A Method to Search the Optimal Hamiltonian Cycle with a Set of Approximations for Travelling Salesman Problem

Y. Wang
School of Renewable Energy
North China Electric Power University
Beijing 102206, China

Abstract—The objective of travelling salesman problem (TSP) is to find the optimal Hamiltonian cycle (OHC) and it has been proven to be NP-complete. A heuristic method is proposed for TSP. It is realized with four steps. A set of approximations are computed with the 2-opt move algorithm firstly. Secondly, a frequency graph is computed with the set of approximations. A sparse graph with small number of edges is generated in the third step. At last, the depth-first graph search algorithm is used to find the OHC or a better approximation with the sparse graph. The experimental results show that the OHC of most of the TSP instances are searched within an acceptable computation time.

Keywords—travelling salesman problem; optimal hamiltonian cycle; frequency graph; sparse graph; depth-first graph search algorithm

I INTRODUCTION

As a well-known NP-complete problem [1], travelling salesman problem (TSP) has been widely studied in the fields of combinatorial optimization. Given a tourist map with n cities, a salesman expects to find a cycle that visits each city once and exactly once and incurs the least distance, time or cost, etc. In a weighted graph (WG), the distance, time or cost between cities is generally taken as weights. The cycle with the minimal weights is the optimal Hamiltonian cycle (OHC). For the symmetrical TSP with n cities, the number of total Hamiltonian cycles (HCS) is (n-1)!/2. It will consume a lot of computation resource for searching the OHC once n is big. In practice, many complex industrial engineering problems, such as routing problems [2], manufacturing planning [3] etc., can be converted into TSP models and the efficient methods are advocated to resolve them within a reasonable computation time.

The recent researches on the exact algorithms, such as the improved cutting plane method [4], branch and bound [5] and branch and cut method [6], illustrate their good performance for the medium size of TSP. For large scale of TSP, the computation time is too long. The experiments also show that the performance of these exact algorithms is hard to improve [7] for large TSP. If we add more constraints, the complexity of TSP would be decreased. However, the number of constraints also increases in proportion to the factorial of the number of cities.

Though the OHC is not guaranteed, most of the approximate methods are efficient for large scale of TSP. An approximation can be found within a polynomial computation time. The nearest neighbours [8], the subset cover methods [9] are efficient whereas the results usually have a relatively big gap with the OHC. The LK or LKH algorithm [10] is taken as one of the competitive approximate algorithms for TSP. The experiments show that the time complexity of LKH is O(n^2) and it always finds an approximate solution near to the OHC. The disadvantage of the approximate algorithms is that they will trap into the local minima. For large scale of TSP, the cycle quality cannot be evaluated due to the lack of the best solutions.

The current best approximate algorithms can find the approximations near to the optimum (the error within 2%) in an acceptable computation time [1]. An approximate method is proposed to detect the OHC or a better approximation based on the approximations. Four steps are necessary. Firstly, a set of approximations are computed with the 2-opt move algorithm [11]. Secondly, the frequencies of the edges are emulated from the set of the approximations and a frequency graph (FG) is computed. Thirdly, a sparse graph (SG) is derived from the FG. The SG includes a small number of edges and the complexity of TSP will be reduced. At last, the depth-first graph search algorithm is used to find the OHC or a better approximation with the SG.

The rest of the paper is organized as follows. Section 2 gives the theoretical foundation of the method. Section 3 introduces the implementation of the method. The experiments and analysis are discussed in section 4. The conclusions and possible future research are given in section 5.

II THEORETICAL FOUNDATION

TSP is often represented as a complete WG. The weight of the OHC is minimal. For TSP with n cities, it is represented as G=[V, E, W], where V(G)=\{v_1, v_2, ..., v_n\} are the vertices sets, E(G)=\{e_{ij}\}_1(1\leq i,j\leq n) are the edges sets and W(G)=\{w_{ij}\}_1(1\leq i,j\leq n) are the weights on the edges, v_i (1\leq i\leq n) is the vertex and e_{ij} (1\leq i,j\leq n) is the edge linking the two vertices v_i and v_j. The weight w_{ij} on the edge e_{ij} is often taken as distance, cost, time etc. The topological relationships between two vertices is represented as an
adjacent matrix \( A(G) = [a_{ij}]_{n \times n} \), where \( a_{ij} = 1 \) if \( (v_i, v_j) \in E(G) \) and \( v_i \) and \( v_j \) are adjacent. Otherwise, \( a_{ij} = 0 \). In general, \( a_{ij} = 1 \) (if \( i \neq j \)) and \( a_{ii} = 0 \) (if \( i = j \)) for TSP.

An HC is a cycle that visits each vertex once and exactly once. For TSP, the objective is to find the HC with the minimal weight, i.e., OHC. An HC with \( n \) vertices is represented as \( H C^{n-1}(v_1, v_2, \ldots, v_n) \). It includes \( n \) different vertices and its two end vertices are identical. Every HC is composed of the paths which are called the local paths (LP). Given an LP with \( i \) (2 \( \leq i \leq n \)) vertices, it is represented as \( L P_i = (v_1, v_2, \ldots, v_i) \). The superscript \( i \) indicates the number of the vertices in the LP. The vertices \( v_1 \) and \( v_i \) on both ends are two end vertices. The other vertices between vertices \( v_1 \) and \( v_i \) are the middle vertices. Especially, if \( i = 2 \), \( L P_2 = (v_1, v_2) \) represents an edge. For the symmetrical TSP, LP is equal to its reversion, e.g., \( L P_i = (v_1, v_2, v_3, \ldots, v_i) \) for TSP.

Similarly, the OHC is composed of a set of LPs with small weights. These kinds of LPs are called the local optimal paths (LOP). For an arbitrary LOP in the OHC, it is shorter than that of the other LPs with the same vertices in case that their two end vertices are identical. With the description of LOPs, the LPs are classified into the LOPs and the non-LOPs whose weights will become smaller when the middle vertices are exchanged. The number of the LPs for symmetrical TSP is computed as formula (1) whereas the number of the LOPs is computed as formula (2) [12].

\[
N_{LHP}^i = \frac{i!}{2} C_n^i
\]

\[
N_{LOP}^i = \frac{i \times (i - 1)}{2} C_n^i
\]

Where \( C_n^i \) means the number of the combinations in case that \( i \) vertices are selected from \( n \) vertices. The number of the LPs is \((i-2)! \) times the number of the LOPs. Most of the LPs are the non-LOPs. When the OHC are viewed as the combinations of the LOPs, the search space of the OHC will become \( O(n^{2n}) \) [12] which is much smaller than \((n-1)!/2\). On the other hand, it is still hard to compute the OHC by emulating all of the LPs for large scale of TSP.

The number of the LOPs is very big whereas a small number of them belong to the OHC. The LOPs with more vertices are composed of the LOPs (\( \varepsilon \) is \( \leq 2 \)) with fewer vertices. Not all of the LOPs belong to the LOPs. If a set of the LOPs are obtained, the frequencies of the LOPs can be emulated from them. The nearer these LOPs approach to the OHC, the bigger the frequencies of the LOPs in the OHC will be. The nearness between the LOPs and the OHC implies they have more intersections, i.e., LOPs, which results in the frequencies of the LOPs in the OHC will be much bigger than that of the other LOPs and non-LOPs.

The LOPs (\( \varepsilon = 2 \)) and the OHC are composed of the edges (LOPns) and the frequencies of edges are more concerned. When the frequencies of the edges are emulated from a set of LOPs, the frequency graph is obtained [12]. If a set of LOPs are used to compute the FG, the values on the edges are their frequencies computed from the set of LOPs. If the set of LOPs approach to the OHC, the edges with big frequencies can be taken as the candidate edges in the OHC.

The LOPns are near to the OHC and there are total \( n^{n(n-1)/2} \) LOPns in view of formula (2). Given there is one OHC in the WG, \( n \) LOPns will belong to the OHC. The LOPns are hard to compute and the approximate LOPns can be computed with the approximate algorithms. These approximate LOPns are taken as the approximations to compute the FG. Though these approximations are different from the OHC, they are nearer to the OHC than the other general HCs do. The 2-opt move algorithm is used to compute the \( n^{n(n-1)/2} \) approximate LOPns. When their two end vertices are connected, they are taken as the approximations. After the \( n^{n(n-1)/2} \) approximations are computed, they are ordered from small to big weights. To reduce the complexity, the previous \( m \) (\( 1 \leq m \leq n \)) approximations with small weights are selected to compute the FGs. These approximations approach to the OHC than the other approximations do in weights. When the FG is computed, the \( k \times n \) edges with big frequencies are used to combine a SG, where \( k \) is a small coefficient. Then the depth-first graph search algorithm is used to search the OHC or a better approximation in the SG. Eppstein [13] gave an algorithm which resolved the 3-regular TSP in \( O(1.260^{n}) \) time. To avoid the complexity of graph search algorithm, \( k \) is assigned as a value less than 1.5 to guarantee that the average degree of the vertices is less than 3.

### III THE IMPLEMENTATION OF THE METHOD

The implementation of the method includes four steps. The \( n^{n(n-1)/2} \) approximations are computed with the 2-opt move algorithm in the first step. Secondly, the \( n^{n(n-1)/2} \) approximations are ordered from small to big weights and the previous \( m \) (\( 1 \leq m \leq n \)) approximations with the minimal weights are used to compute an FG. Thirdly, the edges are ordered from big to small frequencies and the \( k \times n (k \leq 1.5) \) edges with the biggest frequencies are chosen to combine a SG. At last, the depth-first graph search algorithm is used to search the OHC or better approximation in the SG.

It notes that not all of the SGs include the OHC. It depends on two main factors. The first factor is the quality of the approximations. When they have many intersections with the OHC, the frequencies of the edges in the OHC will be much bigger than that of the other edges. The second factor is the variable \( m \). A different FG will be computed if the value \( m \) is changed. The frequencies of the edges will change according to \( m \). In some FGs, the frequencies of some edges in the OHC are not big enough and they will not be selected to form a SG with the OHC. The approximations rely on the 2-opt move algorithm. Not all of the approximations are ensured to approach to the OHC as near as possible. In our experiments, \( n \) FGs are computed according to \( m \) (\( 1 \leq m \leq n \)) approximations. For example, the first FG is computed with the first approximation with the minimal weight and the last FG is computed with the previous \( n \) approximations. For each FG, the \( k \times n (k \leq 1.5) \) edges with the biggest frequencies are selected to compose a SG. The graph search algorithm is used to search the OHC in the \( n \) SGs. The OHC will be found once a SG includes the
The 2-opt move algorithm is first proposed by Croes [11] and it is efficient to find an approximation. Given an initial HC, the local heuristic is executed by exchanging two selected edges with the other two edges until an approximation is generated. An algorithm is designed to compute the \(n \times (n-1)/2\) approximate LOP's and it is given in Figure 1. The maximal computation time complexity is approximately as \(O(N_{max} \times n^2)\), where \(N_{max}\) is the maximal computation cycle. These LOP's are converted into the approximations after their two end vertices are connected.

Given the maximal computation cycle \(N_{max}\) as the terminal condition.

\[
\text{For } (i = 1 \to n) \\
\text{Generate a } LP^i \text{ with two end vertices } v_i \text{ and } v_j \text{ at random.} \\
\text{While (computation cycle less than } N_{max}) \\
\text{ } \\
\text{Obtain an approximate } LOP^i \text{ with two end vertices } v_i \text{ and } v_j. \\
\text{End} \\
\text{End} \\
\text{Obtain the } n \times (n-1)/2 \text{ approximate } LOP^n's \text{ and change them into the approximations.} \\
\]

**FIGURE I. THE 2-OPT MOVE ALGORITHM TO COMPUTE THE N\times(N-1)/2 APPROXIMATE LOPNS.**

After the \(n \times (n-1)/2\) approximations are computed, an arbitrary set of approximations can be used to compute an FG. To reduce the disadvantages of the other approximations, the \(n\) approximations with the minimal weights are used to compute the FGs. The \(n \times (n-1)/2\) approximations are ordered from small to big weights. The \(m(1 \leq m \leq n)\) OHCs are used to compute the FG and total \(n\) FGs are computed.

After the FG is computed, the \(k \times n\) edges with the biggest frequencies will be chosen and a SG is composed of them. If the set of the approximations are useful, the SG will contain the OHC and the complexity of TSP will be reduced greatly. At last, the depth-first graph search algorithm is used to search the OHC or better approximation. The pseudo codes to find the OHC or better approximation is designed and shown in Figure 2.

**TABLE I. THE COMPUTATION RESULTS.**

<table>
<thead>
<tr>
<th>Instances</th>
<th>City Num.</th>
<th>(N_{max})</th>
<th>2-opt move</th>
<th>(k)</th>
<th>Results</th>
<th>Given OHC</th>
<th>TC/ms</th>
<th>AC/ms</th>
<th>Concorde/ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eil51</td>
<td>51</td>
<td>100</td>
<td>435.67</td>
<td>1.3</td>
<td>428.98</td>
<td>426</td>
<td>2652</td>
<td>52</td>
<td>120</td>
</tr>
<tr>
<td>Berlin52</td>
<td>52</td>
<td>100</td>
<td>7732.20</td>
<td>1.2</td>
<td>7544.36</td>
<td>7542</td>
<td>1498</td>
<td>28.81</td>
<td>130</td>
</tr>
<tr>
<td>Si70</td>
<td>70</td>
<td>100</td>
<td>684.85</td>
<td>1.2</td>
<td>677.11</td>
<td>675</td>
<td>15039</td>
<td>214.84</td>
<td>200</td>
</tr>
<tr>
<td>Eil76</td>
<td>76</td>
<td>100</td>
<td>547.30</td>
<td>1.2</td>
<td>544.37</td>
<td>545</td>
<td>52042</td>
<td>684.76</td>
<td>110</td>
</tr>
<tr>
<td>Pr76</td>
<td>76</td>
<td>100</td>
<td>108358.33</td>
<td>1.2</td>
<td>108159.43</td>
<td>108159</td>
<td>46628</td>
<td>613.53</td>
<td>600</td>
</tr>
<tr>
<td>Rat99</td>
<td>99</td>
<td>(n^2)</td>
<td>1219.24</td>
<td>1.1</td>
<td>1219.24</td>
<td>1211</td>
<td>4742</td>
<td>47.90</td>
<td>400</td>
</tr>
<tr>
<td>KroA100</td>
<td>100</td>
<td>(n^2)</td>
<td>21294.40</td>
<td>1.1</td>
<td>21285.44</td>
<td>21282</td>
<td>3869</td>
<td>38.69</td>
<td>310</td>
</tr>
<tr>
<td>KroA100</td>
<td>100</td>
<td>(n^2)</td>
<td>21460.99</td>
<td>1.1</td>
<td>21294.29</td>
<td>21294</td>
<td>601789</td>
<td>6017.89</td>
<td>330</td>
</tr>
<tr>
<td>Lin105</td>
<td>105</td>
<td>(n^2)</td>
<td>14449.60</td>
<td>1.1</td>
<td>14382.99</td>
<td>14379</td>
<td>11840</td>
<td>112.76</td>
<td>220</td>
</tr>
</tbody>
</table>

In view of the results, the OHC are found within an acceptable computation time. Firstly, the frequencies of edges illustrate the number of intersected edges between the set of approximations and the OHC. The OHC exists in the SG composed of the \(k \times n\) edges with the biggest frequencies. We conclude that an approximation generally has more intersections with the OHC than it does with the other HCs. When these approximations are converted into the FG, the

**FIGURE II. THE PSEUDO CODES TO SEARCH THE OHC OR BETTER APPROXIMATION.**

EXAMPLES ANALYSIS AND DISCUSSION

The method is coded with C++ language and runs on the Lenovo computer with a 2.4GHz processor and 512MByte RAM. The Euclidean TSP is used to test the method. The Euclidean TSP instances are downloaded from the website: www2.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/. The results are shown in Table 1. The approximation with the minimal weights searched by the 2-opt move algorithm is also shown in Table 1. The real values of coordinates are used to compute the distance between cities and the length of the approximations. The computation results may be a little bigger than the given OHC computed with integers [10]. The total computation time (TC) of the graph search algorithm is recorded and shown in Table 1. It is the total computation time to traverse the \(n\) SGs. The average computation time (AC) to traverse each SG is computed as TC/n and shown in Table 1. For comparisons, the computation time of Concorde (03.12.19) for these TSP instances is given in the last column. The Concorde package is carried out on a processor of a dual-processor 2.8 GHz Intel Xeon PC with a 533 MHz front-side-bus and 2 GByte RAM. (http://www.tsp.gatech.edu/concorde/benchmarks/bench.html)
frequencies of the edges in the OHC are much bigger than those of the other edges. Secondly, the computation time of the graph search algorithms is greatly reduced with the SG. It is impossible to find the OHC with the depth-first graph search algorithm for these TSP instances whereas it is feasible to find the OHC based on the SGs. The total computation time to search the n SGs is acceptable and the average computation time to traverse one SG is less than 1s for most of these examples. If the efficient algorithms, e.g. the LKH algorithm and the Concorde package, are used to search the OHC based on the SGs, the computation time will be much less.

When the WG is transformed into the FG with a set of approximations, the frequencies on the edges are more useful than the weights to find the OHC. The nearest neighbours related to the weights are usually used to find an approximation and they cannot guarantee to find a good approximation [14]. If the frequencies of edges are taken as the heuristics to choose the nearest neighbours, the complexity of TSP will be largely reduced. With the SGs, The OHC or better approximation is found with the graph search algorithms within an acceptable computation time.

Two points must be mentioned with the method. First, the set of the approximations plays an important role to ensure the frequencies of the edges in the OHC are much bigger than those of the other edges. The nearer the set of approximations approach to the OHC, the more suitable approximation 

experiments show that the computation time of the graph search algorithm is acute to the number of the edges in the SG. With the addition of a few edges, more computation time will be demanded.

In the future, the research will focus on two aspects. (1) The computation of a useful SG for TSP. (2) The large scale of TSP will be used to test the method.

ACKNOWLEDGEMENTS

The project supported by NSFC (Grant No.51205129). 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