**Analysis of the Traveling Salesman Problem**

XIN JIN

*(Computer Science and technology, Nanjing University, Nanjing, China, 210046)*

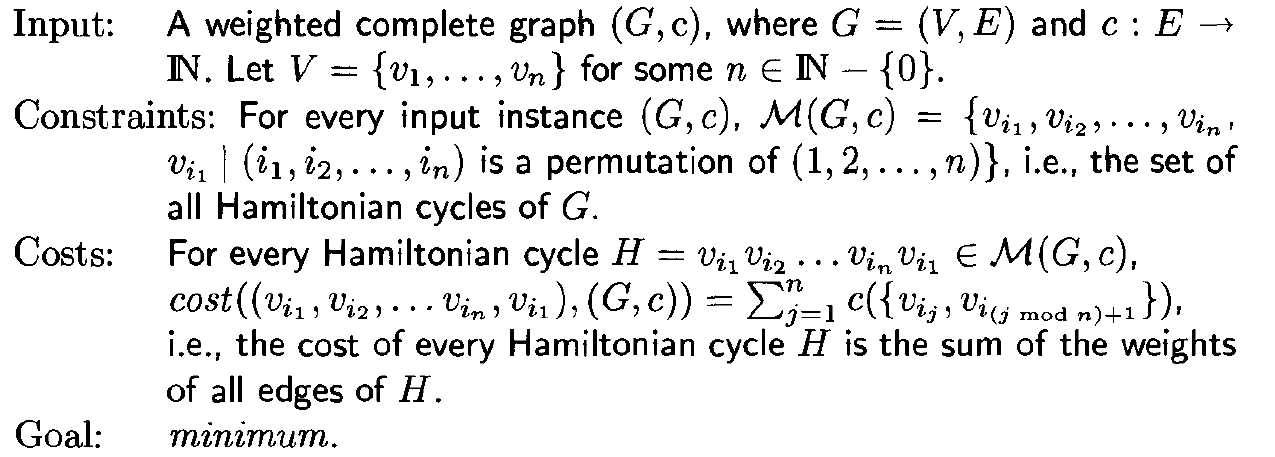
**Abstract**

Traveling salesman problem is a typical optimization problem: Given a list of cities and the distances between each pair of cities, calculate the shortest possible route that visits each city exactly once and returns to the origin city. In this paper, I use C++ to design four kinds of algorithm to solve TSP under different accuracy and using several test files to compare the advantages and disadvantages of each algorithm.

**§1. Introduction**

The Traveling Salesman Problem (TSP), an NP-hard problem, is one of the best known and most fundamental problems in combinatorial optimization. The solution method for TSP has important reference value to solve complex engineering optimization problems. However, there is no completely valid algorithm for TSP which prompts people to continue to explore for good algorithms. To conclude, the main algorithms at present can be divided into traditional optimization algorithms and modern optimization algorithms. The traditional optimization algorithms includes optimal solution algorithm and approximation algorithm. And in this paper, I will introduce two approximation algorithm (MSP and MM) and a greedy algorithm and the improvements I do is to combine the two types of algorithm to get better results. For modern optimization algorithm, there are simulated annealing algorithm (SA), genetic algorithm (GA), Tabu Search Algorithm (TS) and neural network algorithm through which I will only introduce simulated annealing algorithm. At the end of the paper, I will use several test files (berlin52.tsp and krobl00.tsp) [1] to compare the advantages and disadvantages of each algorithm.

**§2.Problem description**

The travelling salesman problem was mathematically formulated in the 1800s by the Irish mathematician [W. R. Hamilton](http://en.wikipedia.org/wiki/William_Rowan_Hamilton) and by the British mathematician [Thomas Kirkman](http://en.wikipedia.org/wiki/Thomas_Kirkman). TSP asks the following question: Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city? The formal definition in mathematical language[2] is that: 

**§3. Algorithms introduce and design**

For any polynomial time computable function α(n)，TSP cannot be approximated within a factor of α(n), unless P=NP. Thus, the following algorithms are designed to solve metric TSP. To show TSP cannot be approximated, I will give a brief proof[3]:

Assume, for a contradiction, that there is a factor α(n) polynomial time approximation algorithm, *A ,* for the general TSP problem. It can be shown that A can be used for deciding the Hamiltonian cycle problem (which is NP-hard) in polynomial time, thus implying P=NP.

The central idea is a reduction from the Hamiltonian cycle problem to TSP, that transforms a graph G on n vertices to an edge-weighted complete graph G' on n vertices such that

* If G has a Hamiltonian cycle, then the cost of an optimal TSP tour in G' is n, and
* If G does not have a Hamiltonian cycle, then an optimal TSP tour in G' is of cost >α(n).

Assign a weight of 1 to edges of G, and a weight of α(n)\*n to none edges, to obtain G'. Now, if G has a Hamiltonian cycle, then the corresponding tour in G' has cost n. On the other hand, if G has no Hamiltonian cycle, any tour in G' must use an edge of cost α(n)\*n, and therefore has cost>α(n)\*n. Proof is done.

**3.1 A simple factor 2 algorithm based on MSP**

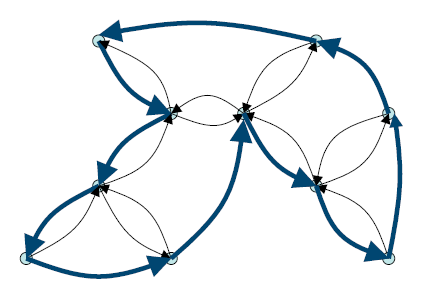
It's very easy to come up with a factor 2 algorithm. The lower bound we will use for obtaining this factor is the cost of an MST in G. This is a lower bound because deleting any edge from an optimal solution to TSP gives us a spanning tree of G.

Factor-2 algorithm:

* Find an MST, T , of G
* Double every edge of the MST to obtain an Eulerian graph.
* Find an Eulerian tour, T\* , on this graph.
* Output the tour that visits vertices of G in the order of their first appearance in T\*. Let C be this tour(a short cut of T\*).

**Analysis：**

As noted above, cost(T)≤OPT. Since T\* contains each edge of T twice, cost(T\*)=2\*cost(T). Because of triangle inequality, after the “short-cutting” step, cost(C)≤cost(T\*). Combining these inequalities we get that cost(C)≤2\*OPT.



**Figure 1**

**3.2 A simple factor 3/2 algorithm based on MSP and Minimum cost perfect matching**

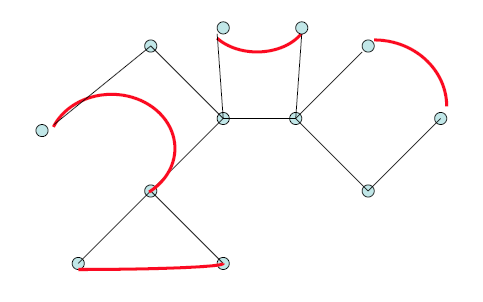
As is known to us, a graph has an Euler tour if and only if all its vertices of ever degrees. Thus, we only need to be concerned about the vertices of odd degree in the MST instead of doubling every edge of the MST[4]. Let V' denote this set of vertices. |V'| must be even since the sum of degrees of all vertices in the MST is even. Now, if we add to the MST a minimum cost perfect matching on V', every vertex will have an even degree, and we get an Eulerian graph. With this modification, the algorithm achieves an approximation of 3/2.

Factor-3/2 algorithm:

* Find an MST, T , of G
* Find a minimum cost perfect matching M among odd degree vertices of T. Add M to T and obtain an Eulerian graph.
* Find an Eulerian tour T\* , on the graph with edges from T and M.
* Output the tour that visits vertices of G in the order of their first appearance in T\*. Let C be the tour(a short cut of T\*).

**Analysis：**

1. As is shown in 3.1’s analysis, we have proved that cost(T)≤OPT. So we just need to prove that cost(M)≤OPT/2. Consider an optimal TSP tour of G, say β. Let β’ be the tour on V’ obtained by short-cutting β. By the triangle inequality, cost(β’)≤cost(β). Now, β’ is the union of two perfect matchings on V’, each consisiting of alternate edges of β’. Thus, the cheaper of these matchings has cost ≤cost(β’)/2≤OPT/2. Hence the optimal matching has cost at most OPT/2. Thus we have cost(C)≤cost(T\*)≤cost(T)+cost(M) ≤3/2OPT.



**Figure 2**

**3.3 Greedy Algorithm**

The main idea of the greedy algorithm is to repeatedly picking the closest unvisited vertex until we have visit all the vertices of the graph and then link the first and last visited vertices. The algorithm is below:

* Start with city i (i=0,1,2,…,n-1)
* Choose the next city with the minimum cost(i,j), j has not been visited before.
* Continue above step, until all cities have been visited.
* Calculate the cost Ci of the cycle.
* Change the start city i , and find the minimum C among Ci.

Another algorithm is to use the idea of Kruskal algorithm for the minimum spanning tree. First, we sort all the edges according to their cost. Then we choose the minimum cost edge from the edges that we have not chosen and do not make a cycle until the last edge we choose and never should a vertex be connected to 3 vertices.

**3.4 simulated annealing algorithm (SA)[5]**

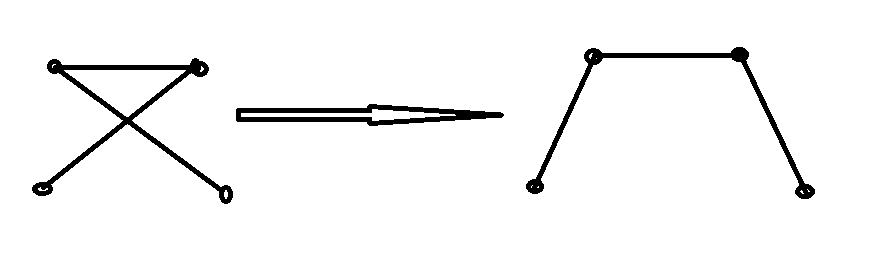
Simulated annealing algorithm is proposed by Kirk Patrick in 1983 and he bring this idea into the field of combinatorial optimization which becomes a method to solve large-scale combinatorial optimization problem, especially for NP-complete combinatorial optimization problem. Simulated annealing algorithm origins from solid annealing principle. According to Metropolis principle, the probability that grain is going to be into balance at temperature T is exp(-E/(k\*T))(T is the internal energy of temperature T, k is Boltzman constant). Using solid annealing method to simulate optimization problems. The internal energy E is modeled as an objective function f, the temperature T evolves control parameter t，getting the simulated annealing algorithm for combinatorial optimization: According to the initial solution I and control parameter t, repeat the present solution for “new solution->calculate the target function difference->accept or discard” and decrease t step by step. The solution is the best approximate solution when the algorithm terminates. So we can get the algorithm according the ideas above to get the simulated annealing algorithm for TSP problem.

* Initialization：initial temperature T(large enough), initial solution(random qualified TSP tour, the cost is written as cost(s)), the repeat times L for every T.
* Do step (3) to step (4) for k=1 to k=L.
* Create a new solution s’ (using 2-opt)
* Calculate the increment cost= cost(s’)-cost(s)
* If t’<0, then let s’ be the new present solution, otherwise, using the probability exp(-t’/T) to accept s’ as the new solution.
* If termination is satisfied, then output the present best solution. (if the loop has been execute for many times without getting new solution, it terminates)
* Decrease T, and T is going to be 0, goto setp (2).

**3.5 Adding a strategy to improve the algorithms above**

According to my observation, I find a strategy to improve the tour that we have found using the algorithm above efficiently. The improve steps I do are shown below:

1. Select two random cities in the present TSP tour.
2. Interchange the two cities predecessors
3. Calculate the weight of resulting tour.
4. If new weight is smaller than old one, replace the old tour by the new one. Continue step 2 until we do the iteration for n^2 times



**Figure3．The Strategy**

**§4. Compare the advantages and disadvantages**

As the time efficient is very nice for the above algorithms, we just compare the accuracy among the algorithms related to the optimization solution for TSP. Using the test file I download from Tsp international standard database, I get the following graphs:

Figure 3

Figure 4

Obviously, the algorithm with my strategy is much better than the original algorithm.

**§5. Conclusion**

In this paper, I mentioned four types of algorithms——Approximation algorithm based on MSP and approximation algorithm based on MSP and MM, the greedy algorithm and the simulated annealing algorithm. In addition, I add a strategy to optimize the algorithms. As the graph above shows, it makes great improvement to the algorithms. However, I haven’t write the program of algorithm MM for which is too hard for me to write, especially the part to calculate the minimum cost perfect matching. But as the study test find on the internet, the analysis of MM implies it gives a much better result than its worst case 1.5OPT. And by the latest study, approximation algorithm of TSP can at most reach 220/219[6].

[1] Tsp international standard database.

[2] Juraj Hromkovic, Algotithmics for Hard Problems—Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics, Springer, 2002.

[3] Vijay V. Vazirani, Approximation Algorithms, Springer, 2001.

[4] Nicos Christofides, Worst-case analysis of a new heuristic for the travelling salesman problem, Report 388, Graduate School of Industrial Administration, CMU, 1976.

[5] Gao YE, Xue Rui, An improved simulated annealing and genetic algorithm for TSP.

[6] http://en.wikipedia.org/wiki/Travelling\_salesman\_problem