hybrid method. These Euclidean TSP instances are downloaded from TSPLIB [9]. The hybrid MST heuristics is coded with C++ language and implemented on a personal computer. The common MST heuristics is also programmed for comparisons. The time complexity of the two algorithms is $O(n^2)$. The computation time of the hybrid MST heuristics may be a little longer but it does not increase much computation time. Therefore, the computation time is neglected in the experiments. The OHC is computed with the Concorde package online, which guarantees to find the optimal solution. The error between the approximations and the OHC is computed as

$$err=(w^A-w^O)/w^O\times 100\%,$$

where w^A and w^O are the weights of the approximation and the OHC, respectively. In the experiments of these examples, the first vertex v0 is assigned as the initial vertex to compute the Euerian graph and Hamiltonian cycle. The computational results are shown in Table 4.

TABLE 4 The results of the experiments

Instances	w^{cMST}	w^{hMST}	w^{OHC}	err^{cMST}/%	err^{hMST}/%
Eil51	609.05	529.92	428.87	42.01	23.56
Berlin52	10376.30	9864.88	7544.36	37.54	30.76
Eil76	708.97	682.54	544.36	30.24	25.38
Pr76	138775.75	138230.67	108159.42	28.31	27.80
Rat99	1767.71	1598.81	1219.24	44.98	31.13
KroA100	28520.93	28249.88	21285.44	33.99	32.72
kroC100	27089.10	26604.00	20750.76	30.55	28.21
kroD100	26603.45	26320.97	21294.28	24.93	23.61
Rd100	10722.93	10459.11	7910.39	35.56	32.22
Eil101	930.45	861.49	640.21	45.34	34.56
Lin105	20878.58	20221.85	14382.99	45.16	40.60
Pr107	59435.99	52274.70	44301.67	34.16	18.00
Pr124	78922.73	76763.76	59030.72	33.70	30.04
Bier127	154983.26	150847.03	118293.44	31.02	27.52
Ch130	8091.69	7733.53	6110.71	32.42	26.56
Pr136	127267.69	127024.27	96780.45	31.50	31.25
Pr144	79284.44	79205.88	58535.21	35.45	35.31
Ch150	9115.37	8819.64	6530.90	39.57	35.04
kroA150	36159.74	34747.96	26524.85	36.32	31.00
kroB150	35607.00	34345.47	26127.34	36.28	31.45
Pr152	92753.64	92733.67	73683.62	25.88	25.85
U159	58689.95	56991.68	42075.65	39.49	35.45
Rat195	3308.15	3258.94	2333.87	41.75	39.64
D198	18932.66	18783.89	15808.65	19.76	18.82
kroA200	39379.79	37956.82	29369.40	34.08	29.24
Ts225	181702.06	164429.02	126645.93	43.47	29.83
Tsp225	5434.96	5276.97	3859.00	40.84	36.74
Pr226	115179.55	113959.75	80370.25	43.31	41.79
Pr264	64208.26	63746.83	49135.00	30.68	29.74
Pr299	64129.90	62086.54	48194.92	33.06	28.82
Lin318	60675.04	58940.90	42042.53	44.32	40.19
Rd400	20940.96	20449.47	15275.98	37.08	33.87
Fl417	16512.90	16333.49	11914.83	38.59	37.09
Pr439	141332.81	139816.73	107215.33	31.82	30.41
Pcb442	65044.95	63557.90	50783.55	28.08	25.15
D493	45799.66	45097.26	35023.07	30.77	28.76
U574	50232.21	49164.83	36934.79	36.00	33.11
Rat575	9430.82	9237.28	6798.25	38.72	35.88
P654	48630.75	48126.13	34646.83	40.36	38.90
D657	65577.11	64103.04	48915.65	34.06	31.05
U724	58221.13	56222.04	41908.10	38.93	34.16
Rat783	12292.80	11978.47	8846.15	38.96	35.41

With the 42 Euclidean TSP examples, we find the hybrid MST heuristics always generates the better approximations than the common MST heuristics does for each of the TSP instance under the same environment. It says the MST heuristics is improved when it is combined with the optimal four-vertex paths. The distributions of the approximations according to their errors are listed in Table 5. With the common MST heuristics, the number of the approximations which are more than 1.4 times of the OHC reaches 10 whereas only 3 such approximations are generated with the hybrid MST heuristics. With the hybrid MST heuristics, most of the approximations are less than 1.4 times of the OHC and all of them are bigger than 1.15 times of the OHC.

TABLE 5 Distributions of the approximations according to their errors

Error (%)	(15,20]	(20,25]	(25,30]	(30,35]	(35,40]	(40,50]
number of examples with hMST	2	2	12	14	9	3
number of examples with cMST	1	1	3	15	12	10

The change of the errors of the approximations with the two algorithms according to the scale of the examples is shown in Figure 3.

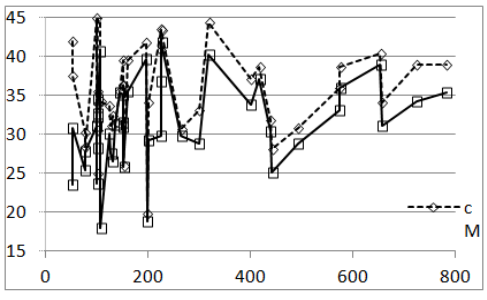

FIGURE 3 The change of the errors of the two algorithms according to the scale of the Euclidean TSP

It clearly shows that the error of the approximation generated with the hybrid MST heuristics is smaller than that of the approximation generated with the common MST heuristics. For small scale of TSP with less than 200 vertices, most of the errors with the hybrid MST heuristics

are much less than those of the common MST heuristics, such as the such as Eil51, Rat99, Eil101 and Pr107, the approximations searched with the hybrid MST heuristics are much better than those generated with the common MST heuristics. On the other hand, for the medium and large size of TSP, the errors of the approximations generated with the two algorithms have a small distinction, such as the Pr264, Fl417, Pr439, D493, U574, Rat575, P654, D657, the approximations are not enhanced too much with the hybrid MST heuristics. There are two reasons. The first is that the initial vertex may not appropriate. If we change the initial vertex to generate the Eulerian graph, a better approximation will be acquired. Of course, the computation time will be added.

6 Conclusions

The common MST heuristics is improved with the optimal four-vertex path for triangle TSP. The optimal four-vertex paths are shorter than the non-optimal four-vertex paths with the same four vertices and two end vertices. The approximation composed of the optimal four-vertex paths may be better than those without them. The time complexity of the hybrid MST heuristics is $O(n^2)$ and it is easy to implement on computers. Through the 42 Euclidean TSP examples, the hybrid MST heuristics generally compute the approximations which are less than 1.4 times of the OHC when the first vertex is appointed.

References

[1] Karp R 1975 On the computational complexity of combinatorial problems *Networks* (USA) 5(1) 45-68
[2] Johnson D S, McGeoch L A 2004 The Traveling Salesman Problem and Its Variations *Combinatorial Optimization* London Springer Press
[3] Held M, Karp R 1962 A Dynamic Programming Approach to Sequencing Problems *Journal of the Society for Industrial and Applied Mathematics* 10 196-210
[4] Bellman R 1962 Dynamic programming treatment of the travelling salesman problem *Journal of ACM* 9(1) 61-3
[5] Matai R, Prakash S, Mittal M 2010 Traveling salesman problem, theory and applications *Croatia InTech*
[6] Applegate D L, Bixby R E, Chvatal V, Cook W J 2006 The Traveling Salesman Problem, A Computational Study *Princeton: Princeton University Press*
[7] Verhoeven M G A, Aarts E H L, Swinkels P C J 1995 A parallel 2-opt algorithm for the Traveling Salesman Problem *Future Generation Computer Systems* 11(2) 175-82
[8] Luc M, Patrick B, Dirk C, Dirk V O 2005 Exploring variants of 2-Opt and 3-Opt for the general routing problem *Operations Research* 53(6) 982-95
[9] Helsgaun K 2012 An effective implementation of the Lin-Kernighan traveling salesman heuristic. Available: http://www.akira.ruc.dk/~keld/research/LKH/LKH-2.0/DOC/LKH_REPORT.pdf.
[10] Adrian D, Joseph S B M 2003 Approximation algorithms for TSP with neighborhoods in the plane *Journal of Algorithms* 48(1) 135-59
[11] Thomas H C, Charles E L, Ronald L R, Clifford S 2006 Introduction to Algorithm second edition *Beijing: China Machine Press*
[12] Christofides N 1976 Worst-case analysis of a new heuristic for the travelling salesman problem Algorithms and Complexity: New Directions and Recent Results *Academic Press*
[13] Binder P M 2008 Frustration in Complexity *Science* 320(5874) 322-3
[14] Ghaziri H, Osman I H 2003 A neural network algorithm for the traveling salesman problem with backhauls *Computers & Industrial Engineering* 44 267-81
[15] Liu Y H 2010 Different initial solution generators in genetic algorithms for solving the probabilistic traveling salesman problem *Applied Mathematics and Computation* 216 125-37
[16] Liu Y, Xiong SW, Liu H B 2009 Hybrid Simulated Annealing Algorithm Based on Adaptive Cooling Schedule for TSP *GEC'09* Shanghai China 895-8
[17] Dorigo M, Gambardella L M 1997 *IEEE Transactions on Evolutionary Computation* 1(1) 53-66
[18] Zhou Y R 2009 *IEEE Transactions Evolutionary Computation* 13(5) 1083-92
[19] Chen W N, Zhang J, Chung H S H, Zhong W L, Wi W G, Shi Y H 2010 *IEEE Transactions on Evolutionary Computation* 14(2) 278-300
[20] Bontoux B, Artigues C, Feillet D 2010 A memetic algorithm with a large neighborhood crossover operator for the Generalized Traveling Salesman Problem *Computers & Operations Research* 37(11) 1844-52
[21] Iordache S 2010 Consultant-guided search-a new metaheuristic for combinatorial optimization problems *GECCO'10* Oregon 225-32
[22] Bläser M. 2008 A New Approximation Algorithm for the Asymmetric TSP with Triangle Inequality *ACM Transactions on Algorithms* 4(4) 1-15
[23] Yong W 2013 A Representation Model for TSP *The 15th IEEE International Conference on HPCC IEEE computer society*
[24] Yong W 2012 The Frequency Graph for the Traveling Salesman Problem *World Academy of Science, Engineering and Technology/ICECECE2012* Bali
[25] Wang Y, He W, Tian D 2012 A hybrid method to search the optimal Hamiltonian circuit *The 2nd International Conference on Materials and Products Manufacturing Technology* 605-607 2149-54

Authors

Hehua Li, July 1976, Henan Province, P.R. China.

Current position, grades: associate professor at Chongqing College of Electronic Engineering and China's Western Light visiting scholar.
University studies: master's degree in computer software and theory at Chongqing University in China.
Scientific interests: network systems integration, information security.

Wei Xiong, February 1980, Chongqing City, P.R. China.

Current position, grades: lecturer at the School of Chongqing College of Electronic Engineering, P. R. China
University studies: Master's degree in Computer Science and Technology at Chongqing University, Chongqing, China in 2008.
Scientific interests: wireless networks, computer programming, mobile computing.

Yong Wang, February 1978, Henan Province, P.R. China.

Current position, grades: associate professor of North China Electric Power University, Beijing, P. R. China.
University studies: master's degree in the field of CAD & CAM at Bei Hang University in China.
Scientific interests: graph theory, meta-heuristic algorithms and their applications.