World Academy of Science, Engineering and Technology
International Journal of Electronics and Communication Engineering
Vol:6, No:10, 2012

The Frequency Graph for the Traveling Salesman Problem

Y. Wang

Abstract**—**Traveling salesman problem (TSP) is hard to resolve when the number of cities and routes become large. The frequency graph is constructed to tackle the problem. A frequency graph maintains the topological relationships of the original weighted graph. The numbers on the edges are the frequencies of the edges emulated from the local optimal Hamiltonian paths. The simplest kind of local optimal Hamiltonian paths are computed based on the four vertices and three lines inequality. The search algorithm is given to find the optimal Hamiltonian circuit based on the frequency graph. The experiments show that the method can find the optimal Hamiltonian circuit within several trials.

Keywords**—**Traveling salesman problem, frequency graph, local optimal Hamiltonian path, four vertices and three lines inequality.

I. Introduction

THE objective of TSP is to find the optimal Hamiltonian circuit (OHC) in a tourist map. It has been proven to be an *NP*-hard problem because the number of the Hamiltonian circuits (HC) increases exponentially in proportion to the number of the cities in the map [1]. The TSP has been widely studied in the field of combinatorial optimization, graph theory and computer science due to its theoretical and practical values once it is resolved within a reasonable computation time.

The traditional graph search algorithms [2], linear programming [3] and dynamic programming [4] methods are exact methods to obtain the OHC. Some of them are efficient when the tourist map includes no more than a thousand cities. For the TSP with over a thousand cities, these exact methods must depend on the powerful computers or the computation time is too long. However, the research on the polynomial algorithms for the hard problem will still continue until P≠NP is verified in the future.

The approximate methods play an important role for the TSP although they do not guarantee to find the OHC. The advantage is that they can detect the *c*-optimal solutions (*c* is bigger than 1 for the MIN TSP and less than 1 for the MAX TSP) with the common computer in a polynomial computation time. The LK and LKH algorithms are taken as the most competitive algorithms for the TSP [5]. It is reported that the algorithms are robust to deal with a large TSP with thousands of cities, up to more than 3,000,000 cities [6]. However, the tours quality is hard to evaluate because the actual OHCs are usually not known for the complex TSP.

The work supported by the NSFC (Grant No.51205129).

Y. Wang is with the North China Electric Power University, Beijing, 102206, P. R. China (phone: 010-6177-2051; fax: 010-6177-2234; e-mail: wangyyong100@ 163.com).

The metaheuristic algorithms detect the optimal or near-optimal solutions according to the evolutionary rules [7], such as heuristic neural network [8], the genetic algorithms [9] and the consultant-guided search algorithm [10], etc. The approximate methods and the metaheuristic methods are combined together to enhance their performance with each other [4]. The metaheuristic methods show good performance, and they are apt to return the local and not the global optimal solutions. Therefore, improvements of these methods are always concentrated on.

Until 2005, researchers [11] claimed that $P = NP$ was still one of the great unanswered questions in mathematics. In order to find the OHC, the frequency graph is constructed according to the edge frequencies emulated from the local optimal Hamiltonian paths (LOHP). The LOHPs are derived from the weighted graph based on the four vertices and three lines inequality. The edge frequency is taken as the local heuristic information to generate the OHC. The search algorithm is designed to find the OHC based on the frequency graph.

II. The Optimal Hamiltonian Circuit and Local Optimal Hamiltonian Path

The graph G including n vertices is represented as G=<V, E>, where V=<v_1, v_2,…, v_n> are the vertex sets and E=<$e_{1\times2}$, $e_{1\times3}$,…, $e_{(n-1)\times n}$> are the edge sets. v_i ($1\le i\le n$) is the vertex and $e_{i\times j}$($1\le i, j\le n$) is the edge linking the two vertices v_i and v_j. The graph G is represented as the adjacent matrix A(G)={$a_{i\times j}$} ($1\le i, j\le n$), where $a_{i\times j}$=1 if (v_i, v_j)∈E(G), and v_i and v_j are adjacent in the graph G. Otherwise, $a_{i\times j}$=0. When the weights W=<$w_{1\times2}$, $w_{1\times3}$, …, $w_{(n-1)\times n}$> are assigned to the edges, the graph G becomes one weighted graph (WG). For the symmetrical TSP, $w_{i\times j}$ is equal to $w_{j\times i}$. The objective of TSP is to find the OHC from the WG in mathematics. For the MIN TSP, the length of the OHC is shortest among those of the HCs, and each vertex is contained in the HC once and only once. The HC wih n vertices is represented as HC^{n+1}=(v_1, v_2, v_3,…, v_i,…, v_n, v_1), and the two end vertices are identical. The superscript n+1 denotes the number of the vertices in the HC.

The HC is composed of the local Hamiltonian paths (LHP), and the OHC is composed of the local optimal Hamiltonian paths (LOHP). The LHP or LOHP containing i vertices is represented as LHP^i=(v_1, v_2, …, v_{i-1}, v_i) or $LOHP^i$=(v_1, v_2, …, v_{i-1}, v_i), where i indicates the number of the vertices in the LHP, v_1 and v_i are the end vertices, and the other vertices between v_1 and v_i are the middle vertices. There are no two identical vertices in the LHP or LOHP. For an arbitrary LOHP in the OHC, the orders of the vertices in the LOHP are determined. By extracting one LOHP from the OHC, the two end vertices in the

LOHP are concluded. Comparing with the other LHPs including the same vertices, the length of the LOHP is the minimum in the case that the two end vertices of these LHPs are identical.

For the symmetrical complete graph including n vertices, the number of the LOHP$^{\mathrm{i}}$s ($2\leq i\leq n$) is computed as the equation (1). If one end vertex is appointed, the number of LOHP$^{\mathrm{i}}$s is computed as the equation (2).

$$N_1^{\mathrm{LOHP^i}} = \frac{\mathrm{P_n^i}}{2\mathrm{P_{i-2}^{i-2}}} = \frac{i\times(i-1)}{2}\mathrm{C_n^i} = \frac{i\times(i-1)}{2}\binom{n}{i} \quad (1)$$

$$N_2^{\mathrm{LOHP^i}} = \frac{\mathrm{P_n^i}}{n\mathrm{P_{i-2}^{i-2}}} = \frac{i\times(i-1)}{n}\mathrm{C_n^i} = \frac{i\times(i-1)}{n}\binom{n}{i} \quad (2)$$

Where $\mathrm{P_n^i}$ is the number of permutations in the case that i vertices are selected from the n vertices, and $\mathrm{C_n^i}$ is the number of the combinations in the case that i vertices are selected from the n vertices. It is found that the number of LOHP$^{\mathrm{i}}$s is much smaller than the total number of LHP$^{\mathrm{i}}$s, $\mathrm{P_n^i} = \frac{n!}{(n-i)!}$. The number of LOHP$^{\mathrm{i}}$s changes as the binomial coefficient multiplied by a factor with the increment of number i. It increases at first, and then decreases. However, the number of the LHPs is always increasing as number i grows. For the symmetrical complete graph, the total number of the LOHP$^{\mathrm{i}}$s is computed using equation (3) with respect to equation (1).

$$N = \sum_{i=2}^{n} N_1^{\mathrm{LOHP^i}} = \sum_{i=2}^{n}\frac{\mathrm{P_n^i}}{2\mathrm{P_{i-2}^{i-2}}} = \frac{1}{2}\sum_{i=2}^{n}\left(i^2-i\right)\binom{n}{i} = n(n-1)2^{n-3} \quad (3)$$

As we see from equation (1), the number of LOHP$^{\mathrm{i}}$s increases not exponentially, but rather polynomially, in proportion to the scale of the TSP when i is far from n/2. On the other hand, the total number of LOHP$^{\mathrm{i}}$s increases exponentially in proportion to the number of the vertices in the WG with respect to the equation (3). The three equations are the basement to reduce the search space of the useful HCs in the complex WG.

III. The Four Vertices and Three Lines Inequality

In practice, the number of the LOHP$^{\mathrm{i}}$s is still large when the WG includes a lot of vertices and i is near to n/2. When i is small and far from n/2, the LOHP$^{\mathrm{i}}$s can be computed within polynomial computation time. When i is equal to 2, 3, all the LHP^{2}s and LHP^{3}s are LOHPs. There is no distinction between these LHP^{2}s or LHP^{3}s. When i is above 3, the number of the LOHP$^{\mathrm{i}}$s is smaller than that of the LHP$^{\mathrm{i}}$s for a simple WG. When i=4, the four vertices and three lines inequality (4) holds for all the LOHP^{4}s and it is convenient to compute the LOHP^4=(v_{i-1}, v_i, v_j, v_{j+1}) ($2\leq i\leq n$, $1\leq j\leq n-1$) in the WG.

$$l_{(i-1)\times i} + l_{i\times j} + l_{j\times(j+1)} \leq l_{(i-1)\times j} + l_{j\times i} + l_{i\times(j+1)} \quad (4)$$

Where $l_{i\times j}$ is the length of edge $e_{i\times j}$ between the vertices v_i and v_j ($2\leq i\leq n$, $1\leq j\leq n-1$).

The four vertices and three lines inequality meet the four point conditions summarized by Vladimir [12] for symmetrical TSP. It is the extension of one of the four point conditions, and it can be derived from the HCs instead of the original WG. The precondition is that there is at least one HC in the WG. It can be also applied to the asymmetrical TSP. The principle of the four vertices and three lines inequality is illustrated as Fig. 1.

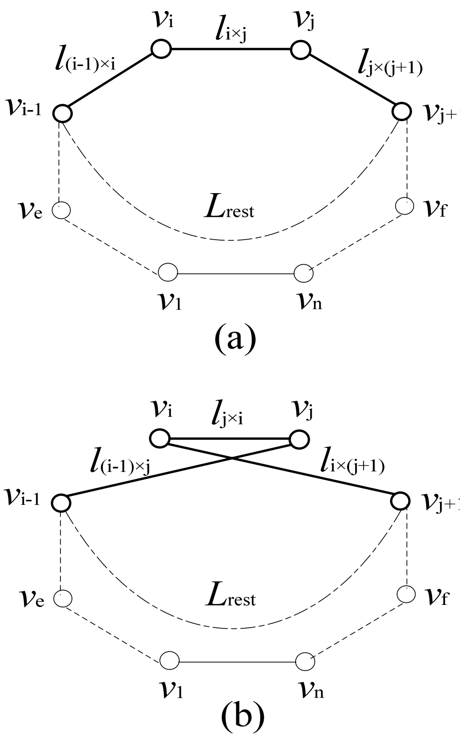

Fig. 1 The principle of four vertices and three lines inequality

Two HCs including n vertices are shown in Fig. 1 (a) and (b), and the HC in Fig. 1 (a) is the OHC. One of the LOHP^{4}s in Fig. 1 (a) is LOHP^4=(v_{i-1}, v_i, v_j, v_{j+1}) ($2\leq i\leq n$, $1\leq j\leq n-1$). The LHP^4 in Fig. 1 (b) includes the same four vertices v_{i-1}, v_j, v_i, v_{j+1} and it is noted as LHP^4=(v_{i-1}, v_j, v_i, v_{j+1}). Given the two rest LHP^{n-4}s are identical except the LOHP^4 and LHP^4 in Fig. 1 (a) and (b), the length of the two LHP^{n-4}s is the same and they are noted as L_{rest} in Fig. 1 (a) and (b). For the LOHP^4 and LHP^4, $l_{i\times j}$ is the length of the edge $e_{i\times j}$ linking the two vertices v_i and v_j. Two pairs of three edges $e_{(i-1)\times i}$, $e_{i\times j}$, $e_{j\times(j+1)}$ and $e_{(i-1)\times j}$, $e_{j\times i}$, $e_{i\times(j+1)}$ link the four vertices v_{i-1}, v_i, v_j, v_{j+1} in LOHP^4 and LHP^4. The length of the LOHP^4 is computed as $l_{(i-1)\times i}+l_{i\times j}+l_{j\times(j+1)}$ and the length of the LHP^4 is computed as $l_{(i-1)\times j}+ l_{j\times i}+l_{i\times(j+1)}$. Because the length of the OHC is shorter than or equal to the length of the HC, the four vertices and three lines inequality holds.

When i is big, the LOHP$^{\mathrm{i}}$s also can be computed using the similar inequalities. However, the number of the necessary inequalities is exponentially in proportion to the number of the vertices in the $LOHP^i$s.

IV. The Construction of the Frequency Graph

Because the number of $LOHP^i$s is equal to the binomial coefficient multiplied by a factor, it will consume a great deal of computation resources to generate the $LOHP^i$s once n is big. To reduce the computation complexity, three hypotheses are given to construct the frequency graph for the TSP.

(1) The longer $LOHP^j$s are composed of the shorter $LOHP^i$s, e.g. $i \le j$. In other words, the $LOHP^i$ is included in $LOHP^j$.

(2) The OHC exists in the WG, and it is composed of n distinct $LOHP^2$s (edges).

(3) The WG is a simple graph, e.g. two adjacent vertices v_i and v_j are connected with one edge $e_{i \times j}$.

The longer $LOHP^j$s are composed of the shorter $LOHP^i$s, and the OHC is composed of the LOHPs. If the longer $LOHP^j$s are obtained, the frequency of the shorter $LOHP^i$s can be counted from the longer $LOHP^j$s. The bigger j is, the more accurately the $LOHP^j$s approximate the OHC, and the bigger the frequencies of the $LOHP^i$s in the OHC are. When OHC^{n+1} is considered as the $LOHP^j$, the frequency of the $LOHP^i$ in it is equal to 1 and the frequency of the other LOHPs (and LHPs) is equal to zero. In view of hypothesis (2), the frequencies of the $LOHP^2$s (edges) should be computed from the $LOHP^j$s ($1 \le j \le n$). When the frequencies of the $LOHP^2$s are emulated from the $LOHP^j$s, the frequency graph is obtained. The $LOHP^2$s with high frequencies are considered as possible edges in the OHC, and the $LOHP^2$s with low frequencies are excluded from the consideration. The frequencies on the edges are the local heuristic information to connect the vertices into the OHC. The two vertices connected by the edges with high frequencies should be adjacent in the OHC or near OHCs. The more vertices the $LOHP^j$s have, the more appropriate the $LOHP^2$s with the high frequencies is in the OHC and the longer the computation time is used to generate the $LOHP^j$s. In reverse, the frequencies of the $LOHP^2$s in the OHC will not be high in the frequency graph and the computation time will be less. Therefore, when the frequencies of the $LOHP^2$s are emulated from the $LOHP^j$s including a smaller number of vertices, several $LOHP^2$s with higher frequencies connected to each vertex should be considered as candidates (nearest neighbors) for searching for the OHC.

The $LOHP^4$s are relatively easy to compute based on the four vertices and three lines inequality. The number of $LOHP^4$s is $\frac{n \times (n-1) \times (n-2) \times (n-3)}{4}$ for the symmetrical complete graph including n vertices. To reduce the computation time further, the m $LOHP^4$s containing each edge with the shortest length are computed. There are a total of $n\times(n+1)/2$ edges and the total number of the shortest $LOHP^4$s is $m\times n\times(n+1)/2$. The frequencies of the $LOHP^2$s (edges) are computed from the $m\times n\times(n+1)/2$ $LOHP^4$s. m is taken as the variable and its maximum value is $(n-2)\times(n-3)/2$. It is expected that the m value does not cause the OHC to be missed in the frequency graph.

TABLE I
THE SEARCH ALGORITHM BASED ON THE FREQUENCY GRAPH

```
Build one initial HC or near OHC with respect to the frequency graph.

While(the terminal condition is not met)
    While(current vertex from 1 to n)
        Generate the candidate vertices adjacent to the current
vertex considering the higher edge frequency between them
        Select the candidate vertices and substitute the adjacent
vertices of the current vertex in the near OHC
        Compute the length of the new near OHC and maintain the
shorter near OHC
    End
End search algorithm for TSP
```

V. The Search Algorithm Based on the Frequency Graph

Given the WG including n vertices, the $m\times n\times(n+1)/2$ $LOHP^4$s are computed, and the frequencies of the $n\times(n+1)/2$ $LOHP^2$s (edges) are calculated from the $LOHP^4$s, and the frequency graph is constructed. The search algorithm based on the frequency graph is designed as that in Table I. The initial HC or near OHC is searched for based on the frequency graph. From an arbitrary vertex, the next vertex is connected to the previous vertex considering the edge between them with the highest frequency until all the vertices are traversed.

There are two computation loops in the algorithm. In the inner computation loop, the vertices in the $LOHP^i$s ($1 \le i \le n-1$) in the near OHC can be substituted by the candidate vertices simultaneously. In this case, the computation time will be lengthened but the better solutions will be obtained. The maximum computation complexity of the inner computation loop is $O(n^3)$. The four vertices and three lines inequality can be applied to the searched near OHCs to generate shorter near OHCs in the computation process, and the convergence of the algorithm is accelerated.

The search algorithm aims to find a tour in which each vertex is connected to two vertices where the edges between them have high frequencies. The performance of the algorithm depends on the frequency graph. If the frequency graph is constructed with the longer LOHPs, the OHC will be found quickly. Otherwise, it will spend much time to find the OHC or near OHCs.

VI. The Illustrative Examples

TABLE II
THE COMPUTATION RESULTS BASED ON THE FREQUENCY GRAPH

Name	Number of cities	m	Computation results	Length of OHC	Computation time/ms
Berlin52	52	6	7544.366211	7542	3532
Eil76	76	6	544.369056	538	61141
Kora100	100	6	21285.437500	21282	9085
Tsp225	225	6	3920.518757	3919	62266

The TSP instances are downloaded from the website: www2.iwr.uni-heidelberg.de/groups/comopt/software /TSPLIB95/tsp/. The method is coded with the C++ language and runs on a Lenovo computer with a 2.3GHz processor and 2G of inner memory. The computation cycles are taken as the terminal condition. The maximum computation cycle is set as the scale of the TSP. m is set as 6 for generating the LOHP^{4}s. The computation results and computation time is shown in Table II. Because floating numbers are used in the computation process, the results are always bigger than the length of the given OHC. The maximum computation complexity of the algorithm is $O(n^4)$. Therefore, the computation time is longer than the LKH algorithm.

The OHCs of the TSP instances are found based on the frequency graph. In our experiments, the OHCs of some TSP instances are not found in one experiment, but they can be found in several trials. The change process of the HCs for Berlin52 is shown in Figure 2. Before the 40th computation cycle, the OHC is found. In view of Figure 2, the length of HCs decreases in great scope in the former computation cycles, which illuminates that the algorithm has a rapid convergence rate.

It also found that the OHC is not unique for some particular TSP instances. Though the orders of the vertices are different, the length of the OHCs is nearly the same. It is simple to generate the frequency graph with the LOHP^{4}s. On the other hand, the frequency graph will be more accurate for detecting the OHC if the LOHPs including more vertices are utilized to generate the frequency graph.

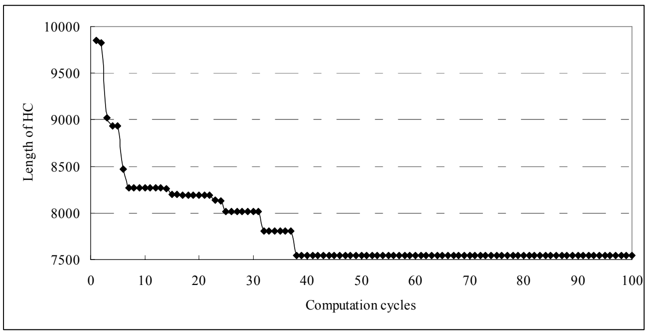

Fig. 2 The change processes of the HCs

VII. Conclusion

The number of the LOHPs is computed, which increases as the binomial coefficient multiplied by a factor. In principle, all the LOHPs can be computed in a polynomial computation time once the scale of TSP is determined. The LOHPs with a smaller number of vertices are easy computed, especially for the LOHP^{4}s based on the four vertices and three lines inequality. The OHC is composed of the LOHPs and the longer LOHPs are composed of the shorter LOHPs. The frequencies of the shorter LOHPs in the OHC are high when they are emulated from the longer LOHPs. The frequency graph is constructed with the frequencies of the LOHP^{2}s computed from the longer LOHPs. The algorithm nearly always finds the OHC based on the frequency graph. The longer the LOHPs are used to construct the frequency graph, the more accurate the frequency graph is at finding the OHC.

The future work is to improve the structure and performance of the algorithm based on the frequency graph.

Acknowledgment

The authors acknowledge Emily Moder for checking the paper. The work benefits from the facilities of National Key Laboratory of New Energy Power System and the Beijing Key Laboratory of New and Renewable Energy, North China Electric Power University, Beijing, China.

References

[1] A. Rodríguez, and R. Ruiz, "The effect of the asymmetry of road transportation networks on the traveling salesman problem," *Computers & Operations Research*, Vol.39, pp.1566-1576, 2012.

[2] B.W. Douglas, *Introduction to graph theory, Section Edition*. Beijing: Pearson Education Asia Limited and China Machine Press, 2006, pp.75-76,

[3] L. Gouveia, S. Voβ, "A classification of formulations for the (time-dependent) traveling salesman problem," *European Journal of Operational Research*, Vol.83, pp.69-82, 1995.

[4] B. Bontoux, C. Artigues and D. Feillet, "A Memetic Algorithm with a large neighborhood crossover operator for the Generalized Traveling Salesman Problem," *Computers & Operations Research*, Vol.37, pp.1844-1852, 2010.

[5] K. Helsgaun, "An effective implementation of the Lin-Kernighan traveling salesman heuristic," Available: www2.iwr.uni-heidelberg.de/groups/comopt/ software/TSPLIB95/tsp/, May, 2012.

[6] D. S. Johnson, L. A. McGeoch, "The Traveling Salesman Problem and Its Variations," *Combinatorial Optimization*, Vol.12, pp. 445-487, 2004.

[7] P.M. Binder, "Frustration in Complexity," *Science*, Vol.320, pp.322, APRIL 2008.

[8] H. Ghaziri, and I. H. Osman, "A neural network algorithm for the traveling salesman problem with backhauls," *Computers & Industrial Engineering*, Vol.44, pp.267-281, 2003.

[9] Y. H. Liu, "Different initial solution generators in genetic algorithms for solving the probabilistic traveling salesman problem," *Applied Mathematics and Computation*, Vol.216, pp.125-137, 2010.

[10] S. Iordache, "Consultant-Guided Search-A New Metaheuristic for Combinatorial Optimization Problems," *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation* (*GECCO'10*), Portland, Oregon, July 7-11, 2010, pp.225-232.

[11] C. Seife, "What Are the Limits of Conventional Computing," *Science*, Vol.309, pp.96, JULY 2005.

[12] V. Deineko, B. Klinz, G. Woeginger, "Four point conditions and exponential neighborhoods for symmetric TSP," *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm* (*SODA 2006*), Miami, January 22-26, 2006, pp.544-553.

Y. Wang was born in Henan Province, P.R.China, 28, February, 1978. He got the Bachelor degree in Manufacturing Engineering of Aircraft at the Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangshu Province, P.R.China, in Jun. 2001. He got the Master degree in the field of CAD&CAM at the Bei Hang University, Beijing, P.R.China, in Apr. 2004, and he got the Doctor degree in the field of CAD&CAM at the Bei Hang University, Beijing, P.R.China, in Dec. 2008.

He now works as a lecturer in the School of Renewable Energy, North China Electric Power University, Beijing, P.R.China. He has published more than three research articles on assembly sequence planning which is one of the combinatorial optimization problems as the Travelling Salesman Problem. His research interests includes combinatorial mathematics, graph theory, meta-heuristic algorithms and their applications, such as network optimization, the assembly sequence planning, machine path optimizations, etc.