A quick Heuristic and a general search algorithm for traveling salesman problem

Chao Wang1,*, Deguang Wang2, and Chun Jin3
1 School of Software, Dalian University of Foreign Languages, Dalian 116044, China
2 School of Software, Dalian Jiaotong University, Dalian 116052, China
3 Institute of Systems Engineering, Dalian University of Technology, Dalian 116024, China

Keywords: Traveling salesman problem, Greedy heuristic, 2-opt operation, Simulate annealing.

Abstract. This paper puts forward a constructive heuristic algorithm called the method of inserting the minimum neighbor edge from outside to the center (IMNEFOTC) that can be applied to solve large-scale and ultra-large-scale travelling salesman problems. Through it and the randomized greedy heuristic algorithm (RGH) which greedy heuristic algorithm is modified, a general meta-heuristic search algorithm framework is built. The general search algorithm (GSA) is based on a set of initial solutions, and continuous 2-opt operations are performed, so as to search for solutions in better quality. The data from the experiment reveals that the performance of IMNEFOTC is better than the traditional greedy algorithm, and the GSA can obtain a pretty satisfactory solution in a reasonable range of time.

1 Introduction

The traveling salesman problem (TSP) is to find a minimum Hamiltonian circle which visits all vertices in a given set. It is widely applied to various fields such as transportation, logistics, circuit design, robot control, and so on. TSP is a classic NP difficult problem in the field of combination optimization, on the basis of which other problems, such as colored TSP, multiple TSP, vehicle routing problems and other related problems have emerged and developed rapidly.

Given a complete undirected graph $G=(V, E)$, where $V=\{v_1, v_2, ..., v_n\}$ is a set with $n$ vertices, and $E$ is a set with $n(n-1)/2$ edges. Every vertex's position is denoted by the 2-dimensional coordinate $v_i(x_i, y_i)$, $d(i, j)$ is the Euclidean distance between vertex $v_i$ and vertex $v_j$ or the cost corresponding to the edge $(v_i, v_j)$ in $E$. For any three vertices $u, v, w$ in $V$, *

* Corresponding author: 13130001974@163.com

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (http://creativecommons.org/licenses/by/4.0/).
the inequality relation, $d(u, w) \leq d(u, v) + d(v, w)$ should be satisfied. Suppose a tour $T=(t_1, t_2, \ldots, t_n)$, where $t_i \in V$, if $i \neq j$ then $t_i \neq t_j$, then $t_{n+1}=t_1$, then

The mathematical model of TSP is given by:

$$\min \sum_{i=1}^{n} d(t_i, t_{i+1}), \quad (i = 1, 2, \ldots, n) \tag{1}$$

The total distance of a tour is denoted by:

$$f(T) = \sum_{i=1}^{n} d(t_i, t_{i+1}), \quad (i = 1, 2, \ldots, n) \tag{2}$$

The algorithms for TSP are divided into three categories: precision algorithms, heuristic algorithms, modern heuristic intelligent algorithms. The precise algorithms include branch and bound and cutting plane, the common heuristics are greedy, nearest neighbor, 2-opt, 3-opt[1] and Lin Kernighan[2] etc., and the intelligent algorithms usually include tabu search (TS), simulated annealing (SA), genetic algorithms (GA), particle swarm optimization[3], ant colony algorithms, bat algorithms etc. The time complexity of most precision algorithms increases exponentially with the scale of the problem, so large-scale TSP problems cannot be solved. Heuristic algorithm makes the solving process fall into local optimal search, and it is not easy to get satisfactory solution of good quality. Most of the intelligent algorithms simulate the biological features of nature and adopt the behavior evolution mode of the population. In addition, the solution of multi-peak problems converted into multi-target optimization problems [4] can also be tried and applied in TSP. The algorithm with better effect is the hybrid algorithm combining intelligent algorithm with the neighbor search algorithm.

This paper proposed a quick heuristic algorithm which is the method of inserting the minimum neighbor edge from outside to the center (IMNEFOTC) and modified the traditional greedy heuristic algorithm to generate multiple approximate optimal solution, and on this basis a general meta-heuristic search algorithm framework (GSA) was built for TSP. The algorithms of IMNEFOTC and GSA are very suitable for large or super large scale problems, and the GSA algorithm can further improve the quality of the solution. GSA starts at a population comprising of various initial solutions, then every solution is improved quickly by 2-opt operator successively. When distance between two adjacent generations of an individual is close enough, global operators and local operators are alternately executed until the maximum iterations. Finally, the algorithm outputs the best solution during the process of the iterative computing.

The remainder of this paper is organized as follows: In section 2, two quick heuristics and the GSA are described in detail. Experimental results are provided in section 3, and some conclusions are drawn in 4 section of this paper.

### 2 The two quick algorithms and the general search algorithm framework

#### 2.1 The method of inserting minimum neighbor edge from outside to the center (IMNEFOTC)

IMNEFOTC assumes that all vertices are uniformly distributed in a circle or approximate circular region, and the minimum adjacent edge is inserted from the outer point of the region. This process approaches to the inner center of the circle until $n-1$ edges have been inserted. The time complexity of the algorithm is $\Theta(n^2)$.
The steps of IMNEFOTC are presented as following:

Input: Points\((x_i, y_i), \text{i}=1,2,...,n\).

**Step 1:** Initialize the set of edges which constructs Hamiltonian cycle, \(H=\emptyset\), and the degrees of all vertices are equal to 0, \(\text{deg}_i=0, \text{i}=1,2,...,n\). Assign each vertex a distinct connected component identifier, \(\text{comp}_i=i\).

**Step 2:** Compute the virtual central with the average coordinate value of all vertices, \(\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}\).

**Step 3:** Calculate the distance from each vertex to the virtual center and store them in the array \(\text{disToCenter}()\). \(\text{disToCenter}(i) = \sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}, \text{i}=1,2,...,n\).

**Step 4:** Sort the array \(\text{disToCenter}(1\text{ to }n)\) in ascending order, and store the index numbers of the corresponding vertices to another array \(\text{Points2}(1\text{ to }n)\).

**Step 5:** Set a pointer, and point to the last element of the array \(\text{Points2()}\), let the current vertex \(k_i = \text{Points2}(n)\).

**Step 6:** If the cardinality of \(H\), \(|H| < n-1\), then do next. Otherwise, turn to step 10.

**Step 7:** Insert the minimum adjacent edge of \(k_i\), which is edge \((Q \rightarrow W)\) into set \(H\) without generating any sub-loop and a degree-3 vertex. Vertex \(Q\) and \(W\) are the two endpoints of the edge.

**Step 8:** The degrees of vertex \(Q\) and \(W\) increase 1 respectively. Dispose the connected component identifiers so that the vertices’ identifiers of each connected component are identical and different from other connected components’.

**Step 9:** When the degree of \(k_i\) is equal to 2, the pointer slides forward to the nearest vertex whose degree is less than 2. Let the current vertex \(k_i\) be the vertex which the current pointer is pointing to. Turn to Step 6.

**Step 10:** Connect two vertices with degree of 1, and add the \(n^{th}\) edge to \(H\).

Output: All edges of \(H\)

### 2.2 Randomized greedy heuristic

The idea of randomized greedy heuristic (RGH) is based on classic greedy heuristic, which generates multiple better initial solutions. Its steps are presented as following:

Input: a complete graph \(G=(V,E)\) with \(n\) vertices, and a random integer \(\beta(1 \leq \beta \ll n)\).

Step 1: Initialize the set of edges which construct a Hamilton circle, \(R=\emptyset\), and the degrees of all vertices are initialized to 0.

Step 2: Sort all edges of \(G\) in non-descending order according to their weights to the set \(E'\), \(\{e_1',e_2',...,e_{n(n-1)/2}\}'\). Let the current edge \(ce = e_1'\).

Step 3: If \(|R|=n-1\), then go to step 5. Otherwise, generate a random integer \(\xi \in [1,\beta]\), and do next.

Step 4: Take the edge \(e'_j\) which is the \(\xi^{th}\) edge from the neighboring edges after the current edge. The two vertices of the edge \(e'_j\) are \(u\) and \(w\), if the degrees of \(u\) and \(w\) are less than or equal to 1, and adding \(e'_j\) to \(R\) do not form any sub-circuit, then \(R \leftarrow e'_j\) and add 1 to the degrees of \(u\) and \(w\), \(ce = e'_j\), turn to step 3. Otherwise \(\xi \leftarrow \xi + 1\), and go to step 4 again.

Step 5: Connect the two vertices with degrees less than 2, and a Hamilton circle is constructed.

Output: All edges of \(R\).
2.3 A general search algorithm (GSA) for TSP

For a TSP with \( n \) cities, GSA starts at a population that consists of various initial solutions \( T_i = (t_{i1}, t_{i2}, \cdots, t_{in}) \), \( i = 1, 2, \cdots, \text{size} \). These initial solutions are generated by six different quick generation algorithms. The population evolves by alternately executing the operators of local search and global search. The fitness function of \( T_i \) is denoted by \( f(T_i) \). The best individual and its fitness which the algorithm finds are saved to \( P_g = (p_{g1}, p_{g2}, \cdots, p_{gn}) \) and \( f(P_g) \).

2.3.1 The algorithm framework of GSA

Step 1: Construct \( \text{size} \) initial solutions \( T_i (i = 1, 2, \cdots, \text{size}) \) by six different quick generation algorithms which are IMNEFOTC, RGH, greedy heuristic, nearest neighbor, modified minimum spanning tree and random generation. \( \text{size} \geq 6, \) let \( \lambda = 0 \).

Step 2: Calculate \( f(T_i) \) for every \( T_i \), and save the best individual to \( P_g \).

Step 3: If \( \lambda \geq \lambda_{\text{max}} \), putout \( P_g \) and \( f(P_g) \), the algorithm is over. Otherwise do next step.

Step 4: Do local search operator to every individual \( T_i^\lambda \) to generate corresponding new individual \( T_i^{\lambda+1} \), and calculate \( f(T_i^{\lambda+1}) \), update \( P_g \), \( \lambda = \lambda + 1 \).

Step 5: Judge the convergence condition, if some individuals meet the convergent criterion then go to step 6, otherwise turn to step 3.

Step 6: Do crossover operator to the individuals \( T_i^\lambda \) which have been convergent, generate \( T_i^{\lambda+1} \), calculate \( f(T_i^{\lambda+1}) \), and update \( P_g \).

Step 7: Do the same as step 6, but replace the crossover operator with the operator of crossover and reverse.

Step 8: \( \lambda = \lambda + 1 \), and turn to step 3.

2.3.2 Implementation details of the GSA

(1) Generation of initial solutions

The steps of the algorithms of IMNEFOTC and RGH have been described in detail in section 2.1 and section 2.2. The other four generations of initial solutions are summarized as follows.

The randomized generation algorithm produces a random permutation from 1 to \( n \). Greedy heuristic starts at the shortest edge, and repeatedly inserting the shortest remaining available edge which does not create a degree-3 vertex or a cycle of length less than \( n \), until the \((n-1)^{th}\) edge are added. Greedy heuristic can be implemented in time \( \Theta(n^2\log n) \) [1]. Generation by modified minimum spanning tree first build the minimum spanning tree by Prime algorithm. Then a vertices' sequence can be got by visiting the tree in preorder walk, which is also an approximate TSP tour. The time complexity of Prime algorithm can be improved to \( \Theta(|E|+n\log n) \). Nearest neighbor starts from a vertex, and always go next to the nearest as-yet-unvisited location until all vertices are visited. The running time of nearest neighbor is \( \Theta(n^2) \).

(2) Local search operator

The local search operator of GSA continuously executes 2-opt operations \( fn \) times, \( n \) is the scale of TSP, \( f \) is the coefficient. The times of the consecutive execution depend on the scale of the problem. The large the scale of TSP is, the more times the operator is executed consecutively.

The 2-opt operator removes two existing edges of a tour, and then adds two other edges, so that it can generate a different effective tour. Replace edges of \((i, j+1)\) and \((j, i+1)\) with other
edges of \((i, i+1)\) and \((j, j+1)\). The distance of a tour is shortened after the exchange, which satisfies the condition \(d(i, i+1)+d(j, j+1)< d(i, j+1)+d(j, i+1)\), and the feasible solutions are improved, as figure 1 shown.

**Fig. 1.** The 2-opt operator.

(3) Global search operators

The process of global search includes two operators which are crossover operator, the operator of crossover and reverse in this paper.

1) Crossover operator

Because of the particularity of TSP, any component of the solution (or any gene of the individual) must be unique. In the global search process, two different parts of an individual are exchanged as shown in figure 2. As indicated in the figure, one part starts at \(i^{th}\) component with the components of length \(l\), and another part starts at \(j^{th}\) component with components of the same length \(l\), crossover operator is to exchange the two parts. Here, \(i, j\) and \(l\) are random integers and satisfy formula (3), \(n\) is the number of cities or the scale of TSP.

\[
1 \leq l \leq \min(|i-j|, n-\max(i, j))
\]  

(3)

**Fig. 2.** Crossover operator.

2) Crossover and reverse operator

On the basis of the crossover operator, crossover and reverse operator further arrange the components of the parts that have been exchanged in reverse order, which is shown as in figure 3.

**Fig. 3.** Crossover and reverse operator.

(4) Convergent criterion

If an individual has already converged, local search could not make much sense to it again, which means that it has fallen into local optimum. Global search operator should be adopted to enable it to escape from the precociousness. Simple convergence criterion is that if the absolute value of the difference between the fitness values of the two successive generations
of the same individual is less than a given threshold value as formula (4), the individual is considered to be convergent.

$$|f(T_{j+1}^λ) - f(T_j^λ)| < CT$$ \quad (4)

where $CT$ is a given small positive number, $j=1,2,\ldots,\text{size}$.

### 3 Experiments and analysis

In experiments part, IMNEFOTC, RGA and GSA algorithms have been tested for some instances of TSPLIB. All tests of this paper have been run on Lenovo ThinkPad, CPU: 2.5GHz, memory: 4G.

For different scale instances, we need to adjust relevant parameters such as $\text{size}$, $g_j$, and $\lambda_{max}$, etc. sometimes. The test data is recorded in table 1, where O.P. presents the real distance of the optimal path known, O.V. denotes the optimal value known. The distance of the best satisfactory solution which was got by GSA is denoted by B.F., B.F. deviates from O.P. is denoted by D (%). ID is the distance of the approximate solution computed by IMNEFOTC, and IDD (%) is its deviation from O.P.. GD is the distance of the tour got by traditional greedy heuristic, and GDD (%) is the deviation from O.P..

<table>
<thead>
<tr>
<th>instance</th>
<th>O.V.</th>
<th>O.P.</th>
<th>GSA</th>
<th>IMNEFOTC</th>
<th>greedy heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAYG29</td>
<td>9074.146</td>
<td>9074.146</td>
<td>0</td>
<td>9413.8</td>
<td>9886.2</td>
</tr>
<tr>
<td>ATT48</td>
<td>33523.71</td>
<td>33607.69</td>
<td>0.25</td>
<td>35854.94</td>
<td>40160.4</td>
</tr>
<tr>
<td>EIL51</td>
<td>426</td>
<td>429.9833</td>
<td>429.53</td>
<td>447.75</td>
<td>481.52</td>
</tr>
<tr>
<td>PR76</td>
<td>108159</td>
<td>108159.4</td>
<td>0</td>
<td>121268.7</td>
<td>147498</td>
</tr>
<tr>
<td>RD100</td>
<td>7910</td>
<td>7910.396</td>
<td>0</td>
<td>9547.615</td>
<td>9254.01</td>
</tr>
<tr>
<td>CH130</td>
<td>6110</td>
<td>6110.862</td>
<td>1.23</td>
<td>6694.897</td>
<td>7844.94</td>
</tr>
<tr>
<td>CH150</td>
<td>6528</td>
<td>6532.281</td>
<td>0.22</td>
<td>7358.887</td>
<td>7743.04</td>
</tr>
<tr>
<td>TSP225</td>
<td>3916</td>
<td>3859</td>
<td>3908.659</td>
<td>4289.228</td>
<td>4550.29</td>
</tr>
<tr>
<td>PCB442</td>
<td>50778</td>
<td>50783.55</td>
<td>3.02</td>
<td>57812.9</td>
<td>62878.8</td>
</tr>
<tr>
<td>RAT783</td>
<td>8806</td>
<td>9204.97</td>
<td>0.97</td>
<td>10045.78</td>
<td>10667.4</td>
</tr>
<tr>
<td>PR1002</td>
<td>259045</td>
<td>259066.7</td>
<td>3.64</td>
<td>299792.2</td>
<td>306842</td>
</tr>
<tr>
<td>PR2392</td>
<td>378032</td>
<td>378062.1</td>
<td>3.1</td>
<td>444597.5</td>
<td>456181.4</td>
</tr>
</tbody>
</table>

From table 1, we find a better path than the optimal path known in instance EIL51, which was found by GSA. The deviations of other instances of GSA do not exceed 4%. The time complexity of the GSA algorithm is polynomial function, and it can get a better trade-off between search time and search precision. From the results of the 10 instances which are randomly selected from TSPLIB, we know that IMNEFOTC performs better than greedy heuristic except instance RD100. IMNEFOTC is more suitable for solving large scale and super large problems.

### 4 Conclusions

This paper proposed a quick heuristic algorithm of IMNEFOTC and a general meta-heuristic search algorithm framework GSA, which are suitable for almost all scales TSP. GSA is a hybrid algorithm, which combines traditional heuristics and meta-heuristic algorithms. Some
instances of TSPLIB have been tested, and the results reveal that IMNEFOTC is better than greedy heuristic, and GSA is a pretty effective algorithm to solve different scales of TSP.

References


