Finding the shortest route between two points in a network

by T. A. J. Nicholson*

A new method is proposed for finding the shortest route between two points in an interconnected network. The shortest route is found by investigating a selection of routes from both the starting point and the terminal point. The selection of routes is decided dynamically by extending one by one the routes which have currently covered the least distance. Once a complete through route has been found, it has to be made certain that it is the minimum. The new method appears to be more efficient than alternative approaches to the problem through linear or dynamic programming. Some applications of the technique to scheduling and other problems are briefly described.

A simple calculation which is needed in some network applications is the discovery of the shortest route between two points of a network. It could arise for example in a traffic routing problem. Finding the shortest route is simple enough in concept, but in large networks the calculation could become clumsy and much effort could be wasted. The thing to avoid doing is evaluating all alternative routes. This paper describes a simple method of finding the shortest route with a comparatively small amount of calculation. Afterwards a few alternative methods are mentioned, and some applications are suggested.

The network structure

The network consists of a set of, say n, junction points i, $(i = 1, n)$ and a set of distances d_{ij} between pairs of points (i, j). The total number of specified distances will correspond to the number of possible routes between adjacent pairs of points, but there will be at least $n - 1$ such connections, as it is assumed that no point in the network is isolated. Also, in presenting the data for the network, it is assumed the d_{ij} are positive, and that both directions d_{ij} and d_{ji} are given so that different directions are distinguished. The method therefore applies to networks with direction restrictions, or where the distances of the forward and reverse directions may differ, i.e. it may be that $d_{ij} \neq d_{ji}$.

The objective and an outline of the method

The objective is to find the minimum distance route between a starting point, say s, and a terminating point, say t, with the least amount of calculation.

The procedure is to examine simultaneously all the routes out of the starting point s and into the terminating point t as far as their adjacent connected points, and to extend further the route which has so far covered the least distance. This process is repeated stage by stage until a route out of s has a point on it which has already occurred on a route into t, or vice versa. It then remains to check that this complete route is the shortest possible route between s and t.

Graphical illustration of the procedure

The procedure is illustrated on the network of **Fig. 1**. The network has nine points and the distances between the points are marked on the connections. It is assumed that there are no direction restrictions. We want to find the shortest route between the points 1 and 9. We need to go through four steps, and the successive steps are illustrated in Figs. 2 to 5.

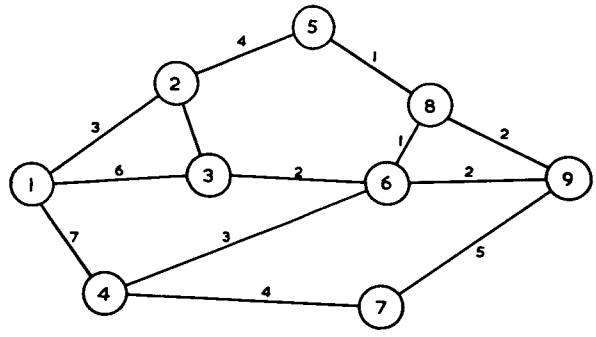


Fig. 1

Step 1. We want to find the routes from point 1 to point 9. We start by examining the routes out of point 1 and into point 9 as far as the adjacent connected points. We can record the distances to these adjacent points in the form (r, x) where r is the adjacent point and x is its distance from point 1 or 9. These results are displayed in **Fig. 2**.

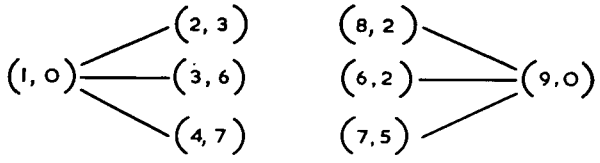


Fig. 2

Step 2. We now look at the results in Fig. 2. Three routes have started out of 1 and three routes into 9. Amongst these six routes the two routes which have so far covered the least distance are the routes 8 to 9 and 6 to 9. They have only covered a distance of 2 units.

* *Applied Mathematics Group, Theoretical Physics Division, U.K.A.E.A., Atomic Energy Research Establishment, Harwell, Didcot, Berkshire.*

Therefore we extend these two routes to their adjacent connected points, and these extensions are displayed in **Fig. 3.** Note that although there is a connection from point 8 to point 6, it is ignored, as its total distance is 3, whereas we already have a route with a distance of 2. In the new (r, x) values the distances are from the base point 9.

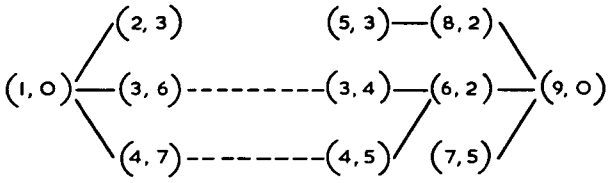


Fig. 3

These new extensions from the point 6 and the point 8 combined with the routes we have already evaluated in step 1 offer two complete routes. They are the routes 1, 3, 6, 9 and 1, 4, 6, 9 and they have total distances 10 and 12 respectively. They are shown as dotted lines in Fig. 3. But we cannot be sure that the route 1, 3, 6, 9 is the shortest route because we have not yet looked into the possibility of a connection between the route from 1 to 2 of distance 3 (the current least distance route from point 1) and the route from 5 to 9 of distance 3 (the current least distance route from point 9). If there was such a connection it might provide a complete route of total distance 6, and as $6 < 10$ we must examine further extensions.

Step 3. Amongst all the routes coming out of point 1 or going into point 1 in Fig. 3 the route 1 to 2 has a length of 3 units and the route 9, 8, 5 also has a length of 3 units. These are the shortest routes, and we first extend the route from 2 to its adjacent connected points. This leads to a revision of the route to a point 3 as the route via the point 2 is shorter. These changes are illustrated in **Fig. 4.**

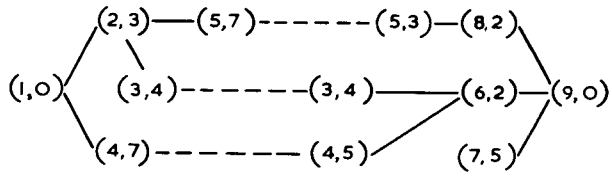


Fig. 4

There are now three complete routes 1, 2, 5, 8, 9; 1, 2, 3, 6, 9; and 1, 4, 6, 9 with distances 10, 8, and 12 respectively. But the minimum distance of an unextended route from 1 has distance 4 and the minimum distance of an unextended route from 9 has distance 3 the minimum possible complete route has distance 7. Therefore we must investigate further the route of 9, 8, 5.

Step 4. If we extend further the route 9, 8, 5, which is the shortest distance route then we obtain no additional new complete routes. The situation is illustrated in **Fig. 5.**

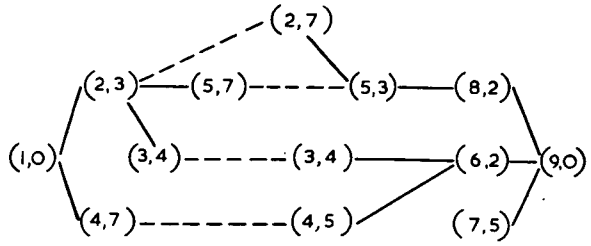


Fig. 5

In this case the minimum distance unextended route from 1 has distance 4 and the minimum distance unextended route from 9 has distance 4. As the minimum distance complete route has distance 8 it is a shortest route. Therefore the shortest distance is 8 and the shortest route is 1, 2, 3, 6, 9.

The notation

The notation which will be used in the algorithm is now described.

n is the number of junction points.

d_{ij} is the distance between the connected points i and j in the direction from i to j. d_{ij} is assumed to be positive.

If there is no connection between i and j then set:

$d_{ij} = 0$, for $i = j$

$d_{ij} = M = n \underset{(i,j)\epsilon S}{\text{Max}} d_{ij}$ where S is the set of specified (i, j) connections.

(The value of M is chosen as a suitably large number so that it is greater than the total distance along any route connecting two points without going through any point more than once.)

s is the starting point

t is the terminating point

$S(i)$ is the current minimum distance from s to i

$T(i)$ is the current minimum distance from i to t

$P(i)$ is the point preceding i in the current optimal route from s to i

$Q(i)$ is the point succeeding i in the current optimal route from i to t

x is the current least distance from s

y is the current least distance from t

The algorithm

The algorithm will be described verbally first, followed by a symbolic description.

The algorithm consists of two steps: the first selects a route for extension and extends the selected route, while the second step provides the check to see if the shortest route has been found. The steps are repeated cyclically until the condition in the second step is satisfied.

The necessary initialization is as follows:

$$\left.\begin{aligned} S(i) &= d_{si} \\ T(i) &= d_{it} \\ P(i) &= s \\ Q(i) &= t \end{aligned}\right\} \text{for all } i$$

(i) *Verbal description of the algorithm*

Step 1. If on any iteration the next shortest route starts from s rather than t, then we proceed as follows:

For each route starting at s of next shortest length, examine all connections from its end point m. If there is a connection from s to the point i via the point m, it will be of length $S(m) + d_{mi}$, and this replaces the existing value of $S(i)$ when it is shorter. When replacement occurs $P(i)$ is set to m. If on any iteration the shortest route starts from t rather than s then we proceed exactly as described above substituting t, $T(m) + d_{im}$, $T(i)$, $Q(i)$ for s, $S(m) + d_{mi}$, $S(i)$, $P(i)$, in the above argument.

Step 2. If on any iteration the least value of $S(i)+T(i)$ (representing a complete route) is at least as small as the sum of the lengths of the shortest route out of s and the shortest route out of t then a shortest complete route has been found. The points on the shortest route can be traced back through the values of $P(i)$ and $Q(i)$.

(ii) *Symbolic description of the algorithm*

Initially $x = y = 0$.

Step 1. If $\underset{S(i) > x}{\text{Min}}\, S(i) \leqslant \underset{T(i) > y}{\text{Min}}\, T(i)$,

then for each m for which $S(m) = \underset{S(i) > x}{\text{Min}}\, S(i)$,

$$\left.\begin{array}{l} \text{reset } S(i) = S(m) + d_{mi} \\ P(i) = m \end{array}\right\}$$

for all i for which $S(i) > S(m) + d_{mi}$.

Also reset x as $\underset{S(i) > x}{\text{Min}}\, S(i)$.

Otherwise:

for each m for which $T(m) = \underset{T(i) > y}{\text{Min}}\, T(i)$

$$\left.\begin{array}{l} \text{reset } T(i) = T(m) + d_{im} \\ Q(i) = m \end{array}\right\}$$

for all i for which $T(i) > T(m) + d_{im}$.

Also reset y as $\underset{T(i) > y}{\text{Min}}\, T(i)$

Step 2. If $\underset{i}{\text{Min}}\,(S(i) + T(i)) \leqslant \underset{S(i) > x}{\text{Min}}\, S(i) + \underset{T(i) > y}{\text{Min}}\, T(i)$

then the shortest distance is $\underset{i}{\text{Min}}\,(S(i) + T(i)) = S(r) + T(r)$

say, and the shortest route is $s, a_{k-1}, \ldots, a_3, a_2, r, b_2, b_3, \ldots, b_{(l-1)}, t$

where $a_1 = r$; $a_i = P(a_{i-1})$, $i = 2, k$; $a_k = s$

and $b_1 = r$; $b_i = Q(b_{i-1})$, $i = 2, l$; $b_l = t$.

Otherwise return to step (2).

Note: if $S(i) \leqslant x$ for all i, then $\underset{S(i) > x}{\text{Min}}\, S(i)$ is taken to be the suitably large number M; and similarly for $\underset{T(i) > y}{\text{Min}}\, T(i)$.

Proof that the algorithm finds the shortest route

The first step of the algorithm, if repeated enough times, ensures that the value of $S(i)$ will monotonically decrease to the length of the shortest distance from s to i. Similarly the value of $T(i)$ will monotonically decrease to the shortest distance from i to t. So that, after enough iterations, if some point k lies on a shortest complete route, then $S(k) + T(k)$ will equal the shortest distance between s and t. But the algorithm states that the iterations can be stopped when the following inequality holds:

$$S(r) + T(r) = \underset{i}{\text{Min}}\,(S(i) + T(i)) \leqslant \underset{S(j) > x}{\text{Min}}\, S(j) + \underset{T(i) > y}{\text{Min}}\, T(k). \tag{1}$$

When this inequality holds many of the values of $S(i)$ and $T(i)$ will not have converged to their final values. It is necessary to show therefore that if $S'(i)$, $T'(i)$, x' and y' are the values of $S(i)$, $T(i)$, x and y when the inequality (1) first holds, and $\bar{S}(i)$, $\bar{T}(i)$ are the final values which $S(i)$, $T(i)$ would converge to if the iterations were continued indefinitely, then

$$\bar{S}(i) + \bar{T}(i) \geqslant S'(r) + T'(r) \text{ for all } i \tag{2}$$

where r is defined so that

$$S'(r) + T'(r) = \underset{i}{\text{Min}}\,(S'(i) + T'(i)) \leqslant \underset{S'(j) > x'}{\text{Min}}\, S'(j) + \underset{T'(i) > y'}{\text{Min}}\, T'(k). \tag{3}$$

For each i, we need to consider four mutually exclusive cases which depend on the values of $S'(i)$ and $T'(i)$, and which cover all possibilities.

Case 1. $S'(i) \leqslant x'$, $T'(i) \leqslant y'$.

In this case, as $S'(i) < \underset{S'(j) > x'}{\text{Min}}\, S'(j)$ and $T'(i) < \underset{T'(k) > y'}{\text{Min}}\, T'(k)$,

$S'(i)$ and $T'(i)$ would not be altered in later iterations by step 1 of the algorithm.

Therefore $\bar{S}(i) = S'(i)$ and $\bar{T}(i) = T'(i)$.

Hence $\bar{S}(i) + \bar{T}(i) = S'(i) + T'(i) > S'(r) + T'(r)$ by (3).

Case 2. $S'(i) > x'$, $T'(i) > y'$.

In this case $S(i)$ and $T(i)$ may be reduced below $S'(i)$ and $T'(i)$ in later iterations. But in later iterations $S(i)$

and $T(i)$ will never be reduced below $\underset{S'(j) > x'}{\text{Min}}\ S'(j)$ and $\underset{T'(j) > y'}{\text{Min}}\ T'(i)$ by step 1 of the algorithm.

Therefore $\bar{S}(i) + \bar{T}(i) > \underset{S'(j) > x'}{\text{Min}}\ S'(j) + \underset{T'(k) > y'}{\text{Min}}\ T'(k)$

$$\geqslant S'(r) + T'(r)$$

by (3).

Case 3. $S'(i) \leqslant x'$, $T'(i) > y'$.

In this case $S(i)$ cannot be reduced below the value of $S'(i)$ in later iterations, but $T(i)$ may be reduced below the value of $T'(i)$. The proof that in this case $\bar{S}(i) + \bar{T}(i) > S'(r) + T'(r)$ requires more detailed examination, so it is treated separately in the next paragraph.

Case 4. $S'(i) > x'$, $T'(i) \leqslant y'$.

The proof in this case that $\bar{S}(i) + \bar{T}(i) > S'(r) + T'(r)$ is identical to the proof for case 3 with the letters S and T interchanged.

Therefore in all cases $\bar{S}(i) + \bar{T}(i) \geqslant S'(r) + T'(r)$ and the inequality which was stated in step 2 of the method as the condition for stopping iterations is correct.

Proof of case 3

It remains to prove that if $S'(i) \leqslant x'$ and $T'(i) > y'$ that $\bar{S}(i) + \bar{T}(i) \geqslant S'(r) + T'(r)$.

The proof requires the examination of the optimum or shortest route from i to t and the use of the properties along this route. Let the final route from i to t be:

$b_1, b_2, \ldots, b_v \ldots, b_w \ldots, b_l$, where $b_1 = i$, $b_l = t$.

The properties of $S'(i)$, $\bar{S}(i)$, $T'(i)$, $\bar{T}(i)$ along this route are as follows:

For all j, $(1 \leqslant j \leqslant l)$

$$\bar{S}(b_{j+1}) \leqslant \bar{S}(b_j) + d_{b_j, b_{j+1}} \tag{4}$$

$$T(b_j) = T(b_{j+1}) + d_{b_j, b_{j+1}} \tag{5}$$

as $b_1, b_2, \ldots b_l$ is the final route from i to t.
Also, as $T'(i) > y'$ this implies $\bar{T}(i) > y'$
and as $T'(t) \leqslant y'$ this implies $\bar{T}(t) < y'$.
Further using (5), $\bar{T}(b_j) > \bar{T}(b_{j+1})$.

Therefore there is a point b_w, $(1 < w \leqslant l)$ such that

$$\bar{T}(b_{w-1}) > y'$$

and

$$\bar{T}(b_w) \leqslant y'$$

i.e.

$$\bar{T}(b_j) > y', \quad (1 \leqslant j \leqslant w - 1) \tag{6}$$

and

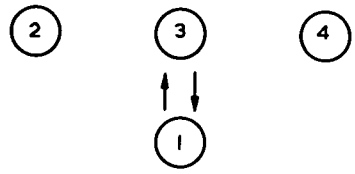
$$\bar{T}(b_j) \leqslant y', \quad (w \leqslant j \leqslant l). \tag{7}$$

The results (6) and (7) imply corresponding results from $T'(b_j)$, namely:

$$T'(b_j) > y', \quad (1 \leqslant j \leqslant w - 1) \tag{8}$$

$$T'(b_j) \leqslant y', \quad (w \leqslant j \leqslant l). \tag{9}$$

Diagrammatically the situation is as illustrated in **Fig 6.**



TRAVELLING SALESMAN PROBLEM WITH FOUR POINTS AND POINT (1) AS THE BASE OF THE TOUR.

Fig. 6

The proof now requires the consideration of three mutually exclusive situations which depend on the values of $S(b_i)$.

(A) If there is a point b_v, $(w \leqslant v \leqslant l)$ such that

$$S'(b_v) \leqslant x',$$

then $\bar{S}(i) + \bar{T}(i) \geqslant \bar{S}(b_v) - \sum_{k=1}^{v-1} d_{b_k, b_{k+1}} + \bar{T}(b_v)$

$$+ \sum_{k=1}^{v-1} d_{b_k, b_{k+1}} \text{ (by (4) and (5))}$$

$$\geqslant \bar{S}(b_v) + \bar{T}(b_v)$$

$$\geqslant S'(b_v) + T'(b_v), \text{ (using (9) and proof of case 1)}$$

$$\geqslant S'(r) + T'(r), \text{ (by (3)).}$$

(B) If there is a point b_v, $(1 < v < w)$ such that

$$S'(b_v) > x',$$

Then $\bar{S}(i) + \bar{T}(i) \geqslant \bar{S}(b_v) + \bar{T}(b_v)$

$$\geqslant \underset{S'(i) > x'}{\text{Min}}\ S'(i) + \underset{T'(i) > y'}{\text{Min}}\ T'(i)$$

(using (8) and proof of case 2)

$$\geqslant S'(r) + T'(r) \text{ (by (3)).}$$

(C) $S'(b_i) \leqslant x'$, $1 \leqslant i \leqslant w - 1$

$S'(b_i) > x'$, $w \leqslant i \leqslant l$.

Then $\bar{S}(i) + \bar{T}(i) \geqslant \bar{S}(b_{w-1}) + \bar{T}(b_{w-1})$

$$\geqslant S'(b_{w-1}) + \bar{T}(b_{w-1})$$

$$\geqslant S'(b_{w-1}) + \bar{T}(b_w) + d_{b_{w-1}, b_w}$$

$$\geqslant S'(b_{w-1}) + T'(b_w) + d_{b_{w-1}, b_w}$$

$$\geqslant S'(b_{w-1}) + T'(b_{w-1})$$

(using the proof of case 1)

(as, although $T'(b_{w-1}) > y'$, it will have been evaluated

and will not be altered since b_{w-1}, b_w is on the final shortest route)

$$\geqslant S'(r) + T'(r) \quad \text{(by (3))}.$$

This concludes the proof.

A worked example

The example of Fig. 1 will be used to demonstrate the procedure. First the d_{ij} matrix is constructed, in which M, the suitably large number for non-connected points, is left as a blank element to improve clarity. Below the matrix the initial values of S, T, P, Q are shown and their changing values over the iterations. (See **Table 1**.)

Table 1

			to							
			1	2	3	4	5	6	7	8	9
d_{ij} from		1	0	3	6	7					
		2	3	0	1		4				
		3	6	1	0			2			
		4	7			0		3	4		
		5		4			0			1	
		6			2	3		0		1	2
		7				4			0		5
		8					1	1		0	2
		9						2	5	2	0
Initial	$S(i)$		0	3	6	7					
	$P(i)$		1	1	1	1	1	1	1	1	1
	$T(i)$							2	5	2	0
	$Q(i)$		9	9	9	9	9	9	9	9	9
1st Iteration	$T(i)$				4	5		2	5	2	0
	$Q(i)$		9	9	6	6	9	9	9	9	9
	$T(i)$				4	5	3	2	5	2	0
	$Q(i)$		9	9	6	6	8	9	9	9	9
2nd Iteration	$S(i)$		0	3	4	7	7				
	$P(i)$		1	1	2	1	2	1	1	1	1
3rd Iteration	$T(i)$			7	4	5	3	2	5	2	0
	$Q(i)$		9	5	6	6	8	9	9	9	9

The iterations run as follows:

Iteration 1. As $\underset{S(j)>0}{\text{Min}}\ S(j) = 3 > \underset{T(k)>0}{\text{Min}}\ T(k) = 2$

$T(i)$ and $Q(i)$ are first reset through row 6 of d_{ij}. Then $T(i)$ and $Q(i)$ are reset through row 8 of d_{ij}. y is reset to 2.

$$\text{Then } \min_i (S(i) + T(i)) = 10 > \min_{S(j)>0} S(j) + \min_{T(k)>2} T(k) = 3 + 3.$$

Iteration 2. As $\underset{S(j)>0}{\text{Min}}\ S(j) = 3 \leqslant \underset{T(k)>2}{\text{Min}}\ T(k) = 3$

$S(i)$ and $P(i)$ are reset through row 2 of d_{ij}. x is reset to 3.

$$\text{Then } \min_i (S(i) + T(i)) = 8 > \min_{S(j)>3} S(j) + \min_{T(k)>2} T(k) = 4 + 3.$$

Iteration 3. As $\underset{S(j)>3}{\text{Min}}\ S(j) = 4 > \underset{T(k)>2}{\text{Min}}\ T(k) = 3$

$T(i)$ and $Q(i)$ are reset through row 5 of d_{ij}. y is reset to 3.

$$\text{Then } \min_i (S(i) + T(i)) = 8 \leqslant \min_{S(j)>3} S(j) + \min_{T(k)>3} T(k) = 4 + 4.$$

The shortest distance from point 1 to point 9 is therefore 8 units and the shortest route is built up as

$$P(3),3,Q(3)$$
$$P(2),2,3,6,Q(6)$$
$$1,2,3,6,9.$$

Alternative methods of finding the shortest route

Both dynamic and linear programming have been applied to the shortest route problem. Dynamic programming might be better than the method described in the paper if the intention was to find the shortest routes between a starting point and a number of finishing points, but on the straightforward shortest route problem dynamic programming evaluates more information than is necessary. In the case of linear programming, the trans-shipment technique handles the shortest route problem, but this serves more to emphasize the scope of linear programming than to provide a really suitable solution.

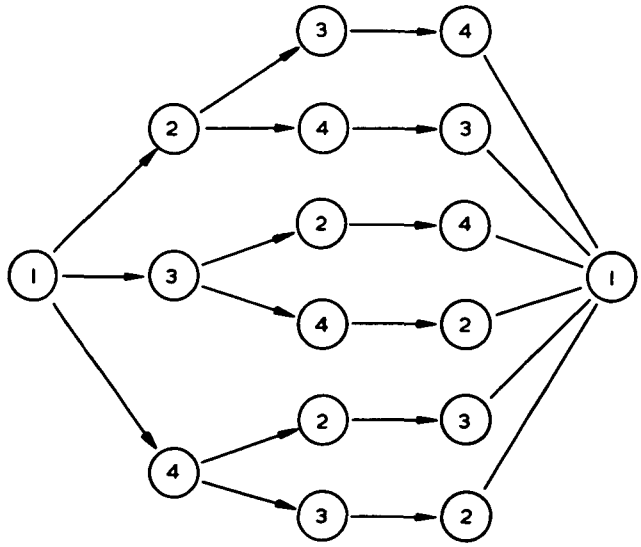
Three computer programs were written for the IBM 7030 computer to compare the execution times of the dynamic and linear programming approaches against the method described above. The results confirmed the advantages of the method described. On a network of 100 points with 10 randomly chosen starting and terminating points the average execution times were as follows:

Method described	0·47 seconds
Dynamic programming	3·29 seconds
Linear programming	82·1 seconds

It is difficult to ensure that this sort of comparison by computer program is completely objective, because one knows how to exploit for programming the particular features of one's own method. But it is certain that unless the numbering of the network is specially arranged to suit the particular case, the new method requires many fewer comparisons than the dynamic programming

approach. The solution of the problem by linear programming is inevitably weak because it cannot exploit the structure of the problem. One further computational point should be mentioned. If the network is sparse then it will be wasteful to use the whole of the square matrix to record the distances. In this case the distances should be stored in a list form, and a few simple alterations to the notation adapt the procedure to this situation.

An alternative approach might be to find a first feasible route between s and t and then try to improve it. Some measure would have to be found associated with each point on the route ascertaining how far it would be worthwhile to deviate from the route. Moreover finding the first feasible route would not be trivial. It might be tackled by "plotting" the network, fully unravelled, and attempting to use a sense of direction in the network branches. But plotting would work only if the network distances satisfied the triangle inequality in which "distance (1,2) $\leqslant$ distance (1,3) + distance (3,2)", and this inequality would not hold in some applications. A first feasible approach therefore looks burdened with a lot of calculation compared with the "least distance" method described above.



ASSOCIATED NETWORK OF POSSIBLE ROUTES.

Fig. 7

Applications of the shortest route problem

The most obvious application of the technique is to traffic routing, either for setting up a system or for dynamically controlling traffic flow, possibly taking account of traffic densities. In the field of computing science, program flow is beginning to be analyzed on a network basis, so that the technique may have applications in systems design. And just as a linear programming trans-shipment technique tackles the shortest route problem, so, conversely, some trans-shipment problems may be treated by a shortest route procedure.

Sometimes it is possible to translate a problem not apparently related to networks into a shortest route problem. Hardgrave and Nemhauser (1963) reduced the Akers–Friedman scheduling problem into the problem of finding the shortest route through a network. Also, Held and Karp (1965) converted the travelling salesman problem and the assembly-line balancing problem into shortest route problems, but the necessary constructions seem to be rather awkward (see **Fig. 7**).

Other applications may occur in any network situation, whether it is an electrical network or one of the network structures now being used for management planning and control. The technique described in this paper is an addition to the increasing number of network procedures now collecting under the name of network algebra.

Acknowledgements

I am very grateful to Mr. A. R. Curtis and Mr. R. G. Garside of A.E.R.E., Harwell, for the valuable suggestions and advice they have given me in designing the technique. I would also like to thank the referee for pointing out a number of obscurities in my original presentation of the paper. Finally I am indebted to Miss C. I. Powell of A.E.R.E. for her assistance with the preparation of the computer programs.

References

HARDGRAVE, W. M., and NEMHAUSER, G. L. (1963). "A geometric model and a graphical algorithm for a sequencing problem", *JORSA*, Vol. 11, p. 889.

HELD, M., and KARP, R. M. (1965). "The construction of discrete dynamic programming algorithms", *IBM Systems Journal*, Vol. 4, No. 2, p. 136.

VAJDA, S. (1961). *Mathematical Programming*, Chap. 6, New York: Addison-Wesley.