DYNAMIC PROGRAMMING
TRAVELING SALESMAN PROBLEM

Introduction

Note that permutation problems usually are much harder to solve than subset problems as there are *n!* different permutations of *n* objects whereas there are only 2^n different subsets of *n* objects ($n! > 2^n$). Let G = *(V, E)* be a directed graph with edge costs c_{ij}. The variable c_{ij} is defined such that $c_{ij} > 0$ for all i and j and $c_{ij} = \infty$ if $<i,j> \notin E$. Let $|V| = n$ and assume $n > 1$. A *tour* of G is a directed simple cycle that includes every vertex in *V*. The cost of a tour is the sum of the cost of the edges on the tour. The *traveling salesperson problem* is to find a tour of minimum cost.

The traveling salesperson problem finds application in a variety of situations. Suppose we have to route a postal van to pick up mail from mailboxes located at *n* different sites. An *n*+1 vertex graph can be used to represent the situation. One vertex represents the post office from which the postal van starts and to which it must return. Edge $<i,j>$ is assigned a cost equal to the distance from site i to site j. The route taken by the postal van is a tour, and we are interested in finding a tour of minimum length.

In the following discussion we shall, without loss of generality, regard a tour to be a simple path that starts and ends at vertex 1. Every tour consists of an edge $<1, k>$ for some $k \in V - \{1\}$ and a path from vertex *k* to vertex 1. The path from vertex *k* to vertex 1 goes through each vertex in $V - \{1, k\}$ exactly once. It is easy to see that if the tour is optimal, then the path from *k* to 1 must be a shortest k to 1 path going through all vertices in $V - \{1, k\}$. Hence, the principle of optimality holds. Let *g(i, S)* be the length of a shortest path starting at vertex i, going through all vertices in *S*, and terminating at vertex 1. The function $g(1, V-\{1\})$ is the length of an optimal salesperson tour. From the principal of optimality it follows that

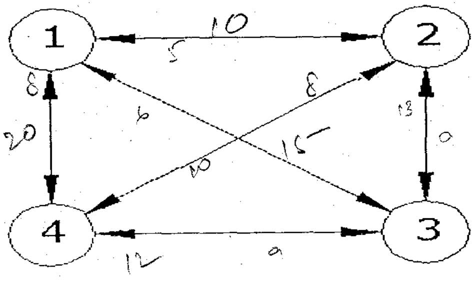
$$g(1, V - \{1\}) = \min_{2 \le k \le n} \{c_{1k} + g(k, V - \{1, k\})\} \quad \text{------------------------- (1)}$$

Generalizing equation (1), we obtain (for $i \notin S$)

$$g(i, S) = \min_{j \in S} \{ c_{ij} + g(j, S - \{j\})\} \quad \text{------------------------- (2)}$$

Equation (1) can be solved for $g(1, V - \{1\})$ if we know $g(k, V - \{1, k\})$ for all choices of k. The g values can be obtained by using equation (2). Clearly, $g(i, \phi) = C_{i1}$, $1 \le i \le n$. Hence, we can use equation (2) to obtain g(i, S) for all S of size 1. Then we can obtain g(i, S) for S with $|S| = 2$, and so on. When $|S| < n - 1$, the values of i and S for which g(i, S) is needed are such that $i \ne 1$, $1 \notin S$, and $i \notin S$.

Example: Consider the directed graph of following Fig (a). The edge lengths are given by matrix c of Fig (b).



(a)

$$\begin{array}{c|c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 8 & 9 & 0 \end{bmatrix} \end{array}$$

(b)

Fig: Directed graph and edge length matrix c

Thus $g(2, \phi) = C_{21} = 5$, $g(3, \phi) = C_{31} = 6$, and $g(4,\phi) = C_{41} = 8$. Using equation (1), we obtain

$g(2, \{3\}) = C_{23} + g(3, \phi) = 9 + 6 = 15$ $\quad$ $g(2,\{4\}) = C_{24} + g(4, \phi) = 10 + 8 = 18$

$g(3, \{2\}) = C_{32} + g(2, \phi) = 13 + 5 = 18$ $\quad$ $g(3,\{4\}) = C_{34} + g(4, \phi) = 12 + 8 = 20$

$g(4, \{2\}) = C_{42} + g(2, \phi) = 8 + 5 = 13$ $\quad$ $g(4,\{3\}) = C_{43} + g(3, \phi) = 9 + 6 = 15$

Next, we compute $g(i, S)$ with $|S| = 2$, $i \neq 1$, $1 \notin S$, and $i \notin S$.

$g(2,\{3,4\}) = \min \{C_{23} + g(3,\{4\}), C_{24} + g(4,\{3\})\} = \min \{9 + 20, 10 + 15\} = 25$

$g(3,\{2,4\}) = \min \{C_{32} + g(2,\{4\}), C_{34} + g(4,\{2\})\} = \min \{13 + 18, 12 + 13\} = 25$

$g(4,\{2,3\}) = \min \{C_{42} + g(2,\{3\}), C_{43} + g(3,\{2\})\} = \min \{8 + 15, 9 + 18\} = 23$

Finally, from equation (1) we obtain

$g(1, \{2, 3, 4\}) = \min \{C_{12} + g(2, \{3, 4\}), C_{13} + g(3, \{2, 4\}), C_{14} + g(4, \{2, 3\})\}$

$= \min \{10 + 25, 15 + 25, 20 + 23\}$

$= \min \{35, 40, 43\} = 35$

So an optimal tour of the graph has length 35. A tour of this length can be constructed if we retain with each $g(i, S)$ the value of j that minimizes the right-hand side of equation (2). Let $J(i, S)$ be this value. Then, $J(1, \{2, 3, 4\}) = 2$. Thus the tour starts from 1 and goes to 2. The remaining tour can be obtained from $g(2, \{3, 4\})$. So $J(2, \{3, 4\}) = 4$. Thus the next edge is <2,4>. The remaining tour is for $g(4, \{3\})$. So $J(4, \{3\}) = 3$. The optimal tour is 1, 2, 4, 3, 1.

Let N be the number of $g(i, S)$'s that have to be computed before (1) can be used to compute $g(1, V - \{1\})$. For each value of $|S|$ there are $n - 1$ choices for i. The number of distinct sets S of size k not including 1 and i is ${}^{n-2}C_k$. Hence

$$N = \sum_{k=0}^{n-2} (n-1)\,{}^{n-2}C_k = (n-1)\,2^{n-2}$$

An algorithm that proceeds to find an optimal tour by using equations (1) and (2) will require $\Theta(n^2 2^n)$ time as the computation of $g(i, S)$ with $|S| = k$ requires k-1 comparisons when solving (1). This is better than enumerating all n! different tours to find the best one. The most serious drawback of this, dynamic programming solution is the space needed, $O(n2^n)$. This is too large even for modest values of n.