A New Exact Algorithm for Traveling Salesman Problem with Time Complexity Interval (O(n^4), O(n^3·2^n))

YUNPENG LI, Southeast University

Traveling salesman problem is a NP-hard problem. Until now, researchers have not found a polynomial time algorithm for traveling salesman problem. Among the existing algorithms, dynamic programming algorithm can solve the problem in time $O(n^2 \cdot 2^n)$ where $n$ is the number of nodes in the graph. The branch-and-cut algorithm has been applied to solve the problem with a large number of nodes. However, branch-and-cut algorithm also has an exponential worst-case running time.

In this paper, a new exact algorithm for traveling salesman problem is proposed. The algorithm can be used to solve an arbitrary instance of traveling salesman problem in real life and the time complexity interval of the algorithm is $(O(n^4), O(n^3 \cdot 2^n))$. It means that for some instances, the algorithm can find the optimal solution in polynomial time although the algorithm also has an exponential worst-case running time. In other words, the algorithm tells us that not all the instances of traveling salesman problem need exponential time to compute the optimal solution. The algorithm of this paper can not only assist us to solve traveling salesman problem better, but also can assist us to deepen the comprehension of the relationship between NP-complete and P. Therefore, it is considerable in the further research on traveling salesman problem and NP-hard problem.

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems; G.2.2 [Discrete Mathematics]: Graph Theory — graph algorithms

General Terms: Algorithms, Design, Theory, Verification

Additional Key Words and Phrases: Traveling salesman problem, positively weighted graph, time complexity interval

1. INTRODUCTION

The traveling salesman problem is a classical combinatorial optimization problem. The research on the exact algorithm for traveling salesman problem is significant not only because the research has sufficient practical significance but also because the research results concern the reasonable cognition on the relationship between NP and P. Until now, researchers have not found a polynomial time algorithm for traveling salesman problem. Among the existing algorithms, dynamic programming algorithm [Held and Karp 1962] can solve the problem in time $O(n^2 \cdot 2^n)$ where $n$ is the number of nodes in the graph. The branch-and-cut algorithm has been applied to solve the problem with a large number of nodes [Padberg and Rinaldi 1987]. However, branch-and-cut algorithm also has an exponential worst-case running time [Cook and Hartmann 1990]. It is still difficult to improve the time bound of exact algorithm for traveling salesman problem. And it has not been determined whether an exact algorithm for traveling salesman problem that runs in time $O(1.9999^n)$ exists [Woeginger 2003].

In this paper, a new exact algorithm for traveling salesman problem is proposed. The algorithm can be used to solve an arbitrary instance of traveling salesman problem in real life, namely, the algorithm is a general exact algorithm for traveling salesman problem in real life. In the algorithm, this paper adopts information diffusion mode to search the optimal solution of traveling salesman problem. Firstly, the algorithm chooses a node of the graph as the source node. Then the source...
transmits information to every other node. In the algorithm, every piece of information travels at the same speed $v_{step}$ and along the shortest path between the information transmission node and the destination node. Meanwhile, each piece of information records the number and traversal sequence of the nodes it has traveled in the process of information transfer and computes the total traveling distance in the traveling process. When a piece of information arrives at the destination node, the information will be retransmitted to every node which the information has not traveled if the information is the first arriving information among the all the pieces of information which have traveled the same node set. If a piece of information has traveled all the nodes in the graph, it will be retransmitted back to the source node and the first returning information will bring back the shortest circle traversal sequence which the algorithm intends to search if the graph is connected. This is because every piece of information travels at the same speed. Therefore, when the algorithm finds that a piece of information returns to the start node or there is no more information to transmit and receive in the node system of the graph $G$ if graph $G$ is disconnected, the algorithm terminates.

In view of theoretical analysis, we demonstrate that the algorithm can obtain correct results only if $v_{step}$ is more than zero and no more than $P_{min}$ which is respectively the minimum value of the shortest path distances between two arbitrary nodes in the graph, namely, the weight value of the shortest edge in the graph. By using Fibonacci heap [Fredman and Tarjan 1987], the final time complexity interval of the algorithm is $O(n^3)$, $O(n^3 \cdot 2^n)$. It means that for some instances, the algorithm can find the optimal solution in polynomial time although the algorithm also has an exponential worst-case running time. In other words, the algorithm tells us that not all the instances of traveling salesman problem need exponential time to compute the optimal solution. The algorithm of this paper can not only assist us to solve traveling salesman problem better, but also can assist us to deepen the comprehension of the relationship between NP-complete and P. Therefore, it is considerable in the further research on traveling salesman problem and NP-hard problem.

The remainder of this paper is organized as follows: in Section 2, we compare our work with the related work on the subject; in Section 3, we present the problem description; in Section 4, we present a generic algorithm considering uniform speed diffusion; in Section 5, we demonstrate the correctness of the algorithm of this paper; in Section 6, we analyze the time complexity and space complexity of the algorithm; in Section 7, we show how to use Fibonacci heap to improve the sequential time bound; in Section 8, we discuss some remarks on traveling salesman problem and the algorithm of this paper; finally, we discuss and conclude this paper in Section 9.

2. RELATED WORK

The traveling salesman problem is a classical problem in the research field of graph algorithm. Richard M. Karp [1972] has proven that the Hamiltonian cycle problem was NP-complete, which implies the NP-hardness of traveling salesman problem. Until now, researchers have not found a polynomial time algorithm for traveling salesman problem. It remains an open problem whether the traveling salesman problem can be solved in time $O(1.999n)$ [Woeginger 2003].

In 1954, Dantzig, Fulkerson, and Johnson [1954] demonstrated that lager instances of the traveling salesman problem could be solved by linear program. In 1962, Held and Karp [1962] proposed a dynamic programming algorithm for traveling salesman problem, which can solve the traveling salesman problem in $O(n^2 \cdot 2^n)$ time. The time complexity is still the best one that is known today. For
A New Exact Algorithm for Traveling Salesman Problem

large-scale traveling salesman problem, the branch-and-cut algorithm is usually applied to compute the optimal solution. However, it also has an exponential worst-case running time.

In this paper, we propose a new exact algorithm for traveling salesman problem. The algorithm can be used to solve an arbitrary instance of traveling salesman problem in real life. The time complexity interval of the algorithm is \( O(n^4), O(n^3 \cdot 2^n) \). It means that for some instances, the algorithm can find the optimal solution in polynomial time although the algorithm also has an exponential worst-case running time. To our best knowledge, the algorithm of this paper is the first general exact algorithm for traveling salesman problem which can solve some instances in polynomial time. Therefore, the algorithm of this paper is considerable in the further research on traveling salesman problem and NP-hard problem.

3. PROBLEM DESCRIPTION

The traveling salesman problem can be simply described as follows: given a finite number of “cities” along with the cost of travel between each pair of them, find the cheapest way of visiting all of the cities and returning to the starting point. The travel costs are symmetric in the sense that traveling from city X to city Y costs just as much as traveling from Y to X [Applegate et al. 1998]. If we describe traveling salesman problem based on the concepts of graph theory, the problem can be stated as follows: for an undirected graph \( G=(V,E) \), where \( V \) is the set of nodes and \( E \) is the set of edges. The weight of its edge \((i,j) \in E \) is \( W_{ij} \) which represents the length or cost of the edge \((i,j) \in E \). In practical problems, the travel cost of arbitrary two adjacent cities is positive, namely, \( W_{ij} > 0 \ (i,j) \in E \). We use \( P_{ij} \) to represent the shortest traversal distance between node \( v_i \) and \( v_j \) \( (v_i, v_j \in V) \) and use \( P_{\text{min}} \) to represent the minimum \( P_{ij} \) in graph \( G \). In reality, \( P_{\text{min}} \) is equal to \( W_{\text{min}} \) which represents the shortest edge weight in graph \( G \).

Definition 1 (circle traversal sequence). For an undirected graph \( G=(V,E) \) whose start node is \( v_0 \), a circle traversal sequence of \( V \) is a traversal sequence which starts and ends with start node \( v_0 \), and includes each node of \( V \). Besides, each node only has one sequence position in a circle traversal sequence except the start node.

For the node set \( V \ (V=\{v_0, v_1, \ldots, v_{n-1}\} \ (n>2)) \), \( s \) is a circle traversal sequence of \( V \). If \( s=<v_0, v_1, \ldots, v_{n-1}, v_0> \), the traversal distance (traversal distance can be also called traversal cost) of \( s \) in graph \( G \) can be expressed as follow:

\[
L(s) = \sum_{i=1}^{n-1} P_{i-1,i} + P_{n-1,0}
\]  

(1)

Where \( L(s) \) represents the traversal distance of \( s \).

Theorem 1. For graph \( G \), \( L(s) \) is the shortest traversal distance of circle traversal sequence \( s \).

Proof: In \( L(s) \), the traversal distance of arbitrary two neighbor nodes of \( s \) is the shortest traversal distance, namely, the traveling salesman travels along the shortest path between the neighbor nodes of \( s \) and \( L(s) \) is equal to the summation of all the traversal distances of neighbor nodes of \( s \). As a result, \( L(s) \) be the shortest traversal distance of circle traversal sequence \( s \) in graph \( G \).

The shortest traversal distance of circle traversal sequence of \( V \) in graph \( G \) can be defined as follow:
\[ L_{\text{min}}(V) = \begin{cases} \min[L(s) : s \text{ is a circle traversal sequence of } V] & \text{if there is a circle traversal sequence of } V \\ +\infty & \text{otherwise} \end{cases} \] (2)

The shortest circle traversal sequence of \( V \) in graph \( G \) is defined as the circle traversal sequence whose traversal distance is equal to \( L_{\text{min}}(V) \) if there is a circle traversal sequence of \( V \). The traveling salesman problem is finding the shortest circle traversal sequence of \( V \) in graph \( G \). For node set \( V \), every node can be the start node of a shortest circle traversal sequence and \( L_{\text{min}}(V) \) is independent of the selection of start node. In practical problems, the distribution map of the cities is not always a complete graph, namely, the graph \( G \) of traveling salesman problem is not always a complete graph.

4. A GENERIC ALGORITHM CONSIDERING UNIFORM SPEED DIFFUSION

Let us imagine the scene as follows: each node in graph \( G \) has a node manager who decides how to retransmit information. The traveling salesman stands at a node (i.e. start node \( v_0 \)) in graph \( G \) and acts as the node manager. Firstly, the traveling salesman transmits \( n-1 \) pieces of information to all the other nodes in graph \( G \). Then, when a piece of information reaches its destination node, the node manager will decide how to retransmit the information. The node manager obeys two retransmission principles: (1) if the manager finds that there is another piece of information which has traveled the same node set and has no longer traversal distance, the just arriving information will not be retransmitted to any other node; otherwise, the manager will record the traversal state of the information and retransmit it; (2) for a piece of selected information, simultaneously retransmit the information to all the nodes which the information has not traveled. If the information has traveled all the nodes in graph \( G \), it will be retransmitted to the start node. If \( G \) is a connected graph, it can be proved that at least one shortest circle traversal sequence will be completely traveled in the process of information transfer. The concrete judging foundation and proof procedure will be explained in the latter part of this paper. In all the processes of information transfer, every piece of information travels at the same speed \( v_{\text{step}} \) \((0 < v_{\text{step}} \leq v_{\text{step}})\) and along the shortest path between the information transmission node and the destination node. Meanwhile, each piece of information records the number and traversal sequence of the nodes it has traveled in the process of information transfer and computes the total traveling distance in the traveling process.

Because at least one shortest circle traversal sequence will be completely traveled in the process of information transfer if \( G \) is a connected graph and all the information travels at the same speed, the information which travels along the shortest circle traversal sequence must return to the start node firstly. Then, based on the returning sequence, the traveling salesman can judge which circle traversal sequence is the shortest circle traversal sequence.

Based on the aforementioned thought, the algorithm of this paper is designed. The basic description of the algorithm is as follows: at the beginning, the traveling salesman transmits \( n-1 \) pieces of information at the start node to all the other nodes in graph \( G \). The original traversal distance, \( d_{si} \), of each piece of information is set as zero when it is transmitted at the start node. Every piece of information travels at the same speed \( v_{\text{step}} \) and along the shortest path between the transmission node and the destination node. Meanwhile, each piece of information records the number, \( T_n \), and traversal sequence of the nodes it has traveled and computes the total traveling distance in the traveling process. When a piece of information reaches its destination
node, the information manager of the destination node will judge whether there is another piece of information which has traveled the same node set and has no longer traversal distance. If there is such a piece of information, the arriving information will not be retransmitted. On the contrary, the information will be simultaneously retransmitted to all the nodes which the information has not traveled. If a piece of information has traveled all the nodes in graph G, it will be retransmitted to the start node. After that, the node manager will record the traversal state of the arriving information which includes the traversal sequence of the nodes it has traveled and the total traveling distance in the traveling process. When the traveling salesman finds that a piece of information returns to the start node or there is no more information to transmit and receive in the node system of the graph G, the algorithm terminates. If G is a connected graph, the traversal sequence of the nodes which the firstly returning information has recorded is the shortest circle traversal sequence of V in graph G.

In the concrete implement of the algorithm, a global clock \( t \) is adopted to calculate the globally identical travelling distance, \( d_{si} \), of all the spreading information. The travelling distance, \( d_{su} \), is equal to \( t^{\text{step}} \). In the algorithm, the global clock \( t \) is corresponding to the round number of the loop. When a node manager finds the traversal distance of one information record is no more than \( d_{su} \), it transmits information for the information record. In the process, every piece of information will directly confirm the traversal state which it will pass through if it arrives at the destination node. The traversal state includes the traversal sequence of the nodes, the number of traversal nodes and the total path distance, \( d_{st} \). If we use \( d_{si} \) to represent the distance which a piece of information has pass through when the information arrive at node \( v_i \) and the information is retransmitted to node \( v_j \), \( d_{si} \) is equal to the sum of \( d_{si} \) and \( P_{ij} \) which is the shortest traversal distance between node \( v_i \) and \( v_j \) (\( v_i, v_j \in V \)). In addition, the information will record the traversal nodes in the shortest path between node \( v_i \) and \( v_j \). If \( v_k \) is the first node which the information will travel in the shortest path from \( v_i \) to \( v_j \) and the information has not travelled node \( v_k \), node \( v_k \) will be added to the end of the traversal sequence the information has recorded and the number of traversal nodes of the information, \( T_n \), increases by one, namely, \( T_n = T_n + 1 \). Then \( T_n \) corresponds to the ordinal position of \( v_k \) in the traversal sequence of the information. In contrast, if the information has travelled node \( v_k \), node \( v_k \) will not be added to the traversal sequence. In other word, one node is added to the traversal sequence of a piece of information only when the information travels the node for the first time. After confirming the traversal state, the information will access the information records of destination node to judge if there is a record whose traversal node set is the same. If there is not such a record, the information will make the manager of the destination node record its arriving traversal state. Conversely, if there is such a record, the information will compare \( d_{st} \) with the traversal distance of the record. If \( d_{st} \) is shorter, the information will reset the record and make the record keep an account of the arriving traversal state of the information. However, if \( d_{st} \) is equal or longer, the information does nothing. In the process, every node only records the traversal state of the information whose traversal distance is currently the shortest one among all the pieces of information which have traveled the same node set and will arrive at itself and only focuses on whether the information which may spread along the shortest traversal sequence from the source node to itself for a certain node set has arrived. Namely, for a certain node set, every node ignores the information which is not possible to spread along the
shortest traversal sequence from the source node to itself because the information has no influence on the final computational result. This design can greatly reduce the time complexity of the algorithm.

Because every piece of information spreads at the same speed $v_{step}$, every piece of information spreads the same distance $d_{su}$ at the same $t$ clock. If a piece of information will spread $d_{st}$ distance when it arrives at the destination node, then it arrives at its destination node only when $d_{st}$ is no longer than $d_{su}$. Because every node only records the information whose traversal distance is currently the shortest one among all the pieces of information which have traveled the same node set and will arrive at itself, we can judge that for a certain node set, the information which spreads along the shortest traversal sequence from the source node to one destination node has arrived at the destination node when the traversal distance of the corresponding information record is no longer than $d_{su}$. Based on the theoretical analysis, we can demonstrate that only if $0 < v_{step} \leq P_{min}$, the traversal distance of the corresponding information record is equal to the shortest traversal distance from source node to the destination node for a certain node set when the traversal distance is no longer than $d_{su}$ and is longer than $d_{su} - v_{step}$ even if the generic algorithm runs in a sequential system (please see the demonstration of Theorem 8). It means that a node manager should transmit information for an information record at the moment that the traversal distance is no longer than $d_{su}$ and is longer than $d_{su} - v_{step}$ on condition of $0 < v_{step} \leq P_{min}$.

Based on the aforementioned analyses, the generic algorithm should check every information record which has not transmitted information to search the information records which satisfy the condition of transmitting information and make them transmit information in every round loop. Then at the end of every round loop, add $d_{su}$ by $v_{step}$, which means that all the spreading information has spread another $v_{step}$ distance. After that, the generic algorithm goes into the next round loop. Repeat the aforementioned processes until the termination of the algorithm. The more detailed description of the generic algorithm is given as follows:

For an undirected graph $G=(V,E)$ ($|V|=N$), we use Dijkstra's algorithm to compute the shortest path between arbitrary two nodes and store the result in a three dimensional array $shortestpath[N][N][2]$. For arbitrary two nodes $i$ and $j$, $shortestpath[i][j][0]$ stores the distance of the shortest path from node $i$ to node $j$, and $shortestpath[i][j][1]$ stores the id of the first node which the information will travel if it spreads along the shortest path from node $i$ to node $j$. $shortestpath[N][N][2]$ is a necessary input for the algorithm of this paper.

In the paper, we establish an information class to conveniently implement the algorithm. The basic description of class information is as follows:

```cpp
Class information {
    int traversal_sequence[N];
    int traversal_size;
    int traversal_state_id;
    double distance;
    int Node_id;
    public:
        information * send_pointer_Pre;
        information * send_pointer_next;
        information(double);
        void settraversal_sequence(int a, int b);[traversal_sequence[a]=b;]
```
A New Exact Algorithm for Traveling Salesman Problem

```c
int gettraversal_sequence(int a){return traversal_sequence[a];}
void compute_traversal_state_id();
void add_traversal_state_id(int);
int gettraversal_state_id(){return traversal_state_id;}
void settraversal_state_id(int id){ traversal_state_id=id;}
void settraversal_size(int size){traversal_size=size;}
int gettraversal_size(){return traversal_size;}
void setNode_id(int id){Node_id=id;}
int getNode_id(){return Node_id;}
void setdistance(double d){distance=d;}
double getdistance(){return distance;}
void addtraversal_sequence(int);
```

The function `information(double)` is the constructed function:

```c
information::information (double d)
{
    for i=0 to N-1
        do traversal_sequence[i]=0;
    end
    traversal_size=0;
    traversal_state_id=0;
    distance=d;
    Node_id=-1;
    send_pointer_Pre=NULL;
    send_pointer_next=NULL;
}
```

The function `compute_traversal_state_id()` is used to compute the traversal state of a piece of information. In this paper, we make each kind of traversal sequence correspond to a binary number and transform the binary number into the corresponding decimal number as the traversal state id of the traversal sequence.

```c
void information:: compute_traversal_state_id()
{
    traversal_state_id=0;
    for i=0 to N-1
        do if traversal_sequence[i]>0
            then value=1;
            else value=0;
        end
        traversal_state_id+=value*pow(2,i);
    end
}
```

The function `add_traversal_state_id(int)` is defined as follows:

```c
void information:: add_traversal_state_id(int a)
{
    if (a>=0& &a<N& &traversal_sequence[a]<=0)
        then
            traversal_state_id+= pow(2,a);
    end
}
```
The function `addtraversal_sequence(int)` is used to add a new traversal node into the traversal sequence of a piece of information. It adjusts the traversal_size and records the traversal order of the new traversal node. In the paper, a piece of information records the traversal order of a node only when the node is travelled for the first time.

```c
void information:: addtraversal_sequence(int a)
{
    if (a>=0&&a<N&&traversal_sequence[a]<=0)
    then
        traversal_size++;
        traversal_sequence[a]=traversal_size;
    end
}
```

In this paper, we establish an index AVL tree [Adelson-Velsky and Landis 1962] for each node based on the traversal state id to improve the searching and comparing efficiency of the information. In the AVL tree, each AVL tree node has four attributes `{state_data, info_pointer, left_node_pointer, right_node_pointer}`, where the state_data stores the traversal state id of the information which the node’s information pointer, info_pointer points to and the left_node_pointer and right_node_pointer respectively point to left child node and right child node. In the node initialization, all the pointer attributes will be set as NULL. In this paper, we define a insert function, `InsertAVL_tree(AVL_tree T, information I)`. In the implement of the function, firstly, it searches the AVL tree to judge whether there is a node whose state_data is equal to the traversal_state_id of I. If there is such a node, it returns a pointer of the node. If there is no such a node, it establishes an AVL tree node whose state_data is equal to the traversal_state_id of I, inserts the node into AVL tree T and returns a pointer of the new establishing node.

**ALGORITHM 1. Traveling Salesman Problem Computation**

**Input:** the three-dimensional array `shortestpath[N][N][2]` of graph G, the source node id, S and the propagation speed, V_stop.

**Output:** output the distance and the traversal sequence of a shortest circle traversal sequence of graph G.

```
\(d_{\text{su}}=0; N=|V|; \text{distance} \_\text{label}=+\infty; \text{flag}=true;\)
Create Doublylinkedlist(send_list);
AVL_tree T[N]; /*Create a AVL tree array T[N]*/
information I(0);
I.settraversal_sequence(S,1);
I.add_traversal_state_id(S);
I.settraversal_size(1);
I.setNode_id(S);
AVL_node_pointer= InsertAVL_tree(T[S], I);
AVL_node_pointer -> info_pointer=& I;
Insert Doublylinkedlist(send_list, I); /*Insert the information in the list tail*/
While ((distance \_\text{label}> \(d_{\text{su}}\) and flag) do
    Current_pointer=head_pointer(send_list);
    List_tail_pointer=tail_pointer(send_list);
    While (Current_pointer!=0) do
        if (Current_pointer ->distance<= \(d_{\text{su}}\) then
            /*The information has travelled all the nodes*/
            if (Current_pointer ->gettraversal_size()==N)
                then destination_id=S;
                send_id=Current_pointer->getNode_id();
                distance_in=Current_pointer->getdistance();
```
A New Exact Algorithm for Traveling Salesman Problem

distance_in += shortestpath[send_id][destination_id][0];
if (distance_in== $\infty$) then flag=false; break;
information I(distance_in);
/*Duplicate the traversal sequence*/
for i=0 to N-1
    do I.settraversal_sequence(i,Current_pointer->gettraversal_sequence(i));
end
I.settraversal_size(N);
I.setNode_id(S);
I.settraversal_state_id(Current_pointer->gettraversal_state_id());
AVL_node_pointer= InsertAVL_tree(T[S], I);
if (AVL_node_pointer->info_pointer==NULL)
    then AVL_node_pointer->info_pointer=&I;
distance_label=I.getdistance();
else
    if (AVL_node_pointer->info_pointer->getdistance()>I.getdistance())
        then distance_label=I.getdistance();
            AVL_node_pointer->info_pointer->setdistance(I.getdistance());
            p= AVL_node_pointer->info_pointer;
            for j=0 to N-1
                do p->settraversal_sequence(j, I.gettraversal_sequence(j));
            end
        end
    end
else
    /*The information has not travelled all the nodes*/
    for k=0 to N-1
        do if (Current_pointer->gettraversal_sequence(k)<=0)
                then destination_id=k;
                    send_id=Current_pointer->getNode_id();
                    distance_in= Current_pointer->getdistance();
                    distance_in+= shortestpath[send_id][destination_id][0];
                    if (distance_in== $\infty$) then flag=false; break;
                    information I(distance_in);
                    for k=0 to N-1
                        do sequence_re=Current_pointer->gettraversal_sequence(i);
                            if (sequence_re>0)
                                then I.settraversal_sequence(l, sequence_re);
                            end
                        end
        end
end
I.settraversal_size(Current_pointer->gettraversal_size());
I.setNode_id(k);
I.settraversal_state_id(Current_pointer->gettraversal_state_id());
/*Record the intermediary node of the shortest path*/
While (send_id!=destination_id) do
    receiver_id=shortestpath[send_id][destination_id][1];
    I.addtraversal_sequence(receiver_id);
    I.addtraversal_state_id(receiver_id);
    send_id=receiver_id;
end
AVL_node_pointer= InsertAVL_tree(T[k], I);
if (AVL_node_pointer->info_pointer==NULL)
    then AVL_node_pointer->info_pointer=&I;
    Insert Doublylinkedlist(send_list, I);
else p= AVL_node_pointer->info_pointer;
if (p->getdistance()>I.getdistance())
    then p->setdistance(I.getdistance);
    for j<0 to N-1
        do sequence_re=I.gettraversal_sequence(j);
           p->settraversal_sequence(j,sequence_re);
end
Re_Current_pointer=Current_pointer;
if (Current_pointer==List_tail_pointer)
    then Current_pointer=0;
else Current_pointer=Current_pointer->send_pointer_next;
end
Remove Doublelinkedlist(send_list,Re_Current_pointer);
else
    if (Current_pointer==List_tail_pointer)
        then Current_pointer=0;
    else Current_pointer=Current_pointer->send_pointer_next;
end
end
dsu+=Vstep;
end

5. THEORETICAL PROOF OF THE CORRECTNESS

In the algorithm, we adopt global clock \( t \) to calculate the globally identical propagation distance, \( dsu \), of all the spreading information. The propagation distance, \( dsu \), is equal to \( t\cdot v_{step} \). In the algorithm, the global clock \( t \) is corresponding to the round number of the loop.

**Theorem 2.** For a connected graph, if every node manager transmits each piece of arriving information to any node which has not retransmitted the information and the algorithm terminates when there is no more information is transmitted, every possible circle traversal sequence including the shortest circle traversal sequence will be completely traveled by a piece of information.

**Proof** Suppose that there is a circle traversal sequence of \( V, s'=<v_0, v_2, v_1, ..., v_0> \) which is not completely traveled by a piece of information. In other words, the information which was assigned to travel along \( s' \) was terminated in advance, before it completely travels the circle traversal sequence. In the algorithm, a piece of information can only be terminated by a node manager. Then we suppose that the information was terminated by the node manager of \( v_i \) and the next neighbor node in the traversal sequence is \( v_j \). It means that the node manager of \( v_i \) did not transmit information to node \( v_j \).

However, this is in contradiction with the premise in Theorem 2. In a circle traversal sequence, one node only has one sequence position. Because node \( v_j \) is behind \( v_i \), the node set which the information records does not include node \( v_j \) when it arrived at node \( v_i \). According to the premise in Theorem 2, every node manager transmits each piece of arriving information to any node which has not retransmitted the information. Then the node manager will transmit the information to node \( v_j \). In other word, the information will not terminate at node \( v_i \). Namely, the previous
A New Exact Algorithm for Traveling Salesman Problem

hypothetical situation is not possible to appear. Then we can conclude that Theorem 2 is valid. It means that all the possible circle traversal sequence will be considered and compared to find the shortest one in this situation.

**Lemma 3.** For an undirected graph $G=(V,E)$, $P_{ij}$ represents the shortest traversal distance between node $v_i$ and $v_j$ ($v_i, v_j \in V$) in the graph. For an arbitrary node of the graph, $v_k$, we have:

$$P_{ij} \leq P_{ik} + P_{kj}$$  \hspace{1cm} (3)

**Proof** Suppose that $P_{ij} > P_{ik} + P_{kj}$. It means that the traversal distance of the path $v_i \rightarrow v_j$ is shorter than the traversal distance of the path $v_i \rightarrow v_j$. Then the path $v_j \rightarrow v_j$ is not the shortest path between node $v_i$ and $v_j$ and $P_{ij}$ is not the shortest traversal distance between node $v_i$ and $v_j$ ($v_i, v_j \in V$) in graph $G$. This is in contract with the premise that $P_{ij}$ represents the shortest traversal distance between node $v_i$ and $v_j$ ($v_i, v_j \in V$) in the graph. This illustrates that the previous assumption $P_{ij} > P_{ik} + P_{kj}$ is invalid. Then we can conclude that Theorem 2 is correct.

**Theorem 4.** If a piece of information, $I_1$ selects a node which it has traveled as a new destination node, there must be another piece of information whose traversal distance is no longer than that of $I_1$ when they return to the start node $v_0$.

**Proof** Suppose that a piece of information, $I_1$ has the node sequence $s_1=<v_0, v_1, v_2, \ldots, v_n>$ and arrives at node $v_1$. $I_1$ is retransmitted to node $v_0$ before traveling the node set $V$, which consists of the nodes that have not been traveled by itself in graph $G$. Suppose that $s_2=<v_i, v_j, \ldots, v_0>$ represents the shortest one of all the traversal sequences which start at node $v_i$, end with $v_0$ and include all the nodes of $V$ and $s_3=<v_i, v_k, v_n, \ldots, v_1, v_0>$ represents the shortest one of all the traversal sequences which start with the sequence $v_i, v_k>$, end with $v_0$ and include all the nodes of $V$. Based on Lemma 3, we can judge that the traversal distance of $s_1=<v_i, v_2, \ldots, v_n, v_0>$ is no longer than that of $s_3$ because of $P_{sa} \leq P_{ik} + P_{ka}$. Because $s_2=<v_i, v_j, \ldots, v_0>$ is the shortest one of all the traversal sequences which start at node $v_i$, end with $v_0$ and include all the nodes of $V$, the traversal distance of $s_2$ is no longer than that of $s_4$. As a result, the traversal distance of $s_2$ is no longer than that of $s_3$. It means that the traversal distance of the information which travels along the sequence $s_1 \rightarrow s_2$ is no longer than the shortest traversal distance which $I_1$ can travel when it returns to the start node. Consequently, we can conclude that Theorem 4 is valid. It means that for a piece of arriving information, the node manager only needs to transmit information to the nodes which it has not traveled. In other words, based on Theorem 4, we can judge whether a circle traversal sequence is possible to be the shortest one at some retransmission node and terminate the information which cannot travel along a shortest circle traversal sequence in advance.

**Theorem 5.** There are two pieces of information, $I_1$ and $I_2$ which start with start node, end with the same information transmission node and travel the same node set. If the traversal distance of $I_1$ is no longer than that of $I_2$, the possible shortest traversal distance of $I_1$ is no longer than that of $I_2$ when they return to the start node $v_0$. 
Proof Suppose that \( I_1 \) and \( I_2 \) have traveled the same node set \( S_i=[v_0, \ldots, v_k, \ldots, v_l] \) and arrive at node \( v_i \). The traversal distance of \( I_1 \) is \( d_1 \) and the traversal distance of \( I_2 \) is \( d_2 \). Based on the premise, \( d_1 \leq d_2 \). \( V \) represents the node set which consists of the nodes that are not included by \( S_i \) in graph \( G \). Suppose that \( s_2=<v_{i1}, \ldots, v_{ik}, \ldots, v_{il}> \) represents the shortest one of all the traversal sequences which start at node \( v_i \), end with \( v_0 \) and include all the nodes of \( V \) and the traversal distance is \( d_{\min} \). Then the possible shortest traversal distance of \( I_1 \) and \( I_2 \) are respectively equal to \( d_1+d_{\min} \) and \( d_2+d_{\min} \) when they return to the start node \( v_0 \). Because \( d_1 \leq d_2 \), \( d_1+d_{\min} \leq d_2+d_{\min} \).

This illustrates that Theorem 5 is valid. It means that for all the arriving information that travels the same node set, the node manager only needs to transmit information for the shortest one. This is because the algorithm only needs to find one shortest circle traversal sequence.

**Theorem 6.** For a directed graph \( G=(V,E) \), where \( V \) is the set of nodes and \( E \) is the set of edges. \( P_{ij} \) represents the shortest traversal distance between node \( v_i \) and \( v_j \) \((v_i, v_j \in V)\) in graph \( G \). Suppose \( s=<v_{i1}, v_{i2}, \ldots, v_{ik}> \) is the shortest traversal sequence which starts with node \( v_{i1} \), ends with node \( v_{ik} \) and whose traveling node set is \( V_s \) \((V_s=[v_{i1}, v_{i2}, \ldots, v_{ik}])\). For arbitrary \( i, j \) \((1 \leq i \leq j \leq k)\), suppose \( s_{ij}=<v_{i}, v_{i+1}, \ldots, v_{j}> \) is the subsequence of \( s \) from \( v_i \) to \( v_j \). Then \( s_{ij} \) is the shortest traversal sequence which starts with node \( v_i \), ends with node \( v_j \) and whose traveling node set is \( V_{s'} \) \((V_{s'}=[v_{i}, v_{i+1}, \ldots, v_{j}])\).

\[
L(s) = L(s_{u}) + L(s_{j}) + L(s_{ik}) \quad \text{Suppose that there is a traversal sequence} \quad s_u \quad \text{which starts with node} \quad v_i \quad \text{ends with node} \quad v_j \quad \text{and whose traveling node set is} \quad V_u, \\
\text{and} \quad L(s_{j}) \leq L(s_{uj}). \quad \text{and} \quad L(s_{j}') \leq L(s_{uj}). \quad \text{Then} \quad v_i \rightarrow v_j \rightarrow v_k \text{is a traversal sequence which starts with node} \quad v_i, \quad \text{ends with node} \quad v_k \quad \text{and whose traveling node set is} \quad V_u, \quad \text{and the traversal distance} \quad L(s_u) + L(s_{j}) + L(s_{ik}) \quad \text{is less than} \quad L(p). \quad \text{This is in contradiction with the assumption that} \quad s \quad \text{is the shortest traversal sequence which starts with node} \quad v_i, \quad \text{ends with node} \quad v_k \quad \text{and whose traveling node set is} \quad V_u. \quad \text{Then we can conclude that Theorem 6 is valid.}
\]

**Theorem 7.** For a connected graph, at least one shortest circle traversal sequence of \( V \) is completely traveled by a piece of information in the generic algorithm.

**Proof** Based on Theorem 2, we know that the algorithm will consider and compare all the possible circle traversal sequence to find the shortest one. Based on the Theorem 4 and Theorem 5, we know that the node manager will not terminate all the shortest circle traversal sequences if the node manager transmits information following the two retransmission principles which are described in section 3. It means that at least one piece of information travels along one shortest circle traversal sequence of \( V \) and returns to the start node successfully in the generic algorithm. Then we know that Theorem 7 is valid.

**Theorem 8.** Even if the generic algorithm runs in a sequential system, only if \( 0 < v_{\text{step}} \leq P_{\min} \), the traversal distance of an information record is equal to the shortest traversal distances of all the possible traversal sequences which start with the start node, end with the information record holder and has the same traveling node set when the traversal distance is no longer than \( d_{su} \) and is longer than \( d_{aur} v_{\text{step}} \).
Proof we use $V_s$ to represent the subset of $V$ and use $d_{s,\rightarrow v}$ to represent the traversal distance of the information record whose traveling node set is $V_s$ at node $v_i$ ($v_i \in V$). And we use $d_{min,\rightarrow s,\rightarrow v}$ to represent the traversal distance of the shortest traversal sequence which starts with the start node $v_0$, ends with node $v_i$ and whose traveling node set is $V_s$. Suppose that the $d_{s,v_0}$ $\neq d_{min,\rightarrow s,\rightarrow v}$ (i.e. $d_{s,v_0} > d_{min,\rightarrow s,\rightarrow v} \lor d_{s,v_0} < d_{min,\rightarrow s,\rightarrow v}$) when $d_{s,v_0} - v_{step} < d_{s,v_0} \leq d_{s,v_0}$ on the condition of $0 < v_{step} \leq P_{min}$.

(I) Suppose that for arbitrary node $v_i$ ($v_i \in V$) which is different from the start node, $d_{s,v_0} > d_{min,\rightarrow s,\rightarrow v}$ when $d_{s,v_0} - v_{step} < d_{s,v_0} \leq d_{s,v_0}$ ( $d_{s,v_0} = t_0 * v_{step}, t_0 \geq 1$) on condition of $0 < v_{step} \leq P_{min}$ . We assume that shortest traversal sequence which starts with the start node, ends with node $v_i$ and whose traveling node set is $V_s$ is $s_{min}$ ($s_{min} = (s_0, v_i, v_j, v_i)$) and $P_{ji}$ is the shortest traversal distance between node $v_i$ and $v_j$ ($v_i, v_j \in V$) in graph G. Based on the execution process of the algorithm, we can judge $d_{s,v_0} > d_{min,\rightarrow s,\rightarrow v}$.

Proof

If $d_{s,v_0} > d_{min,\rightarrow s,\rightarrow v}$, $d_{s,v_0} + P_{ji} \leq d_{min,\rightarrow s,\rightarrow v} + P_{ji} < d_{s,v_0}$

$\therefore d_{s,v_0} \leq d_{s,v_0} = t_0 * v_{step}$ and $P_{ji} \geq P_{min} \geq v_{step}$.

$d_{s,v_0} = d_{min,\rightarrow s,\rightarrow v} < t_0 * v_{step} - P_{ji} \leq (t_0 - 1) * v_{step}$

This means that node $j$ retransmitted information to node $i$ at $t_o-1$ clock and compare $d_{s,v_0} = t_0 * v_{step}$ with $d_{s,v_0}$. The directed result is $d_{s,v_0} \leq d_{s,v_0} + P_{ji}$ at $t_0$ clock, which is determined by the mechanism of algorithm. Then $d_{s,v_0} = d_{min,\rightarrow s,\rightarrow v}$, it is in contradiction with $d_{s,v_0} > d_{min,\rightarrow s,\rightarrow v}$.

Consequently, we can judge $d_{s,v_0} > d_{min,\rightarrow s,\rightarrow v}$ when $d_{s,v_0} - v_{step} < d_{s,v_0} \leq d_{s,v_0}$ on condition of $0 < v_{step} \leq P_{min}$. Based on the same argument, we can judge $d_{s,v_0} > d_{min,\rightarrow s,\rightarrow v}$ and $d_{s,v_0} > d_{min,\rightarrow s,\rightarrow v}$.

However, we set the original traversal distance, $d_{ai}$, of each piece of information which is transmitted by the traveling salesman at the beginning of the algorithm as zero which is equal to the shortest traversal distance $d_{min,\rightarrow s,\rightarrow v}$, namely, $d_{s,v_0} = d_{min,\rightarrow s,\rightarrow v}$. This means that $d_{s,v_0} > d_{min,\rightarrow s,\rightarrow v}$ is not possible to appear in the algorithm because it is in contradiction with the initial setting. It also means that $d_{s,v_0} > d_{min,\rightarrow s,\rightarrow v}$ is not possible to appear in the algorithm when $d_{s,v_0} - v_{step} < d_{s,v_0} \leq d_{s,v_0}$. As a result, we can conclude that $d_{s,v_0} = d_{min,\rightarrow s,\rightarrow v}$ when $d_{s,v_0} - v_{step} < d_{s,v_0} \leq d_{s,v_0}$ on condition of $0 < v_{step} \leq P_{min}$.

(II) Suppose that for arbitrary node $i$ which is different from the start node, $d_{s,v_0} < d_{min,\rightarrow s,\rightarrow v}$ when $d_{s,v_0} - v_{step} < d_{s,v_0} \leq d_{s,v_0}$ ( $d_{s,v_0} = t_0 * v_{step}, t_0 \geq 1$) on
condition of $0 < v_{\text{step}} \leq P_{\text{min}}$. We assume that $d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_i}$ is the traversal distance of the traversal sequence $s(=v_0, v_1, \ldots, v_k)$. Then $d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_i} = d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}} + P_{ki}$ where $V_s = \{v_0, v_1, \ldots, v_k\}$ and $P_{ki}$ is the shortest traversal distance between node $v_k$ and $v_i$ ($v_i, v_k \in V$) in graph $G$. Based on the execution process of the algorithm, we can judge $d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}} < d_{\min - s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}}$.

**Proof**

If $d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_i} \geq d_{\min - s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}}$, $d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}} = d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-2}} + P_{ki}$, $d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}} = d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-2}} + P_{ki} \geq d_{\min - s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}}$.

This is in contradiction with $d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}} < d_{\min - s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}}$. Consequently, we can judge $d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}} < d_{\min - s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}}$ when $d_{s \rightarrow v_{\text{step}}} < d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}}$ on condition of $0 < v_{\text{step}} \leq P_{\text{min}}$. Based on the same argument, we can judge $d_{s \rightarrow [v_j, v_k]} < d_{\min - s \rightarrow [v_j, v_k]}$ and $d_{s \rightarrow [v_i, v_j]} < d_{\min - s \rightarrow [v_i, v_j]}$.

However, we set $d_{s \rightarrow [v_i, v_j]} = d_{\min - s \rightarrow [v_i, v_j]}$ at the beginning of the algorithm. This means that $d_{s \rightarrow [v_i, v_j]} < d_{\min - s \rightarrow [v_i, v_j]}$ is not possible to appear in the algorithm. As a result, we can conclude that $d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}} = d_{\min - s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}}$ in the algorithm.

Combining all the aforementioned analyses, we can conclude that for arbitrary node $i$ which is different from source node, $d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}} = d_{\min - s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}}$ when $d_{s \rightarrow v_{\text{step}}} < d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}} \leq d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}}$ on condition of $0 < v_{\text{step}} \leq P_{\text{min}}$. Additionally, we set $d_{s \rightarrow [v_i, v_j]} = d_{\min - s \rightarrow [v_i, v_j]}$ at the beginning of the algorithm. Then we can judge that Theorem 8 is valid.

Given Theorem 5 and Theorem 8, every node manager only needs to transmit information for its information record when the traversal distance is no longer than $d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}}$ and is longer than $d_{s \rightarrow v_0 \rightarrow \cdots \rightarrow v_{k-1}}$ in the generic algorithm. And this makes the generic algorithm have the least total frequency of information transmission.

6. **ANALYSIS OF THE COMPLEXITY**

**Definition 2** (shortest path graph). For an undirected graph $G=(V,E)$, the shortest path graph is a spanning sub-graph of $G$, which includes each node of $V$. And for arbitrarily two nodes $v_i$ and $v_j$, the shortest path distance between $v_i$ and $v_j$ in the shortest path graph is the same as that of $G$.

**Definition 3** (minimum shortest path graph). For an undirected graph $G=(V,E)$, the minimum shortest path graph is the shortest path graph whose number of edges is the least among all the shortest path graphs. In the minimum shortest path graph, every edge is the unique shortest path between the two end-nodes of the edge.

For an undirected graph $G=(V,E)$, the node number of $V$ is $n$ ($n \geq 2$) and the number of edges is $m$ ($m \geq 0$). $P_{ij}$ is the shortest traversal distance between node $v_i$ and $v_j$ ($v_i, v_j \in V$) and $P_{\text{min}}$ is the minimum $P_{ij}$ in graph $G$. In the generic algorithm, an information record is established once, inserted into an AVL tree once, inserted into the record list once and removed once from the record list once. And the
algorithm transmits at most $n-1$ pieces of information for an information record. Additionally, in the record list, every information record will compare its traversal distance with $d_{uv}$ in every round loop. Then the time complexity of the algorithm is determined by the sum of operation frequency of all the six parts. We use $k_i$ ($1 \leq k_i \leq n$) to represent the traveling node number of a piece of information when it arrives at node $v_i$.

**Theorem 9.** If $G$ is a complete graph and the minimum shortest path graph is the same as itself, the total number of information records in the generic algorithm is $O(n \cdot 2^n)$.

**Proof** for arbitrary node $v_i$ which is different the start node, $k_i$ can be arbitrarily positive integer which is more than one and no more than $n$. And node $v_i$ will have at most $C_{n-2}^{k_i-2}$ information records whose traveling node number is $k_i$. This is because the arriving information can have any possible node combined state when the minimum shortest path graph of $G$ is a complete graph. Then node $v_i$ will have $2n-2$ ($2^{n-2} = C_{n-2}^0 + C_{n-2}^1 + ... + C_{n-2}^{n-2}$) information records. Because graph $G$ has $n$ nodes, the total number of information records in the generic algorithm is $O(n \cdot 2^n)$. Then we can conclude Theorem 9 is valid. Besides, it is not difficult to find that for an arbitrary graph, the total number of information records in the generic algorithm is $O(n \cdot 2^n)$.

For each information records, a node manager transmits at most $n-1$ pieces of information. Given the $n-1$ pieces of information which are transmitted by the traveling salesman at the start node, the total frequency of information transmission in the generic algorithm is at most $(n-1)^2 \cdot 2^{n-2} + n - 1$ which is less than $n^2 \cdot 2^n$ when $n \geq 2$. Then the total frequency of information transmission in the generic algorithm is $O(n^2 \cdot 2^n)$.

**Theorem 10.** If $G$ is a graph whose minimum shortest path graph is a cyclic graph, the total number of information records in the generic algorithm is $O(n^3)$.

**Proof**

![Cyclic Graph](image)

As Figure 1 shows, $G$ is a cyclic graph and the minimum shortest path graph is the same as itself. For arbitrarily node $v_i$ which is different the start node $v_0$, the information which is transmitted by the traveling salesman has two candidate directions to arrive at node $v_i$, the clockwise direction and counter-clockwise direction. Suppose that there are $l_i$ ($0 \leq l_i \leq n-2$) intermediary nodes in the clockwise path from the start node $v_0$ to node $v_i$. Suppose that the clockwise path is $p=<v_0,v_1,v_2,...,v_{i-1},v_i>$. If a piece of information arrives at node $v_i$ via node $v_{i-1}$, it must travel all the
nodes whose node id is no more than i because every piece of information will only be retransmit to nodes which it has not travelled before. At the same time, it also can travel some nodes whose node id is more than i.

**Lemma 11.** For the graph G in Figure 1, a piece of information arrives at node $v_i$ via node $v_{i+1}$ and travels node $v_j$ ($j>i$). If there are $l_j$ intermediary nodes in the counter-clockwise path from the start node $v_0$ to node $v_j$, the traveling node number of the information is no less than $l_i + l_j + 3$.

**Proof** because there are $l_j$ intermediary nodes in the counter-clockwise path from the start node $v_0$ to node $v_j$, then the information travels at least $l_j + 2$ nodes including node $v_0$ when it arrives at node $v_j$ along the counter-clockwise direction. At the same time, if the information arrives at node $v_i$ via node $v_{i+1}$, it has to travel the clockwise path $p$ whose node number is $l_i + 2$ including node $v_0$. Consequently, the total traveling node number of the information is no less than $l_i + l_j + 3$ in which node $v_0$ is counted only once. We can find that, in the situation of Lemma 11, there is at most one node combined state for each possible value of $k_i$. This is because every piece of information travels along the shortest path between the transmission node and destination node, every piece of information records the traversal order of a node only when the node is travelled for the first time and for each selected information record, every node manager only transmits information to the nodes which are not included by the information record. Then, there will be at most $n - l_i - 1$ information records for all the possible information which arrives at node $v_i$ via node $v_{i+1}$.

**Lemma 12.** For the graph G in Figure 1, a piece of information arrives at node $v_i$ via node $v_{i+1}$ and travels node $v_j$ ($j<i$). If there are $l_j$ intermediary nodes in the clockwise path from the start node $v_0$ to node $v_j$, the traveling node number of the information is no less than $n - l_i + l_j + 1$.

**Proof** because there are $l_j$ intermediary nodes in the clockwise path from the start node $v_0$ to node $v_j$, then the information travels at least $l_i + 2$ nodes including node $v_0$ when it arrives at node $v_j$ along the clockwise direction. At the same time, if the information arrives at node $v_i$ via node $v_{i+1}$, it has to travel the counter-clockwise path whose node number is $n - l_i$ including node $v_0$. Consequently, the total traveling node number of the information is no less than $n - l_i + l_j + 1$ in which node $v_0$ is counted only once. We can find that, in the situation of Lemma 12, there is also at most one node combined state for each possible value of $k_i$. This is also because every piece of information travels along the shortest path between the transmission node and destination node, every piece of information records the traversal order of a node only when the node is travelled for the first time and for each selected information record, every node manager only transmits information to the nodes which are not included by the information record. Then, there will be at most $l_i + 1$ information records for all the possible information which arrives at node $v_i$ via node $v_{i+1}$.

Considering the analyses of Lemma 11 and Lemma 12, we can conclude that node $v_i$ has at most $n$ information records. Because graph G has $n$ nodes, the total number of information records in the generic algorithm is $O(n^2)$. Then we can conclude Theorem 10 is valid.

For each information records, a node manager transmits at most $n - 1$ pieces of information. Given the $n - 1$ pieces of information which are transmitted by the traveling salesman at the start node, the total frequency of information transmission in the algorithm is at most $(n - 1)(n^2 - n + 1)$ which is less than $n^3$. Then the total frequency of information transmission in the generic algorithm is $O(n^3)$.

In the generic algorithm, when a piece of information is transmitted, it will directly confirm the traversal state which it will pass through if it arrives at the
destination node. The traversal state includes the traversal sequence of the nodes, the number of traversal nodes, the traversal state id and the total path distance, \( d_{st} \). In these processes, we need \( O(n) \) operations. Besides, the information will try to reset the information record at the destination node. In the process, the information will traverse the AVL tree of the destination node to find if there is AVL tree node whose state data is equal to the traversal state id of the information. If there is such an AVL tree node, the information will compare its distance with the distance of the information record which the information pointer of the AVL tree node points to. If the distance of the information record which the information pointer of the AVL tree node points to is shorter, the algorithm does nothing. Then the distance comparison and traversal state recording only need \( O(1) \) operations. Conversely, if the distance of the information record which the information pointer of the AVL tree node points to is longer, the corresponding information record will duplicate the traversal sequence of the information and record the distance. Then the distance comparison and traversal state recording need \( O(n) \) operations. We can also make the information pointer of the AVL tree node point to the arriving information instead of duplicating the traversal sequence. Then we should guarantee that the arriving information will be inserted into the same position of the comparing information record in the send_list. At the same time, we should release the storage space of the comparing information record. In the situation, the distance comparison and traversal state recording basically need \( O(1) \) operations. If there is not such an AVL tree node, the algorithm will establish a new AVL tree node whose information pointer points to the arriving information and insert it into the appropriate position. At the same time, the information will be inserted into the tail of send_list as a new information record.

In the algorithm, the inserting and removing of doubly linked list needs \( O(1) \) operations for an information record. And because every node has at most \( 2^{n/2} \) information records, the height of every AVL tree is no more than \( n \). As a result, the AVL tree searching and inserting only need \( O(n) \) operations [Adelson-Velsky and Landis 1962]. Based on the aforementioned analyses, we can conclude that we need \( O(n) \) operations to deal with each piece of information.

In the algorithm, an information record of a node is established when the first corresponding information is transmitted. Since the algorithm directly records the first corresponding information as the information record, the establishment cost of an information record has actually been included by the cost of information transmission.

**Theorem 13.** For an information record which corresponds to a traveling node set, it will need at most \( \frac{P_{\max}}{P_{\min}} + 1 \) \( (P_{\max} \) is the maximum \( P_{ij} \) in graph \( G \)) comparison frequencies from the time when it is established to the time when the holding node transmits information for it in the generic algorithm.

**Proof** For arbitrary node \( v_j \) \( (v_j \in V) \), this paper adopts doubly linked list to store the information records which the node manager has not transmitted information for. Every information record in the doubly linked list will be checked to compare the traversal distance with \( d_{st} \) and judge if the node manager should transmit information for it in every round loop. We suppose that at \( t_0 \) \( (t_0 \geq 1) \) round loop, node \( v_i \) \( (v_i \in V) \) transmits a piece of information to node \( v_j \) for its information record whose traversal distance is \( d_{ii} \) \( ((t_0 - 1) \ast v_{step} < d_{ii} \leq t_0 \ast v_{step}) \) and establishes a
corresponding information record, \( r_j \) at node \( v_j \). The traversal distance of \( r_j \) is \( d_{sj} (d_{sj} = d_{si} + P_{ij}) \) where \( P_{ij} \) is the shortest traversal distance between node \( v_i \) and \( v_j \) \((v_i, v_j \in V)\). Based on Theorem 8, we know that node \( v_i \) transmits information for \( r_j \) and removes \( r_j \) from the record linked list only when \( d_{sj} = d_{j\text{min}} \), where \( d_{j\text{min}} \) represents the shortest traversal distance which the traversal distance of \( r_j \) can be. Based on the enactment of the algorithm, we know that \( d_{j\text{min}} \leq d_{sj} + P_{ij} \). We use \( t_1 \) to represent the round number of the loop when node \( j \) removes \( r_j \) from the record linked list. Then \( t_1 \) satisfies the condition of \((t_1 - 1) \cdot v_{\text{step}} \leq d_{j\text{min}} \leq t_1 \cdot v_{\text{step}} \). And the total time of \( r_j \) in the linked list, \( t_1 \cdot t_0 \) is less than \( \frac{d_{j\text{min}} - d_{si}}{v_{\text{step}}} + 1 \leq \frac{P_{ij}}{v_{\text{step}}} + 1 \leq \frac{P_{\text{max}}}{v_{\text{step}}} + 1 \). For reducing the time complexity, we set \( v_{\text{step}} \) as \( P_{\text{min}} \) which is the minimum \( P_{ij} \) in the graph \( G \). As a result, \( t_1 \cdot t_0 \) is less than \( \frac{P_{\text{max}}}{P_{\text{min}}} + 1 \). Then in the record linked list, the comparing frequency of an information record is at most \( \frac{P_{\text{max}}}{P_{\text{min}}} + 1 \). And Theorem 13 is valid.

Based on aforementioned analyses, we can conclude that the time complexity interval of the generic algorithm is \((O(n^4 + (\frac{P_{\text{max}}}{P_{\text{min}}} + 1)n^2), O(n^2 \cdot 2^n + (\frac{P_{\text{max}}}{P_{\text{min}}} + 1)n^2))\). In the generic algorithm, the space complexity is mainly determined by the number of information record. Based on Theorem 9 and Theorem 10, we can conclude that the space complexity interval of the algorithm is \((O(n^3), O(n^2 \cdot 2^n))\), where each information record need \( O(n) \) storage space. In reality, the generic algorithm of this paper is also applicable to the traveling salesman problem in a directed graph whose edge weights are all positive. And the time complexity and space complexity are the same.

7. THE OPTIMIZATION ALGORITHM USING FIBONACCI HEAP

In the generic algorithm, we assume that all the information spreads at the same speed \( v_{\text{step}} \) \((0 < v_{\text{step}} \leq W_{\text{min}})\). Then in the execution process of the generic algorithm, if the minimal traversal distance of all the existing information records is \( d_{\text{min}} \) after one round information transmission, the next round loop \( t_0 \) when there is some information record transmitting information must satisfy the condition of \((t_0 - 1) \cdot v_{\text{step}} < d_{\text{min}} \leq t_0 \cdot v_{\text{step}} \) \((t_0 > 0)\). Based on Theorem 8, we know that if we assume that all the information spreads at the same speed \( v_{\text{step}} \) \((0 < v_{\text{step}} \leq W_{\text{min}})\), the information records whose traversal distances are in the same interval \((t_0 - 1) \cdot v_{\text{step}} < d_{\text{min}} \leq t_0 \cdot v_{\text{step}} \) \((t_0 > 0)\) can transmit information at the same clock, namely, in the same round loop and it does not affect the correctness of the final result. Based on the reality, we can optimize the generic algorithm using Fibonacci heap.

In the optimization algorithm, we establish a Fibonacci heap based on all the existing information records. In the Fibonacci heap, every node corresponds to one existing information record of the optimization algorithm. We use \( H \) to represent the Fibonacci heap, use \( \text{min}[H] \) to represent the pointer of the heap and use \( \text{min-node} \) to
represent the node which \( \min[H] \) points to in \( H \). We use \( k_{\min} \) to represent the key value of \( \text{min-node} \) which \( \min[H] \) points to and use \( \text{child}[x] \) to represent the pointer which points to the child node list of node \( x \). We add a Boolean quantity, flag as a new attribute of \( H \). The process in every round loop can be described as follows:

Fib-Extract-Min-and-Transmit-Information \((H, v_{\text{step}})\)

(i) If \( \min[H] \) is equal to NULL, go to (ix). Otherwise, if flag is false, go to (iii), else if \( v_{\text{step}} \) is equal to zero, \( y = k_{\min} \), otherwise, \( y = \left\lfloor k_{\min} / v_{\text{step}} \right\rfloor \times v_{\text{step}}, \) if \( y < k_{\min} \), \( y = y + v_{\text{step}} \).

Then go to (iii);

(ii) If the key value of the node, \( \text{min-node} \) which \( \min[H] \) points to is no more than \( y \), go to (iii), else go to (viii);

(iii) Transmit information according to \( \text{min-node} \) and the algorithm needs either some insert operations or some decrease key operations which are both the basic operations of Fibonacci heap. In the process, every node which is added to the root list of \( H \) will be added to the front of \( \text{min-node} \), namely, the node will be the left brother of \( \text{min-node} \).

(iv) If flag is false, go to (vi). Otherwise, if \( \text{child[\text{min-node}]} \) is not equal to NULL and the key value of the node, \( z \) which \( \text{child[\text{min-node}]} \) points to is no more than \( y \), go to (v), else go to (vi);

(v) Remove node \( z \) from the child list of \( \text{min-node} \) and add it to the back of \( \text{min-node} \), namely, node \( z \) will be the right brother of \( \text{min-node} \). At the same, if node \( z \) has a right brother, make the \( \text{child[\text{min-node}]} \) point to the right brother of node \( z \). Then go to (iv). If node \( z \) has no right brother, set \( \text{child[\text{min-node}]} \) as NULL and go to (vii);

(vi) If \( \text{child[\text{min-node}]} \) is not equal to NULL, directly remove all the children of \( \text{min-node} \) from the child list and add them to the front of \( \text{min-node} \), namely, the last node of the child list will be the left brother of \( \text{min-node} \) if we regard the node which \( \text{child[\text{min-node}]} \) points to as the first node of the child list. In the process, every child node does not compare its key value with \( y \);

(vii) Remove \( \text{min-node} \) from the root list of \( H \). If there is a right brother of \( \text{min-node} \), make \( \min[H] \) point to the right brother. If there is not a right brother, set \( \min[H] \) as NULL. If \( \min[H] \) is not equal to NULL and flag is true, go to (ii), else go to (viii);

(viii) If \( H \) is not empty, merge the root list of \( H \) to reduce the tree number and make \( \min[H] \) point to the node whose key value is the minimum. This process also belongs to the basic operation process of Fibonacci heap.

(ix) If \( \min[H] \) is not equal to NULL and either the key value of \( \text{child[\text{min-node}]} \) or the key value of the right brother of \( \text{min-node} \) is equal to \( k_{\min} \), set flag as true, else set flag as false.

In the end of every round loop, the optimization algorithm will judge whether checking the child nodes and the right brothers of \( \text{min-node} \) in the next round loop. If the algorithm finds a new node which satisfies the condition of transmitting information in the next round loop, then check the child nodes and right brothers in the next round loop. Otherwise, execute the next round loop directly and only deal with the \( \text{min-node} \). In the process of checking the child nodes and the right brothers of \( \text{min-node} \), the optimization algorithm will check the child nodes of \( \text{min-node} \) to judge whether the child nodes satisfy the condition of transmitting information until it finds a child node which does not satisfy the condition. Besides, the optimization algorithm will also traverse the root list to find the root nodes which satisfy the condition of transmitting information until it finds a root node which does not satisfy the condition. The reason for the design as above is to make the optimization algorithm possible to deal with more than one node in one round loop and the
checking operations will not take too much time in the worst case that the optimization algorithm deals with only one node in one round loop. If the optimization algorithm can deal with more than one node in one round loop, the round number of the loop of the optimization algorithm will decrease and the scale of the Fibonacci heap could be smaller in the subsequent loops. The direct result is the time complexity can be lower than that of the algorithm using Fibonacci heap which only deals with one node in one round loop.

In the worst case, the optimization algorithm deals with only one node in one round loop. Based on Theorem 9, we know that there are at most \( n \cdot 2^n \) information records which the optimization algorithm need to deal with. Then the time complexity of the optimization algorithm is \( O(n^3 \cdot 2^n) \), namely, \( O(n^3 \cdot 2^n) \) which is better than that of the general algorithm in the same situation when \( P_{\text{max}} / P_{\text{min}} \) is large enough. In the situation of Theorem 10, there are at most \( n^2 \) information records which the optimization algorithm need to deal with. Then the time complexity of the optimization algorithm is \( O(n^4 + n^2 \cdot 2^k) \) where the optimization algorithm pay \( O(2n^2) \) time to judge whether checking the child nodes and the right brothers of \( \text{min-node} \) in the next round loop, namely, \( O(n^4) \) which is better than that of the generic algorithm in the same situation when \( P_{\text{max}} / P_{\text{min}} \) is large enough. Based on the aforementioned analyses, we can conclude that the time complexity interval of the optimization algorithm is \( (O(n^4), O(n^3 \cdot 2^n)) \) which is better than that of the generic algorithm when \( P_{\text{max}} / P_{\text{min}} \) is large enough.

In reality, the criterion that is used to judge whether checking the child nodes and the right brothers of \( \text{min-node} \) in the next round loop of the optimization algorithm is very conservative. That is in order to minimize the cost of computing the value of \( y \) in the worst case that the optimization algorithm only deals with one node in one round loop. Similarly, the criterion that is used to terminate the checking operations in one round loop is also very conservative. That is in order to reduce the cost of profitless checking operations in the worst case. The result is that the optimization algorithm has lower time complexity in the worst case, but has less possibility to perform better than the algorithm using Fibonacci heap which only deals with one node in one round loop. The design selection is a kind of trade-off. However, based on the analysis of the complexity of Algorithm 1, we can deduce that when \( P_{\text{min}} \) is larger than zero, the less the value of \( P_{\text{max}} / P_{\text{min}} \) is, the more nodes (information records) could transmit information in one round loop. Based on the property, we can revise the optimization algorithm. If the \( P_{\text{min}} \) is larger than zero, the less the value of \( P_{\text{max}} / P_{\text{min}} \) is, we make the optimization algorithm have more possibility to check the child nodes and the right brothers of \( \text{min-node} \) in the next round loop and make the optimization algorithm check more nodes which do not satisfy the condition of transmitting information before terminating the checking operations. The design revision can make the optimization algorithm more possible to perform better than the algorithm using Fibonacci heap which only deals with one node in one round loop.
and have fewer risks to perform very badly. The result is that the revised optimization algorithm can have better average performance than before.

We use $b$ to represent the number of the nodes which the optimization algorithm check but do not satisfy the condition of transmitting information before terminating the checking operations and use $C$ to represent the number of the information records which the optimization algorithm needs to deal with. Because the root list and the child list of min-node both have at most $\lg C$ nodes [Fredman and Tarjan 1987] at the beginning of every round loop, $b$ should be located in the interval $[1, \lg C]$. If we make the optimization algorithm check the root list and the child list of min-node in every round loop, the algorithm will pay at most $O(2bC)$ time which is no more than $O(2C\lg C)$ in the worst case that the optimization algorithm only deals with one node in one round loop. Then time complexity of the optimization algorithm is still $O(Cn^2+ClgC)$ which is also acceptable and is almost the same as that (i.e. $O(Cn^2+ClgC))$ of the algorithm using Fibonacci heap which only deals with one node in one round loop. Consequently, if we know the value of $P_{\text{max}} / P_{\text{min}}$, we can daringly adopt suitable $b$ to pursue better performance.

8. REMARKS

**Definition 3** (NP-C polynomial solution graph). G=(V,E) is an undirected graph and $n \geq 2$ is the node number of V. If the traveling salesman problem in graph G can be solved in $O(f(n))$ time where $f(n)$ is a polynomial of $n$, we name graph G as an NP-C polynomial solution graph.

**Definition 4** (k-NP-C polynomial solution graph). G=(V,E) is an undirected graph and $n \geq 2$ is the node number of V. The traversal distance of the shortest circle traversal sequence is $d_{\text{emin}}$. And $G'$ is the minimum shortest path graph of G. In the graph $G'$, for arbitrary two nodes $v_i$ and $v_j$ ($v_i, v_j \in V$), the number of the path between $v_i$ and $v_j$ whose traversal distance is no more than $d_{\text{emin}}$ is $N_{ij}$. We use $N_{max}$ to represent the maximum $N_{ij}$ in graph $G'$. If $N_{max} \leq cn^k$ ($c>0$), the algorithm of this paper can solve the traveling salesman problems of the graph $G'$ and G in $O(n^{k+3})$ time. Then we name graph G as a k-NP-C polynomial solution graph.

For an undirected graph which has $n \geq 2$ nodes, the less the $N_{max}$ is, the less the time complexity of the optimization algorithm usually is. Consequently, for reducing the runtime of the algorithm, we use a modified Dijkstra’s algorithm to compute the shortest path between two given nodes. In the algorithm, if a node finds that there is a new path whose distance is no more than the currently holding distance, the node will reset the path precursor node and the currently holding distance based on the state of the new path. It means that every node will record the path precursor node which belongs to the shortest path that has more intermediary nodes in the modified Dijkstra’s algorithm. We run the algorithm of this paper to solve the traveling salesman problem based on the computing result of the modified Dijkstra’s algorithm which ascertains the memory contents of shortestpath[N][N][2]. It is equivalent to running the algorithm of this paper in a shortest path graph of the supergraph, which may have fewer edges and less $N_{max}$. The direct result is that the running time of the algorithm can be less.

Based on the algorithm of this paper, we can conclude that not all the instances of traveling salesman problem need exponential time to compute the optimal solution. Because the decision version of the traveling salesman problem is an NP-complete
problem which R. M. Karp, E. L. Lawler, and R. E. Tarjan (see [Karp 1972]) have proved, then we can also conclude $\text{NPC} \cap P \neq \emptyset$. However, since the algorithm of this paper also has an exponential worst-case running time, it is still an open problem that whether $\text{NP}$ is not equal to $\text{P}$. The property of the algorithm of this paper makes the boundary between deterministic polynomial solvable space and deterministic non-polynomial solvable space more indistinct. Then one part of the future work is to identify the boundary accurately.

9. CONCLUSION

This paper proposes a new exact algorithm for traveling salesman problem. The algorithm is a general exact algorithm for traveling salesman problem in real life. Namely, it can be used to solve an arbitrary instance of traveling salesman problem in real life. The time complexity interval of the algorithm is $(O(n^4), O(n^4 \cdot 2^n))$. It means that for some instances, the algorithm can find the optimal solution in polynomial time although the algorithm also has an exponential worst-case running time. Based on the property, the algorithm of this paper can not only assist us to solve traveling salesman problem better, but also can assist us to deepen the comprehension of the relationship between $\text{NP}$-complete and $\text{P}$. Therefore, it is considerable in the further research on traveling salesman problem and $\text{NP}$-hard problem.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (No.61170164, and No. 61472079), the Funds for Distinguished Young Scholars of Jiangsu Province of China (No.BK2012020), and the Program for Distinguished Talents of Six Domains in Jiangsu Province of China (No.2011-DZ023).

REFERENCES

Georgy M. Adel'son-Vel'skiĭ and Evgenii M. Landis. 1962. An algorithm for the organization of information. Doklady Akademii Nauk SSSR 146: 263–266.

David Applegate, Robert Bixby, Vašek Chvátal and William Cook. 1998. On the Solution of Traveling Salesman Problems. Doc. Math. J. DMV Extra Volume ICM III, 645-656.

William Cook and M. Hartmann. 1990. On the Complexity of Branch and Cut Methods for the Traveling Salesman Problem. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. Vol. 1: 75-81.

George B. Dantzig, Delbert R. Fulkerson, and Selmer M. Johnson. 1954. Solution of a large-scale traveling salesman problem. Operations Research 2, 393–410.

Michael L. Fredman and Robert Endre Tarjan. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM. Vol. 34, 3(7), 596-615.

Michael Held and Richard M. Karp. 1962. A Dynamic Programming Approach to Sequencing Problems. Journal of the Society for Industrial and Applied Mathematics 10(1): 196–210.

Richard M. Karp. 1972. Reducibility among Combinatorial Problems. In R. E. Miller and J. W. Thatcher (editors). Complexity of Computer Computations. New York: Plenum. 85–103.

Manfred Padberg and Giovanni Rinaldi. 1987. Optimization of a 532-city symmetric travelling salesman problem by branch and cut. Operations Research Letters. Vol. 6, 1(1987), 1-7.

Gerhard J. Woeginger. 2003. Exact Algorithms for NP-Hard Problems: A Survey. Combinatorial Optimization (Edmonds Festschrift), Lecture notes in computer science, vol. 2570, Springer, 185–207.