JOURNAL OF MATHEMATICAL ANALYSIS AND APPLICATIONS 38, 328–334 (1972)

A Generalized Permanent Label Setting Algorithm for the Shortest Path between Specified Nodes

GEORGE L. NEMHAUSER

Department of Operations Research, Cornell University, Ithaca, New York 14850

Submitted by Richard Bellman

Received October 26, 1970

Given a set of nodes $N = \{0, 1, \ldots, n\}$ and the $(n + 1) \times (n + 1)$ distance matrix $D = \{d_{ij}\}$, where d_{ij} is the length of an arc ij directed from node i to node j, the problem is to find a shortest length path from node 0 to node n. The length of a path is the sum of the arc lengths over the arcs in the path. It is assumed, for ease of exposition, that there is an arc joining every pair of nodes. If, in fact, there is no arc between nodes i and j, let d_{ij} be a very large positive number. Otherwise, the d_{ij} are arbitrary real numbers, except $d_{ii} = 0$ and there are no cycles (i.e., closed paths) of negative length.

The shortest path problem has been studied extensively. Dreyfus [1], in a recent and comprehensive survey, evaluates several methods. An algorithm of Dijkstra [2] is classified by Dreyfus as being the best method for finding a shortest path between specified nodes when $d_{ij} \geqslant 0$ for all i and j, because it has the least upper bound on the number of computations.

Our algorithm is a generalization of Dijkstra's. The upper bound on the number of computations is, in general, larger. However, in many cases, the actual number of computations is smaller and the restriction of $d_{ij} \geqslant 0$ is removed.

PRELIMINARY RESULTS

Let h be a real-valued function with domain $N - \{0\}$ satisfying the property $(*)$

$$h(i) \leqslant h(j) + d_{ij}\,, \qquad \text{for all } i, j \in N - \{0\}. \tag{*}$$

Note that property $(*)$ extends to paths. Consider the path $P = (i_1, \ldots, i_k)$ with length

$$L(P) = d_{i_1 i_2} + \cdots + d_{i_{k-1}, i_k}\,.$$

From property (*), we have

$$\begin{aligned} h(i_1) &\leqslant d_{i_1 i_2} + h(i_2), \\ h(i_2) &\leqslant d_{i_2 i_3} + h(i_3), \\ &\vdots \qquad\qquad \vdots \\ h(i_{k-1}) &\leqslant d_{i_{k-1} i_k} + h(i_k). \end{aligned} \tag{1}$$

Adding all the inequalities of (1) yields

$$h(i_1) \leqslant h(i_k) + d_{i_1 i_2} + \cdots + d_{i_{k-1} i_k}.$$

LEMMA. *The function h exists if and only if there are no cycles of negative length.*

Proof. Suppose there is a closed path or cycle $C = (i_1, \ldots, i_k, i_1)$ of length $L(C) < 0$. Applying (*) to the closed path yields

$$h(i_1) \leqslant h(i_1) + L(C)$$

or $L(C) \geqslant 0$, which is a contradiction.

If there are no negative cycles, a shortest path exists between nodes i and n for every i. Let $f(i)$ be the length of a shortest path between nodes i and n. It is well-known that $f(i) \leqslant f(j) + d_{ij}$ for all $i, j \in N$. Therefore, take $h(i) = f(i)$ for all i. Q.E.D.

THE ALGORITHM

The function h is an input to the algorithm. Its significance will be discussed after the algorithm is given.

Step 0. Initialize $S = \{0\}$, $S \cup R = N$, $S \cap R = \phi$, $d(0) = 0$, $d(j) = d_{0j}$, $j \neq 0$.

Step 1. Let $d(j^*) + h(j^*) = \min_{j \in R}[d(j) + h(j)]$. If $j^* = n$ terminate, $d(n) =$ length of a shortest path from node 0 to node n. Otherwise, let $R = R - \{j^*\}$.

Step 2. For all $j \in R$, let

$$d(j) = \min(d(j), d(j^*) + d_{j^* j}).$$

Return to Step 1.

Clearly, the algorithm is finite, since Step 1 cannot be executed more than n times. Let $\Theta(i)$ be the length of a shortest path from node 0 to node i.

THEOREM. *The choice of j^* in Step 1 is such that $d(j^*) = \Theta(j^*)$.*

Proof. The proof is inductive. Initially, after Step 0, the following propositions are true:

1. For all $i \in S$, $d(i) = \Theta(i)$.

2. For all $j \in R$, $d(j)$ = length of a shortest path from 0 to j that contains no nodes from R except for j itself $\equiv \Theta_S(j)$.

Now assume, as an induction hypothesis, that after $k-1$ executions of Step 1 and 2, the two propositions are valid. We claim that after the k-th execution of Step 1, $d(j^*) = \Theta(j^*)$. Suppose, to the contrary, that $d(j^*) \neq \Theta(j^*)$. By Proposition 2 and the induction hypothesis, $d(j^*) = \Theta_S(j^*) \geqslant \Theta(j^*)$. Therefore, there must exist a path $P = (0, k_1, \ldots, k_t, j^*)$ containing at least one node from R, other then j^*, whose length $L(P) < d(j^*)$. Let k_α be the first node in P that is in R and $L(P) = L(P_1) + L(P_2)$, $P_1 = (0, \ldots, k_\alpha)$, $P_2 = (k_\alpha, \ldots, k_t, j^*)$. Since P_1 contains no nodes from R except k_α, without loss of generality, we can assume $L(P_1) = \Theta_S(k_\alpha)$. By Proposition 2 and the induction hypothesis, $L(P_1) = d(k_\alpha)$. Therefore,

$$L(P) = d(k_\alpha) + d_{k_\alpha k_{\alpha+1}} + \cdots + d_{k_t j^*},$$

and we assume

$$d(k_\alpha) + d_{k_\alpha k_{\alpha+1}} + \cdots + d_{k_t j^*} < d(j^*). \tag{2}$$

From property $(*)$ on h, we have

$$h(k_\alpha) \leqslant h(j^*) + d_{k_\alpha k_{\alpha+1}} + \cdots + d_{k_t j^*}. \tag{3}$$

Adding (2) and (3) yields

$$d(k_\alpha) + h(k_\alpha) < d(j^*) + h(j^*). \tag{4}$$

Since $k_\alpha, j^* \in R$, (4) contradicts the assumption that

$$d(j^*) + h(j^*) = \min_{j \in R}[d(j) + h(j)].$$

Finally, defining $R = R - \{j^*\}$, the length of a shortest path from 0 to $j \in R - \{j^*\}$ that passes through no nodes in $R - \{j^*\}$ except j itself is, from Proposition 2 and the induction hypothesis, given by

$$d(j) = \min(d(j), d(j^*) + d_{j^* j}).$$

This is precisely what is done in Step 2 of the algorithm. Q.E.D.

Discussion

If $d_{ij} \geqslant 0$ for all $i, j \in N$, $h(i) = 0$ for all $i \in N - \{0\}$ satisfies property $(*)$. With $h(i) = 0$ for all $i \in N - \{0\}$, our algorithm is exactly Dijkstra's. The bookkeeping procedures described by Dreyfus for tracing an optimal path in Dijkstra's algorithm apply as well to our algorithm and will not be discussed here. Dreyfus' upper bounds on the number of computations are easily extended to include nonzero $h(i)$—exclusive of any calculations required to determine $h(i)$. The bound on the number of comparisons is still $2n^2$, but the bound on the number of additions increases from $n^2/2$ to n^2 because of the additions $d(j) + h(j)$. The bound is achieved whenever all nodes are labelled permanently (removed from R). Thus, when it is desired to find shortest paths between node 0 and all other nodes, it will be most efficient to take $h(i) = 0$ for all i, provided that $h(i) = 0$ satisfies property $(*)$.

The generalization of Dijkstra's algorithm is also applicable to other permanent label setting shortest path algorithms, e.g., Dantzig's [3]. Is the generalization just a mathematical nicety, or can something be gained by choosing h different from $h(i) = 0$ for all $i \in N - \{0\}$? We can suggest two reasons for using nonzero $h(i)$.

Dijkstra's algorithm, and all other permanent label setting algorithms that I know, require $d_{ij} \geqslant 0$. Our algorithm does not have this restriction and appears to be the first permanent label setting algorithm without it. There are shortest route algorithms that can treat negative d_{ij}, but all of them have upper bounds on the number of computations that involve n^3 (see Dreyfus). Of course, this apparent advantage of our algorithm could be eradicated by the effort required to find suitable $h(i)$. Note that when there are negative d_{ij}, $h(i) = 0$ for all $i \in N - \{0\}$ will not satisfy property $(*)$. We will come momentarily to the question of determining nonzero $h(i)$.

Our second reason for using nonzero $h(i)$ is to decrease the number of computations. With all $h(i) = 0$, the nodes are permanently labeled (removed from R) in order of increasing $\Theta(i)$. Thus, if node n is far away from node 0, Steps 1 and 2 will have to be executed many times. Let $h(n) = 0$, then any $h(i)$ that satisfy $(*)$ have the property $h(i) \leqslant f(i)$, where $f(i)$ is the length of a shortest path from i to n. Thus, the $h(i)$ are lower bounds on the length of a shortest path from i to n. In our algorithm, the nodes are permanently labeled in order of increasing $\Theta(i) + h(i)$. The effect of the $h(i)$ is to hasten the labeling of the nodes close to n and n itself. For example, in finding a shortest route between Baltimore and New York, with all $h(i) = 0$, it would be necessary to go in the wrong direction and label Washington. With good bounds the permanent labeling of Washington could be avoided. Specifically, if n is the furthest node from 0 and $N^1 = \{i \mid \Theta(i) + h(i) > \Theta(n)\}$, then cardinality (N^1) permanent labelings will be eliminated.

The identification of the $h(i)$ as lower bounds on the length of a shortest path from i to n suggests a method for determining the $h(i)$. In many shortest route problems, particularly those associated with finite stage dynamic programs, lower bounds satisfying property $(*)$ can be obtained by relaxing problem constraints. The relaxation is constructed such that the minimization problem at node i has a feasible solution with objective value $h(j) + d_{ij}$, where $h(j)$ is the lower bound at node j.

For example, in one dynamic programming or shortest route formulation of the knapsack problem, a state or a node corresponds to the yet unfilled space of the knapsack and to a subset of items not yet considered. A lower bound on the node is obtained by removing the integer constraints on the items. Let c_t be the unit cost of an item and a_t its unit weight. Given $d_{ij} = \alpha c_k$, α a nonnegative integer, the bounds $h(i)$ and $h(j)$ are obtained from

$$h(i) = \min \sum_{t=1}^{k} c_t x_t$$

$$\sum_{t=1}^{k} a_t x_t = i$$

$$x_t \geqslant 0, \qquad t = 1, \ldots, k$$

and

$$h(j) = \min \sum_{t=1}^{k-1} c_t x_t$$

$$\sum_{t=1}^{k-1} a_t x_t = i - \alpha a_k = j.$$

$$x_t \geqslant 0, \qquad t = 1, \ldots, k-1.$$

It is clear that $h(i) \leqslant h(j) + \alpha c_k$.

In the shortest route (dynamic programming) formulation of the traveling salesman problem, a node (state) corresponds to a subset of unvisited cities. Bounds satisfying $(*)$ can be found by solving an assignment problem on the unvisited cities. Taking this approach yields an algorithm for the traveling salesman problem that is a combination of branch-and-bound and dynamic programming. In fact, as shown by Doulliez [4], branch-and-bound can be imbedded into a dynamic program to reduce the number of states that must be evaluated. Since all finite state, N-stage dynamic programs are shortest route problems, our method can be considered as a generalized branch-and-bound, dynamic programming algorithm. Note that, in the spirit of branch-and-bound, if the graph is not completely connected, $h(j)$ would not have to

be calculated until a node i such that arc ij existed was permanently labeled and bounds could be changed during the course of calculation.

An Example

Consider the problem shown in Fig. 1. Because $d_{12} = -2$, we cannot begin with all $h(i) = 0$. However, by inspection, $h(1) = -2$, $h(j) = 0$ otherwise, satisfies $(*)$. It is desired to find the length of a shortest path between nodes 0 and 5.

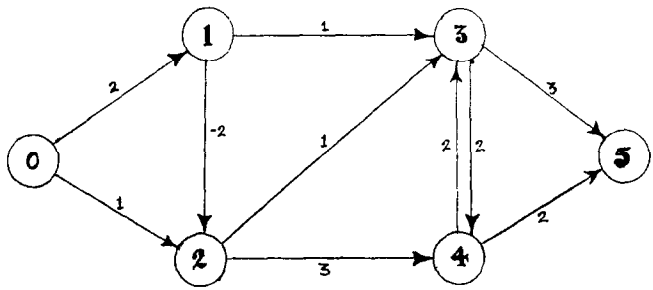

Figure 1

The calculations are given below.

Step 0. $S = \{0\}$, $R = \{1, 2, 3, 4, 5\}$, $d(0) = 0$, $d(1) = 2$, $d(2) = 1$, $d(j) = \infty$ otherwise.

Step 1. $d(1) + h(1) = 0 < d(2) + h(2) = 1$, $j^* = 1$, $R = \{2, 3, 4, 5\}$.

Step 2. $d(2) = \min(1, 2 - 2) = 0$,
$d(3) = 2 + 1 = 3$.

Step 1. $j^* = 2$, $R = \{3, 4, 5\}$.

Step 2. $d(3) = \min(3, 0 + 1) = 1$,
$d(4) = 0 + 3 = 3$.

Step 1. $j^* = 3$, $R = \{4, 5\}$.

Step 2. $d(4) = \min(3, 1 + 2) = 3$,
$d(5) = 1 + 3 = 4$.

Step 1. $j^* = 4$, $R = \{5\}$.

Step 2. $d(5) = \min(4, 3 + 2) = 4$.

Step 1. $j^* = 5$, $d(5) = 4$.

Acknowledgment

I want to thank Guy de Ghellinck for several helpful suggestions.

References

1. S. E. Dreyfus, An appraisal of some shortest-path algorithms, *Operations Res.* **17** (1969), 395–412.
2. E. W. Dijkstra, A Note on Two Problems in Connexion with Graphs, *Numer. Math.* **1** (1959), 269–271.
3. G. B. Dantzig, On the shortest route through a network, *Management Sci.* **6** (1960), 187–190.
4. P. Doulliez, Optimal Capacity Planning of Multiterminal Networks, Doctoral Dissertation, Université Catholique de Louvain, Louvain, Belgium, 1969.