JOURNAL OF MATHEMATICAL ANALYSIS AND APPLICATIONS 3, 547–559 (1961)

Solutions of the *k*th Best Route Through a Network — A Review

MAURICE POLLACK

Stanford Research Institute, Menlo Park, California

Submitted by Richard Bellman

The problem of finding the kth best route through a network is the natural generalization of the shortest-route problem [1]. The general problem can be formulated as follows. A set of N cities (nodes) is given with every two connected by a road (link). Also given are the distances between cities which are not in general symmetric — that is, the distance from node i to node j is not necessarily equal to that from j to i. All distances are assumed to be nonnegative. Where links do not exist between nodes, distances can be thought of as being infinite. Also, the "distance" between any pair of nodes need not be the actual physical distance connecting the nodes. The kth best route from node i to node j is defined as the kth best valued route that does not include any loops.

Solutions of the kth best route have important applications. If, for any reason, the shortest (best) route is unavailable, then alternate routes are desirable. Or, it may be acceptable in some problems to use any route whose length is within, say, 10 percent of the shortest route. Such alternate routes are useful in road traffic studies to determine the routes which motorists might use; another use is in determining alternate message routes in communication networks.

There are several known methods for the solution of the kth best route problem. One of the methods is described in detail, while the others are only described briefly. All but one of them require previous knowledge of the shortest route.

A SHORTEST ROUTE METHOD

In ref. [2], the author describes a simple method which is of value when k is small — that is, second or third best routes. Given the shortest route from node i to j consisting of m links, the length of each link in this route is set, in turn, to "infinity." The shortest-route problem is solved for each such case. The best of these m routes is the desired second best route in the original network. If m is small, the method can be very efficient. The method can be extended to the kth best route but it very quickly becomes computationally overburdening. For example, to obtain the third best route, assume that the second best route consists of n links. Then, if the best and second best routes consist entirely of different links, the maximum number of shortest-route problems to be solved will be the product mn, the best of which will be the third best route. An advantage of this method is that routes with loops will not be found since shortest routes will never have loops. Obviously, this type of solution capitalizes on the "topology" of the network; thus no general estimates can be made as to the number of computations required.

Method of Bock, Kantner, and Haynes

Bock, Kantner, and Haynes [3] describe a method for determining the k best routes using "stems" and "trees." The method is essentially a systematic listing of all routes between a given origin and destination. Hence the shortest route need not be known in advance. After all routes are found, they are ranked. One may also specify an upper limit, and the method will find all routes less than this limit. However, this is only a variation of the general method. This method is limited to small networks; however, it may be the best method in this case.

Method of Hoffman and Pavley

Hoffman and Pavley [4] and [5], describe a method for determining the kth best route from node i (the origin) to node j (the destination) which involves "deviations" from the shortest route. In addition to the shortest route from node i to j, it is necessary to know the value of the shortest route from node i to all other nodes in the network — the shortest-route tree. A shortest-route tree is defined as that set of links which uniquely connects node i to each of the other nodes, such that the route from i to each node is the shortest route. A tree contains no loops; hence, if more than one shortest route exists to any node, only one is used in the tree.

The method depends on the following theorem:

"*The kth best route is a deviation from some rth best route where* $r < k$." A deviation from route P is defined to be a route which coincides with route P from the destination (node j) for a number of links which might be zero, which then contains exactly one link joining a node of path P to another node, and which then proceeds from this other node to the origin (node i) via the shortest-route tree.

The method is illustrated as follows. Given the shortest route from node i to j, one starts at node j and deviates using one link to each of the other nodes in the network. (There is one exception to this rule which is that one does not deviate to the node just preceding node j on the shortest route from i to j; otherwise, one simply obtains the shortest route.) To each node in the network, there is associated a value which is the shortest distance to this node from node i. For each node, then, form the sum of its shortest route value and the length of the deviation link. The next step consists in going backwards on route P — the shortest route — one link to a node b. Now compute for each node in the network (again with one exception) the sum of (1) the value of the shortest route to the node from node i, (2) the value of the deviation link from the node in question to node b, and (3) the distance from b to j. This process is repeated, going back one link at a time, for each node in the best route from i to j. From the totality of sums choose the best one; this will be the value of the second best route. To find the third best route, one must calculate all deviations from the just determined second best route. The third best route will then be the best of either (1) the second best deviation from the best route, or (2) the best deviation from the second best route. To determine the fourth best route, one must first compute all the deviations from the third best and then select the remaining best, and so on. A complete list of deviations must be made at each stage in order to determine the next best route.

It is very important to note that a *k*th best route generated by this method can include one or more loops. These paths may be extraneous solutions to the defined, or physical, problem but they are absolutely necessary in the generation of succeeding deviations, because new routes are produced from older routes by only a one-deviation change. This point is illustrated in Fig. 1.

Assume that the shortest-route tree, based on node A, is shown in Fig. 1(a). Then the shortest route from A to D is $ABCD$. As one determines all deviations from this route, a possible deviation could be (1) go backwards from D to C, (2) deviate to E, and (3) return from E to A via the shortest-route tree — $ECBA$. Hence this deviation from $ABCD$ is the route $ABCECD$ — a perfectly legitimate deviation which includes a loop as shown in Fig. 1(b). It is possible that this particular deviation is the best possible so that it becomes the second best valued route. One of the deviations from $ABCECD$ is the route $ABECD$, shown in Fig. 1(c), which could turn out to be the third best valued route. It is impossible to generate $ABECD$ as a deviation from $ABCD$, since $ABECD$ is two deviations from $ABCD$. However, in this case $ABCECD$ is not acceptable as the second best route. Therefore, $ABECD$ is the second best valued route with no loops.

An advantage of this method is that the "topology" of the network may be such that few deviations need be determined for each stage. A disadvantage is the extra effort that may be needed to generate routes which contain loops. It is not possible to predict how many such routes

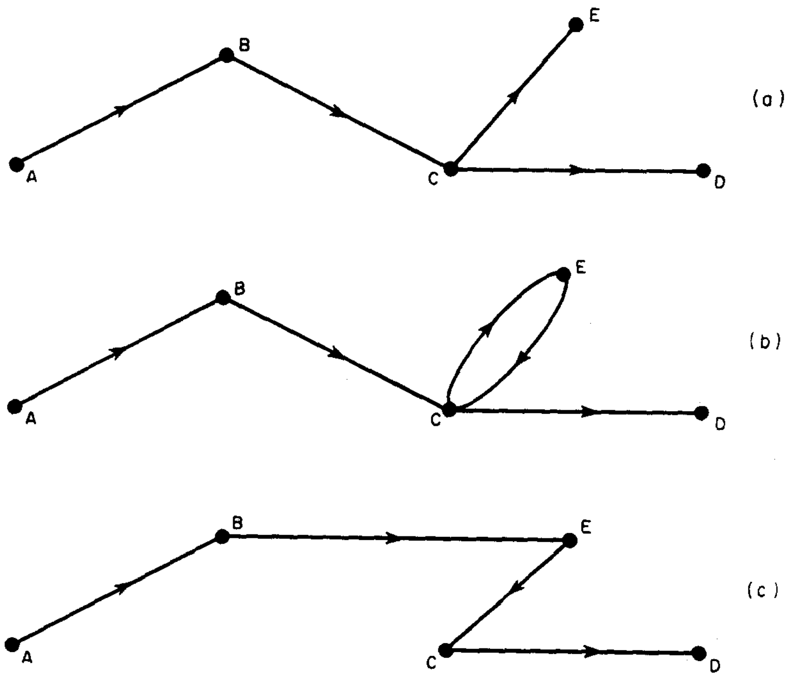

FIG. 1

must be generated before the next nonloop route emerges. Also, there is the problem of equal valued routes. If, for example, there are two shortest routes, then the second best route would be the best deviation from either of them. All equal alternate routes must thus be accounted for at each stage and deviations generated from each of them. Therefore, no general estimate can be made beforehand as to the number of computations required. Perhaps the most important result of this method is the insight obtained about the process of generating successively longer routes.

METHOD OF BELLMAN AND KALABA

Bellman and Kalaba [6] present a method which appears superior to all the others for larger networks. This method determines the kth best route from all nodes to a given node. The method is outlined below, and expanded in detail to describe the actual relations used for calculation. The same notation and equation numbers that are used in [6] will be used here, with the exception that equations numbered (2.9) and higher do not appear in the referenced paper.

$k = 1$

Let t_{ij} be the traversal time, or the distance, of the link connecting node i to j, where $i, j = 1, 2, \ldots, N$, and $t_{jj} = 0$ for all j. The shortest distance is then defined as

$$u_i = \text{the shortest distance from node } i \text{ to } N, \qquad i = 1, 2, \ldots, N-1$$
$$u_N = 0. \tag{2.1}$$

If the shortest route from i to N first passes through node j, then the route from j to N must be the shortest. Therefore, the following system of equations results.

$$u_i = \min_{j \neq i} (t_{ij} + u_j), \qquad i = 1, 2, \ldots, N-1, \tag{2.2}$$
$$u_N = 0.$$

These equations cannot be solved in this form, and the method of solution which is suggested is an iterative procedure using the sequences $\{u_i^k\}$. The sequence $\{u_i^0\}$ is obtained from the given distance matrix.

$$u_i^0 = t_{iN}, \qquad \mathrm{i} = 1, 2, \ldots, N-1, \tag{2.3}$$
$$u_N^0 = 0.$$

Then

$$u_i^1 = \min_{j \neq i} (t_{ij} + u_j^0), \qquad i = 1, 2, \ldots, N-1,$$
$$u_N^1 = 0.$$

And, in general,

$$u_i^{k+1} = \min_{j \neq i} (t_{ij} + u_j^k), \qquad i = 1, 2, \ldots, N-1, \tag{2.4}$$
$$u_N^{k+1} = 0.$$

The iterations terminate either when the sequence $\{u_i^{k+1}\} = \{u_i^k\}$ or when $k + 1 = N - 2$. The maximum number of intermediate nodes between i and N is $N - 2$ and is therefore the longest route that can exist without loops.

$\underline{k = 2}$

This same approach is taken for the second best route. The definitions of v_i are

$$v_i = \text{the second shortest distance from node } i \text{ to } N, \qquad i = 1, 2, \ldots, N-1,$$
$$v_N = 0. \tag{2.5}$$

Also a new notation is introduced

$$\min_n (x_1, x_2, \ldots, x_N) = \text{the } n\text{th smallest } x_i. \tag{2.6}$$

This function may not always be defined, as, for example, occurs if all the x_i are equal, and $n \geqslant 2$.

To obtain the equations connecting the various members of the sequence $\{v_i\}$ the argument is as follows. If the route from i to N is to be the second shortest, then whatever link is used as the first link, the

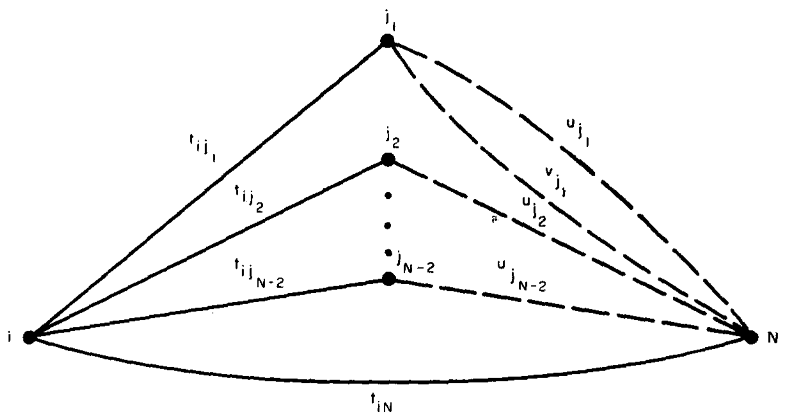

FIG. 2

continuation must be either a shortest route or a second shortest route. This relationship is illustrated in Fig. 2. Node i is a particular node and node N is the common destination. Nodes j_1 through j_{N-2} represent all the other nodes in the network. The solid lines represent links in the network; the dashed lines represent continuing routes of one or more links. The best route from i to N is assumed to have the value $(t_{ij_1} + u_{j_1})$ and thus node j_1 is the first node after node i on this best route. If the first node after node i, of the desired second best route, is one of the nodes $j_2, j_3, \ldots, j_{N-2}$, then the continuation must be a shortest route.

As a special case, if the first node after i is node N, then the continuation is zero. If the first node after i is the node j_1, then the continuation must be the second best route v_{j_1}.

It thus follows that v_i must be equal to one of the expressions

$$\min_{2,\ j\neq i} (t_{ij} + u_j), \qquad (2.7)$$

$$\min_{1,\ j\neq i} (t_{ij} + v_j).$$

Since it must equal the smaller of these, the desired relation is

$$v_i = \min \begin{bmatrix} \min\limits_{j\neq i}{}_2 (t_{ij} + u_j) \\ \min\limits_{j\neq i}{}_1 (t_{ij} + v_j) \end{bmatrix} \qquad \begin{matrix} i = 1, 2, \ldots, N-1 \\ j = 1, 2, \ldots, N. \end{matrix} \qquad (2.8)$$

Once the sequence $\{u_i\}$ has been computed, the sequence $\{v_i\}$ can be determined using successive approximations in the manner outlined above.

The actual iterations proceed as follows. As before, define the sequences $\{v_i^k\}$. The sequence $\{v_i^0\}$ is defined as

$$v_i^0 = \begin{cases} t_{iN}, & \text{if} \quad t_{iN} > u_i, \qquad i = 1, 2, \ldots, N-1, \\ \min\limits_j (t_{ij} + t_{jN}), & \text{if} \quad t_{iN} = u_i, \qquad j = 1, 2, \ldots, N. \end{cases}$$

$$v_N^0 = 0 \qquad (2.9)$$

The variable v_i^0 is the value of the single link from i to N. However, if this coincidentally yields the value u_i, then one uses the best pair of links from i to N.

Using (2.9) as a starting point for the iterations, the next step is to obtain

$$v_i^1 = \min \begin{bmatrix} \min\limits_{j\neq i}{}_2 (t_{ij} + u_j) \\ \min\limits_{j\neq i}{}_1 (t_{ij} + v_j^0) \end{bmatrix} \qquad \begin{matrix} i = 1, 2, \ldots, N-1 \\ j = 1, 2, \ldots, N, \end{matrix} \qquad (2.10)$$

$$v_N^1 = 0.$$

In general

$$v_i^{k+1} = \min \begin{bmatrix} \min\limits_{j\neq i}{}_2 (t_{ij} + u_j) \\ \min\limits_{j\neq i}{}_1 (t_{ij} + v_j^k) \end{bmatrix} \qquad \begin{matrix} i = 1, 2, \ldots, N-1 \\ j = 1, 2, \ldots, N, \end{matrix} \qquad (2.11)$$

$$v_N^{k+1} = 0.$$

When evaluating $(t_{ij} + v_j^k)$, j cannot take on the value $j = N$ if $t_{iN} = u_i$. The iterations terminate when either $\{v_i^{k+1}\} = \{v_i^k\}$, or when $k + 1 = N - 1$. Relations (2.10) and (2.11) are not strictly correct, as it is possible to generate routes with loops. The necessary modifications to avoid this problem are described subsequently.

A brief proof of the validity of this iterative procedure is as follows. Let r_i denote the first node after i, on the shortest route from i to N. If the first node, after node i, on the second best route is not r_i, then the correct value of v_i will be determined at the first iteration. The other

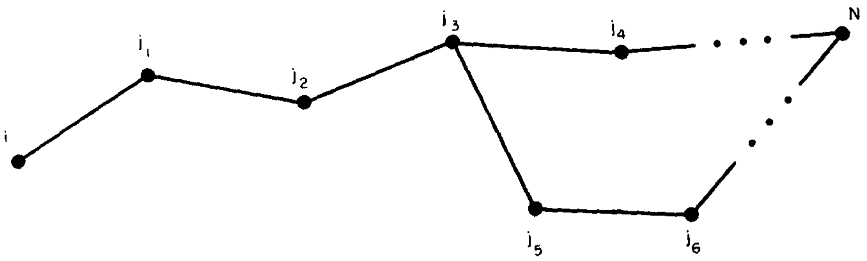

FIG. 3

possibility is that node r_i is the first node, after node i, on the second best route. In this case, the correct value of v_i cannot be determined until v_{r_i} is determined. As soon as v_{r_i} is obtained, v_i will be determined in the next iteration. The correct value of v_{r_i} will be determined at the first iteration if its route's continuation, after the first link, is a shortest route. Otherwise, the correct value of v_{r_i} cannot be determined until its next node's second best value is found, and so on. The second best route, from node i to N, may thus coincide with the best route, from i to N, for a number of links. This is illustrated in Fig. 3. The shortest route, from i to N, is the route $(i, j_1, j_2, j_3, j_4, \ldots, N)$. The correct second best route is $(i, j_1, j_2, j_3, j_5, j_6, \ldots, N)$. The route from j_5 onward is the shortest route from j_5 to N. Thus the correct value of v_{j_3} will be found at the first iteration; the correct value of v_{j_2} will be found at the second iteration; v_{j_1} at the third iteration; and v_i at the fourth iteration.

The final portion of the second best route for a given node will (almost) always be a shortest route for some other node. This relationship is also used in the deviation method of Hoffman and Pavely. As a special case, the second best route could be the direct link from i to N. The iterative procedure allows for this, since j is allowed to take on the value N in relation (2.10). In the iterative procedure, a final value of v_i is determined for at least one value of i at the first iteration. Each subsequent iteration will also determine at least one final value of v_i, for some i. It should be noted that there is an exception to the statement above "that the final portion of any second best route is always a shortest route." The exception occurs if the use of a shortest route, for the final portion, causes a loop to form. In this case a next best final portion is used.

The important question of generating a route with a loop still remains to be answered, since it is possible to generate such a route with the given iterative relations. Loops can be avoided if care is taken at each stage of the iterations. Returning to relation (2.11) it is necessary to define $\min_{2,\, j\neq i} (t_{ij} + u_j)$ as the second best $(t_{ij} + u_j)$ that does not represent a route with a loop. Similarly, it is necessary to define $\min_{1,\, j\neq i} (t_{ij} + v_j^k)$ as the best value of $(t_{ij} + v_j^k)$ that does not contain a loop.

When these definitions are used, the final second best values that are determined will represent routes without loops. However, it is important to note that some of these final values may not be the correct second best values. The term final is used here to denote the value obtained when the iterations are completed for all the second best values. The explanation for this is as follows. It is possible that $(t_{ir_i} + v_{r_i})$ may represent a route with a loop. In this case, the second best value of v_i will be equal to the best value of $\min_{2,\, j\neq i} (t_{ij} + u_j)$ that does not represent a route with a loop. However, the correct second best value v_i could be equal to $(t_{ir_i} + w_{r_i})$, where w_i denotes the value of the third shortest distance from i to N. It is not necessarily true that if $(t_{ir_i} + v_{r_i})$ represents a route with a loop, then the correct second best value is $(t_{ir_i} + w_{r_i})$; but it could be true. In this case, one must determine the values of w_i for each such case before a positive statement can be made as to the correct value of v_i.

One must also be careful in using the value of $\min_{2,\, j\neq i} (t_{ij} + u_j)$. Let s_i denote the first node (node j) after node i, in the route corresponding to the value of $\min_{2,\, j\neq i} (t_{ij} + u_j)$. If this route contains a loop, then one must consider the next best value of j, such that $\min_{2,\, j\neq i} (t_{ij} + u_j)$ does not represent a route with a loop. However, it is possible that the value $(t_{is_i} + v_{s_i})$ is smaller than the next best value of $\min_{2,\, j\neq i} (t_{ij} + u_j)$.

It is possible to write expressions similar to (2.10) and (2.11) which include the proper restrictions to insure that no routes with loops will be generated. However, the resultant expressions would be quite involved. The descriptions given above should be sufficient if one wishes to use this method for actual computation.

$k \geqslant 3$

One must examine the formulation for the third best route before the generalization to the kth best route is apparent. Let w_i denote the value of the third shortest distance from node i to N. As an illustration of the difficulties that arise, assume that for a given node i, the shortest route and the second best route both use the same link as their first link. Then it is true that

$$w_i = \min \begin{bmatrix} \min_2\limits_{j \neq i} (t_{ij} + u_j) \\ \min_1\limits_{j \neq i} (t_{ij} + w_j) \end{bmatrix} \qquad j = 1, 2, \ldots, N. \tag{2.12}$$

If, however, the shortest route and the second best route do not use the same link for their first link, then

$$w_i = \min \begin{bmatrix} \min_3\limits_{j \neq i} (t_{ij} + u_j) \\ \min_1\limits_{j \neq i} (t_{ij} + w_j) \end{bmatrix} \qquad j = 1, 2, \ldots, N. \tag{2.13}$$

Define a new variable for each i.

$$r_{1ij} = (t_{ij} + u_j), \qquad i = 1, 2, \ldots, N-1, \tag{2.14}$$
$$j = \text{all values } 1, 2, \ldots, N \text{ except } i.$$

Eliminate from this collection of $N - 1$ values for each i, all values which represent routes with loops. Also, eliminate all duplicate values. Denote the remaining collection as S_{1i}. Then it is clear that $\min_1 S_{1i}$ is simply u_i, and also that $\min_2 S_{1i}$ is what has been defined as $\min_2\limits_{j \neq i} (t_{ij} + u_j)$. Relation (2.8) for v_i can be written as

$$v_i = \min \begin{bmatrix} \min_2 S_{1i} \\ \min_1\limits_{j \neq i} (t_{ij} + v_j) \end{bmatrix} \qquad \begin{matrix} i = 1, 2, \ldots, N-1 \\ j = 1, 2, \ldots, N. \end{matrix} \tag{2.15}$$

Define another variable r_{2ij}, for each i, as

$$r_{2ij} = (t_{ij} + v_j), \qquad i = 1, 2, \ldots, N-1, \tag{2.16}$$
$$j = \text{all values } 1, 2, \ldots, N \text{ except } i.$$

Eliminate from this collection of $N - 1$ values for each i, all values which represent routes with loops. Combine the remaining values of r_{2ij} with the collection S_{1i}. Eliminate all duplicate values in this new collection and denote the remainder as S_{2i}. As before, $\min_1 S_{2i}$ is simply u_i, but $\min_2 S_{2i}$ is v_i. And $\min_3 S_{2i}$ is one of the candidates for the value of w_i. The other possibility for w_i is the value $\min\limits_{j \neq i}{}_1 (t_{ij} + w_j)$. The proper relation for w_i is then

$$w_i = \min \begin{bmatrix} \min_3 S_{2i} \\ \min\limits_{j \neq i}{}_1 (t_{ij} + w_j) \end{bmatrix} \qquad \begin{aligned} i &= 1, 2, \ldots, N-1, \\ j &= 1, 2, \ldots, N, \end{aligned} \tag{2.17}$$

$$w_N = 0.$$

Relation (2.17) is a proper combination of both (2.12) and (2.13).

The actual iterative calculations are started by defining the sequen

$$w_i{}^0 = \begin{cases} t_{iN}, & \text{if} \quad t_{iN} > u_i \quad \text{or} \quad v_i \\ \min\limits_{i} (t_{ij} + t_{jN}) \text{ such that } w_i{}^0 > u \quad \text{or} \quad v_i, & \text{if} \quad t_{iN} = u_i \quad \text{or} \quad v_i, \end{cases}$$

$$w_N{}^0 = 0 \qquad \begin{aligned} i &= 1, 2, \ldots, N-1, \\ j &= 1, 2, \ldots, N \end{aligned} \tag{2.18}$$

Then the general iteration becomes

$$w_i^{k+1} = \min \begin{bmatrix} \min_3 S_{2i} \\ \min\limits_{j \neq i}{}_1 (t_{ij} + w_j{}^k) \end{bmatrix} \qquad \begin{aligned} i &= 1, 2, \ldots, N-1 \\ j &= 1, 2, \ldots, N. \end{aligned} \tag{2.19}$$

$$w_N^{k+1} = 0$$

As before $(t_{ij} + w_j{}^k)$ is undefined, if for $j = N$, this sum is equal to either u_i or v_i. Also, at each iteration, $\min_1 (t_{ij} + w_j{}^k)$ is defined as the best sum that does not contain a loop.

The generalization of (2.17) to the *k*th best route is now clear. For example, if x_i is the value of the fourth best route from i to N, then

$$x_i = \min \begin{bmatrix} \min_4 S_{3i} \\ \min\limits_{j \neq i}{}_1 (t_{ij} + x_j) \end{bmatrix} \qquad \begin{aligned} i &= 1, 2, \ldots, N-1, \\ j &= 1, 2, \ldots, N, \end{aligned} \tag{2.20}$$

$$x_N = 0$$

where S_{3i} is analogously defined.

The proof of the iterative relations for the above generalization is similar to that given for $k = 2$. Similar definitions (to those for $k = 2$) for the minimizations must be used to avoid determining routes with loops. It is also necessary to examine, for each node i, the manner in which the final value is obtained. If there is a possibility that a final value is not a correct value, then further calculations are needed to obtain the next longer route for each such case.

Final Remarks

It is important to note that none of the given relations yield the actual routes; they only determine the values of the routes. To obtain the route itself, one must resort to a method of "backward subtraction." For example, assume one has computed the correct values of u_i, v_i, w_i, and x_i for all i — based on a particular value of N — where u, v, w, and x represent the values of the first, second, third, and fourth best routes. Consider first the problem of finding the best route from a particular node a to node N. For this problem, it is only necessary to know the value u_i for each node and the distance of each link. Starting from node a, find an adjacent node whose value u is less than u_a by exactly the amount of the connecting link distance. This node is then the next node in a shortest route from i to N. One then repeats this process locating successively closer nodes until node N is reached. There may be more than one best route if at any stage of the subtraction process more than one adjacent node is found that qualifies. Consider now the problem of finding the fourth best route from a to N. It is now necessary to have all the values u_i, v_i, w_i, and x_i for each node, and also the distance of each link. Starting from node a, subtract from the value x_a the distance of a connecting link. Denote the node at the other end of this connecting link j. Compare the result of the subtraction $(x_a - t_{aj})$ with all the values u_j,v_j, w_j and x_j. If one of these values is equal to the value of the subtraction, then node j is on a possible fourth best route leading from node a to node N. Starting with node j, continue the subtraction process until node N is reached. As before, more than one route may be found from node a to N. If more than one fourth best route is found, then each must be examined to see that it does not contain any loops.

As a final point, it is worthwhile to note that this method will also determine the kth best routes from a given node N to all other nodes. Consider the set of primed variables t'_{ij}, where $t'_{ij} = t_{ji}$ for all values of i and j. If the primed variables are used in all the iterations, rather than the unprimed ones, then all the routes obtained will be directed from N to each of the other nodes.

The disadvantage of the Bellman and Kalaba method is that one must generate the kth best route values from all i to a given N, in the process of getting the kth best route between any one i-to-N pair. If one wants the kth best routes from all i to a given N, then this method is far superior to any of the others mentioned. If one wants the kth best route between just one pair of nodes, it is not clear whether one of the other methods is superior or not.

REFERENCES

1. POLLACK, M., AND WIEBENSON, W., Solutions of the shortest-route problem — A review. *Operations Research* 8, No. 2, 224–230 (1960).
2. POLLACK, M., The kth best route through a network. *Operations Research*, 9, No. 4, 578–580 (1961).
3. BOCK, F., KANTNER, H., AND HAYNES, J., An algorithm (The rth best path algorithm) for finding and ranking paths through a network. Research Report, Armour Research Foundation, Chicago, Illinois, November 15, 1957.
4. HOFFMAN, W., AND PAVLEY, R., Applications of digital computers to problems in the study of vehicular traffic. *Proc. Western Joint Computer Conf.*, May, 1958, pp. 159–161.
5. HOFFMAN, W., AND PAVLEY, R., A method for the solution of the Nth best path problem. *J. Assoc. Computing Machinery* 6, No. 4 (1959).
6. BELLMAN, R., AND KALABA, R., On kth best policies. *J. Soc. Indust. Appl. Math.* 8, No. 4, 582–588 (1960).