TSP (TRAVELING SALESMAN PROBLEM)

By GrMikeD
GrMikeD@Freemail.gr

March 15, 2003

1 Introduction

The *TSP (Traveling Salesman Problem)* is considered as one of the most difficult to solve problems and belongs to the category of the NP Complete problems. The situation it deals with is as follows: A salesman has to visit n cities, each one exactly once. He starts from one of them (base city) and after visiting all the other, he returns back to his base. The transport cost between any pair of cities is known. Notice that it is not necessary that every two cities are connected. We want to find the optimal path for his visits. Optimal path, is the path with the smallest possible cost. This is a famous problem of the graph theory where the nodes of the graph represent the cities, and the edges represent the paths between them. When there is no edge connecting two nodes, it means that there is no way to reach one city from the other. Here, it is assumed that only one path can connect two cities and no more (the graph is simple). For this problem they have been implemented two algorithms. One is the *heuristic* and the other is the *exhaustive*. The heuristic algorithm does not always give the best solution but is very quick and low costing. Contrary to the heuristic algorithm, the exhaustive always gives the best solution but is very slow and resource consuming. For example, for more than 20 cities the exhaustive algorithm run on the most powerful and modern computing system, will spend some centuries until it gives back the solution! In practice, for input n more than 15, the algorithm is very slow.

2 Heuristic Algorithm

This algorithm is based on the 'greedy method'. As it has been mentioned earlier, this algorithm manages to give a fast solution rather than the optimal solution. In some cases, it is possible that it will give us a solution that is much worse than the optimal one. Its basic advantage is that it is very quick. The basic idea of the algorithm is as follows: it start from the base city and after calculating its distance with all its neighboring cities, it selects the city with the smallest distance. It then continues the same way with the next city and all the other until they have been visited all the cities. It should be emphasized here that because it has been assumed that a city might not be connected with another, this algorithm may come across a dead end. In such a case, it finishes unsuccessfully. Lets suppose that the salesman has to visit n cities. For

the first city the algorithm makes (n-1) comparisons (one for each other city). After selecting the second city, it makes other (n-2) comparisons (as the already selected city cannot be selected again) and so goes on until it is reached the (n-1) city and the algorithm finishes. The complexity of the algorithm is:

$$\delta_n = (n-1) + (n-2) + (n-3) + \ldots + 0 + c = \frac{(n^2 - n)}{2} + c$$

The complexity is polynomial O(n^2).

3 Exhaustive Algorithm

Here the algorithm starts from the base city and tries all the possible connection paths that exist. Their costs are calculated and the path with the smallest cost is the output. It is obvious that the number of the possible paths is the number of the permutations of the n cities that is (n-1)! Also, for each of the (n-1)! comparisons 2n arithmetic operations are approximately made. The complexity of the algorithm is:

$$\delta_n = (n-1)! \cdot (2n) + c = 2n! + c$$

The complexity is factorial O(n!).

4 Practical Results

Here are some practical results[1] taken by the TSP Solver program:

	Heuristic			**Exhaustive**		
N	Flop	Time (sec)	Cost	Flop	Time (sec)	Cost
4	13	0,0003	4993	29	0,0005	4993
5	21	0,0002	9254	143	0,0006	9045
6	31	0,0007	11770	839	0,0014	9693
7	43	0,0002	6253	5759	0,0015	6253
8	57	0,0004	13695	45359	0,0036	9286
9	73	0,0004	12322	403199	0,0206	8520
10	91	0,0005	14456	3991679	0,1852	12969
11	111	0,0003	13899	4354559	1,9580	6960
12	133	0,0004	16307	5189183	22,6500	11008

It is clear that for the heuristic algorithm, the flops and time values remain in low levels. In contrast, for the exhaustive algorithm they increase very quickly as the input n value increases. For values of n over 9 the exhaustive algorithm is extremely slow.

[1]The calculation times may vary significantly from one computer to another. For this study, a computer equipped with Intel Celeron 1100 MHz processor and 512 MB SDRAM installed was used.