RAIRO. RECHERCHE OPÉRATIONNELLE

BRIAN BOFFEY

The all-to-all alternative route problem

RAIRO. Recherche opérationnelle, tome 27, n° 4 (1993), p. 375-387

<http://www.numdam.org/item?id=RO_1993__27_4_375_0>

Recherche opérationnelle/Operations Research
(vol. 27, n° 4, 1993, p. 375 à 387)

THE ALL-TO-ALL ALTERNATIVE ROUTE PROBLEM (*)

by Brian BOFFEY ([1])

Abstract. – *There are various K-best route problems associated with a network. The one studied here is that of finding, for all node pairs (s, t), a best route from s to t together with an alternative route which is optimal subject to not containing the first edge of the best route. The relevance of this problem to dynamic vehicle guidance and to routing in communication networks is briefly discussed. An algorithm is developed whose complexity, under conditions likely to be met in practice, is established. An illustrative example is given.*

Keywords: Network; *K*-best routes problem; alternative routes.

Résumé. – *Il y a plusieurs problèmes d'itinéraire « K-best » associés à un réseau. Celui qui est étudié ici est celui qui consiste à trouver pour toutes les paires de nodes (s, t) un meilleur trajet pour aller de s à t ainsi qu'un trajet alternatif qui n'utilise pas le premier arc de la meilleure route. La pertinence de ce problème à* « dynamic vehicle guidance » *et en réseaux de communications est discuté brèvement. Un algorithme est développé, dont la complexité est établi sous des conditions que l'on rencontre probablement en pratique. Un exemple est joint.*

Mots clés : Réseau; *K*-meilleur routes problème; routes alternatives.

1. INTRODUCTION

The usual aim in routing problems is to find a best (shortest, least cost, quickest, most reliable, ...) route from an *origin s* to a *destination t*; or, more generally, best routes from a set of origins to a set of destinations. Sometimes, there may be features which are of importance, but are not easily or conveniently included in the model. In this case, a useful strategy is to generate the best K routes according to the model, with the final choice being made in the light of non-modelled features.

(*) Received March 1992.
([1]) Statistics and Comp. Mathematics Dept., University of Liverpool, Liverpool, L69 3BX, Great Britain.

Recherche opérationnelle/Operations Research, 0399-0559/93/04/$ 4.00

There is, however, not a single type of K-best route problem. Here we shall distinguish three types (though no doubt there are more), and these will be described for a single origin s and a single destination t.

1. *Unrestricted K-best route problem*

The K best routes from s to t are computed subject to no restrictions except, possibly, that the routes contain no repeated nodes (e.g. Perko [12]). The final choice of route is made by the 'decision-maker' in the light of non-modelled features (e.g. [7]).

2. *Disjoint K-best route problem*

It is required to find a set of K routes from s to t which are edge-disjoint and such that the *worst* of these is as good as possible. A slightly more restrictive version is to require node-disjointness. Such problems arise in connection with communication networks for which reliability is paramount (e.g. communication networks in banking). For example it may be required that a network be designed with specified pairs of nodes connected by two (more generally K) node-disjoint routes (Monma and Shallcross, [8]).

3. *Restricted K-best route problem*

It is required to find a set of K routes from s to t such that the first edge of the k-th best is not contained in the q-th best for $1 \leq k < q \leq K$ [14]. Such problems arise in connection with networks subject to congestion [2, 3] and when edges may fail.

To reduce congestion in computer networks, Rudin [13] suggested that packets of information from node s to node t should be sent from s along the first edges of the first, ..., K-th best restricted routes from s to t with probabilities $p_i(s, t)$ satisfying

$$p_1(s, t) \geq p_2(s, t) \geq \ldots \geq p_K(s, t).$$

Such a strategy might be useful in connection with the emerging *dynamic guidance systems* for road traffic [1]. A possibility for a *directive* system would seem to be to take $k=2$ and to dispatch traffic from s along the first links of the first and second best routes. (If the second best route is 'too much worse' than the first then $p_2(s, t)$ should be set to zero otherwise confidence in the system is likely to be lost.)

Topkis [14] discussed the restricted K-best route problem in relation to link failure in computer networks employing virtual circuit routing. There might also be scope for its use in a dynamic guidance system for diverting traffic

in the immediate aftermath of a road being 'blocked' due to an accident (*cf.* section 4).

Most networks are sufficiently reliable for the probability of more than one edge being faulty simultaneously to be negligible. That is, the case $K=2$ is particularly relevant and the associated problem will be termed the *alternative route problem*. It is the all-to-all version of this problem that will form the principal subject of this paper.

In section 2, the Nemhauser (A^*) algorithm is briefly described and its application to the single-origin single-destination alternative route problem given. Section 3 then develops an algorithm for the all-to-all version of the alternative route problem, and an illustrative numerical example is given. Finally, some conclusions are given in section 4 where it is argued that the proposed algorithm has computational advantages under conditions likely to be met in practice.

2. THE SINGLE-ORIGIN SINGLE-DESTINATION PROBLEM

For convenience, undirected networks will be replaced by the corresponding symmetric directed networks with each edge $i-j$ of length a being replaced by a pair of oppositely directed arcs $i-j$ and $j-i$ each of length a. Routes in these directed networks will be termed 'paths'. In the subsequent development, the following notation will be used:

$N=(V, A, a)$	a directed network with node set V, arc set A and arc length function a
a_{ij}	the length of arc i-j from node i to node j
$d(i, j)$	the (shortest) distance from node i to node j
$d(j)$	an abbreviation for $d(s, j)$ where s is the origin
$dist(j)$	label at node j [an upper estimate of $d(j)$]
$CAND$	set of candidate nodes for scanning.

To find the (shortest) distance from node s to node t in network N the following general algorithm may be adopted. For simplicity, it is assumed that there does exist a path from s to t, and the details of pointer manipulation relating to shortest *path* calculation are omitted.

Label Setting Algorithm

STEP 1 (Setup) Set $dist(s)=0$; $dist(j)=\infty$ all $j\neq s$; $CAND=\{s\}$.

STEP 2 (Selection for scanning)

Select $u\in CAND$; $CAND \leftarrow CAND-\{u\}$.

If $u=t$

then terminate with $d(s, t) = dist(t)$.
STEP 3 (Scanning and label update)
For each arc u-ν **do**
begin if $dist(\nu) > dist(u) + a_{u\nu}$
then $dist(\nu) \leftarrow dist(u) + a_{u\nu}$; $CAND \leftarrow CAND \cup \{\nu\}$.
end
Return to step 2.

It will be assumed that the **select** operator of step 2 is such that a node u satisfying

$$dist(u) + h(u) = \underset{x \in CAND}{\text{MIN}} \{dist(x) + h(x)\}$$

is chosen for some specified function h and that preference is given to node t when it satisfies the minimisation criterion. This yields a valid algorithm for finding the distance from s to t if h is a *consistent* function [9]; that is, it satisfies the generalised triangle inequality

$$h(x) \leq a_{xy} + h(y) \quad \text{for all arcs } x - y.$$

Moreover, when u is selected for scanning $dist(u) = d(s, u)$ [9].

Note that if h is a consistent function then so is h^* defined by

$$h^*(x) = h(x) - constant \quad \text{for all nodes } x.$$

Consequently, without loss of generality, it will be assumed that $h(t) = 0$ implying that $h(x)$ is a *lower bound* to $d(x, t)$ for all nodes x [9].

The label-setting algorithm with node selection as described will be termed Nemhauser's algorithm. (This algorithm, under the name 'A*', appeared in the artificial intelligence field in 1968 [6].) In the particular case of $h \equiv 0$, Nemhauser's algorithm effectively reduces to the well-known Dijkstra algorithm; however, taking $h \equiv 0$ is potentially a poor choice for h as may be seen from the next two results.

THEOREM 1: *If* $\{b_{ij}\}_{i-j \in A}$ *is any set such that* $b_{ij} \leq a_{ij}$ *for all arcs* $i-j$, *then* h^b *is a consistent function where*

$$h^b(x) = d^b(x, t) \quad \textit{for all nodes } x \in V.$$

$d^b(x, t)$ *denotes the distance from node* x *to node* t *when the set* $\{b_{ij}\}_{i-j \in A}$ *is used to provide arc lengths.*

Proof: The result is true since, for any arc $x-y$,

$$h^b(x) = d^b(x, t) \leq d^b(x, y) + d^b(y, t) \leq a_{xy} + d^b(y, t) = a_{xy} + h^b(y).$$

THEOREM 2: *If* $\{b_{ij}\}_{i-j\in A}$ *and* $\{c_{ij}\}_{i-j\in A}$ *are two sets of numbers satisfying* $c_{ij}\leq b_{ij}\leq a_{ij}$ *for all arcs* $i-j$, *then Nemhauser's algorithm with consistent function* h^b *results in no more nodes being scanned than if consistent function* h^c *were used.*

Proof: When t is selected for scanning the set of nodes that have been scanned is ([6], [9], [10])

$$S(h) = \{x|dist(x) + h(x) \leq dist(t) + h(t)\}$$
$$= \{x|dist(x) + h(x) \leq dist(t)\}.$$

The desired result follows by noting that $S(h^b)\subseteq S(h^c)$.

It is thus seen that, providing the effort expended on calculating $h(x)$ is not too great, it is advantageous to choose h to be as 'strong' a consistent function as possible [that is $h(x)$ should be as near as possible to $d^a(x, t)=d(x, t)$].

A consistent function is easily obtained for networks carrying non-linear flow with the cost $a_{ij}=f_{ij}(\nu_{ij})$ of flow in arc $i-j$ being an increasing function f_{ij} of the flow intensity ν_{ij} in $i-j$. Setting $b_{ij}=f_{ij}(0)\leq a_{ij}$ and using h^b provides a consistent function (*cf.* theorem 1) which we will call the *zero intensity function*. Specific examples are: $f_{ij}(\nu_{ij})=b_{ij}/(1-\nu_{ij})$ when the cost (delay) is modelled as an (M/M/1) queuing system; $f_{ij}(\nu_{ij})=b_{ij}+k_{ij}\nu_{ij}^4$ for the cost (delay) in a road network, k_{ij} being a constant.

We now turn to the problem of finding the shortest and alternative routes from node s to node t. First, the shortest path, π say, may be found by a shortest path algorithm (e.g. Dijkstra's algorithm or, preferably, a label-correcting algorithm [5]) applied to find the shortest route *from* every node to node t. Then the alternative path is found by 'removing' the first arc of π and applying Nemhauser's algorithm with the zero-intensity function providing a consistent function.

3. THE ALL-TO-ALL ALTERNATIVE ROUTE PROBLEM

For the general restricted K-best route problem Topkis [14] introduced an algorithm based on Dijkstra's algorithm. Its complexity is $O(KD)$ where $O(D)$ is the complexity of Dijkstra's algorithm. The Topkis algorithm could indeed be used to solve the all-to-all alternative route problem ($K=2$).

However it requires a relatively large amount of initialisation with $K+1$ labels having to be set at every node x. Consequently, it seems that it might be worthwhile seeking an alternative algorithm. A description of one such attempt follows.

The approach consists of two phases:

Phase (1) : first, for each node x, determine shortest paths *from* every other node.

Phase (2) : for each node x, find the restricted second shortest path *to* every other node.

Phase (1) may be effected by using a Dijkstra-like algorithm or, preferably, a label-correcting algorithm based on deques (Gallo and Pallottino, [5]) applied for each node x. Alternatively, an integrated scheme may be used [4]. However, whatever the overall scheme used, the effort of finding all first shortest paths must be embedded somewhere in the calculations. Applying the simple scheme of section 2 would require Nemhauser's algorithm to be applied for every pair of nodes. This would be time consuming and it seems there should be scope for utilising the information gained in phase (1) to reduce the effort of finding alternative paths. A description of how this might be realised will now be given.

First, the set of nodes $V-\{s\}$ is partitioned into sets $F(j)$, with one for each neightbour j of s, such that x is in $F(j)$ only if arc $s-j$ is the first arc on a shortest path from s to x. This is achieved by setting $F(j)$ to be the set of descendants of j relative to some shortest path tree rooted at s. Figure 1b shows an example of such a partitioning for $s=16$

$F(12)=\{1, 2, 5, 12\}$

$F(13)=\{4, 7, 8, 9, 13\}$

$F(17)=\{3, 6, 10, 11, 14, 15, 17, 18\}$.

Note that there are two distinct shortest path trees and node 3 could equally have been included in $F(12)$.

For nodes in the set $F(j)$ the alternative path may be found by applying a shortest path algorithm from origin s with link $s-j$ removed (or, equivalently, its length set to ∞). Note that if link $s-j$ does not provide the shortest path from s to j [that is, $j \notin F(j)$] then the shortest path calculations from s with $s-j$ removed are redundant and may be omitted. It is, therefore, assumed from now on that $s-j$ is the first shortest path from s to j. Then j becomes, in a sense, the most 'inconvenienced' node when link $s-j$ is removed. This is formalised in theorem 3 [3].

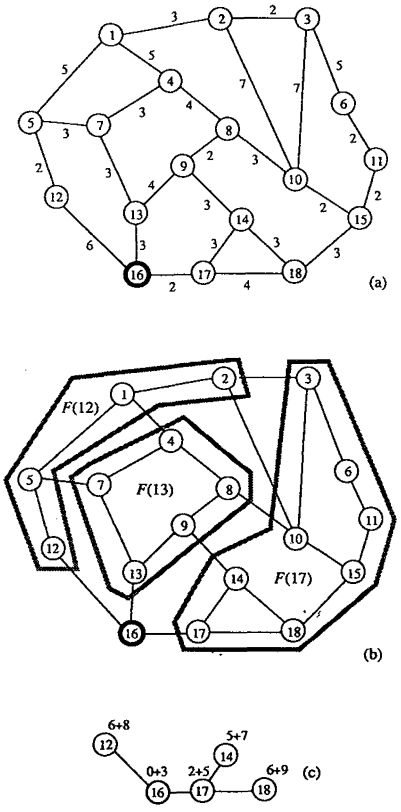

Figure 1. – With regard to the network of (a) **and origin node 16,** (b) **shows** $F(12)$, $F(13)$ **and** $F(17)$. **In** (c) **is given the tree obtained using Nemhauser's algorithm to find the restricted second shortest path from node 16 to node 13; the numbers beside nodes are of the form** $dist(x)+h(x)$.

THEOREM 3: *$d_2(s, j)-d_1(s, j)\geq d_2(s, x)-d_1(s, x)$, for all $x\in F(j)$, where $d_m(a, b)$ denotes the length of the m-th best restricted path from node a to node b. [Thus, $d_1(a, b)=d(a, b)$ for all node pairs (a, b).]*

Proof: By the definition of $F(j)$,

$$d_1(s, x) = d_1(s, j) + d_1(j, x). \tag{1}$$

The second best restricted shortest path from s to j followed by the shortest path from j to x provides a path from s to x which is not the designated first shortest path. Hence

$$d_2(s, x) \leq d_2(s, j) + d_1(j, x). \tag{2}$$

The desired result follows immediately from (1) and (2).

Note that the stronger result $d_m(s, j) - d_1(s, j) \geq d_m(s, x) - d_1(s, x)$, for all $x \in F(j)$, is in fact true. The proof is almost identical to that of theorem 3.

Consider now the application of Nemhauser's algorithm from origin s with a consistent function h defined by $h(x) = d^a(x, j)$ for all x (that is, $b_{ij} = a_{ij}$ all $i - j \in A$ in the notation of theorem 1). Suppose the algorithm is terminated as soon as a node u satisfying

$$h(u) < d_1(u, s) + d_1(s, j) \tag{3}$$

is selected for scanning.

THEOREM 4: *Let CAND be the set of temporarily labelled nodes when application of Nemhauser's algorithm is terminated by condition (3). Then, for all* $x \in F(j)$

$$d_2(s, x) = \min_{\nu \in CAND} \{dist(\nu) + d_1(\nu, x)\}. \tag{4}$$

Proof: Let w be any value of ν for which the minimum of (4) occurs with the restriction that if there is a tie and u minimises (4) then $w = u$. Now suppose there is a shortest path π from w which goes to x via s. Firstly, w cannot be equal to u for then

$$\begin{aligned}
d_1(u, x) &= d_1(u, s) + d_1(s, x) && s \text{ is on } \pi \text{ and } w = u \\
&= d_1(u, s) + d_1(s, j) + d_1(j, x) && x \in F(j) \\
&> d_1(u, j) + d_1(j, x) && \text{by (3)} \\
&\geq d_1(u, x) && \text{by the triangle property}
\end{aligned}$$

which is impossible. Hence, $w \neq u$ and

$$\begin{aligned}
& dist(u) + d_1(u, x) \\
&> dist(w) + d_1(w, x) \\
&= dist(w) + d_1(w, s) + d_1(s, x) && \text{by assumption} \\
&= dist(w) + d_1(w, s) + d_1(s, j) + d_1(j, x) && x \in F(j) \\
&\geq dist(w) + d_1(w, j) + d_1(j, x) && \text{by the triangle property} \\
&\geq dist(u) + d_1(u, j) + d_1(j, x) && \text{by definition of } u \\
&\geq dist(u) + d_1(u, x). && \text{by the triangle property}
\end{aligned}$$

But this is a contradiction and therefore no shortest path from w goes via s.

Let π_1 be a shortest path from s to w *not* including $s-j$, and π_2 be a shortest path from w to x. Then, $\pi' = \pi_1 \pi_2$ is a path from s to x which does not include $s-j$. Suppose π' is not a shortest such path there being a shorter one π^*. If m is the first node on π^* which does not belong to *CAND*, then

$$\begin{aligned} dist(m)+h(m) &= \text{length of } \pi^* && \text{predecessor of } m \text{ has been scanned} \\ &= d_2(s, x) && \text{definition of } \pi^* \\ &\leq \text{length of } \pi' && \text{definition of } \pi^* \\ &\leq dist(w)+h(w) && \text{length of } \pi_1 \leq dist(w) \\ &\leq dist(m)+h(m) && \text{definition of } w. \end{aligned}$$

Therefore, the inequalities must be equalities; in particular,

$$d_2(s, x) = dist(w) + h(w).$$

Moreover, π' provides the alternative path from s to x.

When Nemhauser's algorithm is terminated in this way, it is unlikely that j will have been labelled and the number of nodes scanned will often be *very* small. In section 4, it is shown that, under reasonable conditions, the computational effort required in phase (2) is essentially quadratic in the number of nodes in the network; that is, the calculation is *asymptotically* dominated by phase (1).

Example 1: For the network of figure 1 *a*, find the alternative paths from node 16 to all nodes in $F(13)$.

Solution: After nodes 12 and 17 have been labelled node 17 is selected for scanning. Since the termination criterion (3) is not met, node 17 is scanned and nodes 14 and 18 become labelled. At this point node 14 is selected for scanning but

$$h(14) = 7 < 8 = d_1(14, 16) + d_1(16, 13).$$

It can be concluded that the alternative path from node 16 to node 13 is via the first shortest path 16-17-14 from 16 to 14 followed by the first shortest path 14-9-13 from 14 to 13.

When Nemhauser's algorithm terminates $CAND = \{12, 14, 18\}$ and, by theorem 4

$d_2(16, 4) = \min(6+8, 5+9, 6+12) = 14$, alternative path is 16-12-5-7-4 or 16-17-14-9-8-4

$d_2(16, 7) = \min(6+5, 5+10, 6+13) = 11$, alternative path is 16-12-5-7

$d_2(16, 8) = \min(6+12, 5+5, 6+8) = 10$, alternative path is 16-17-14-9-8

$d_2(16, 9) = \min(6+12, 5+3, 6+6) = 8$, alternative path is 16-17-14-9

$d_2(16, 13) = \min(6+8, 5+7, 6+9) = 12$, alternative path is 16-17-14-9-13.

4. DISCUSSION

What of the complexity of the method when the zero-intensity function is utilised? It would be inappropriate to treat the arc lengths as being unrestricted since:

(1) routing tends to even out arc loads;

(2) if an arc does become heavily overloaded then access to it is likely to be restricted in some way: in a computer network this is achieved by congestion and flow control [2]; in a dynamic guidance system a seriously overloaded arc could be regarded as being failed when the next shortest path computation [phase (1)] is carried out.

Consequently, the length of any arc might be expected not to exceed a 'small' multiple of its minimal length. Accordingly, the set of arc lengths will be said to be of *M-bounded variation* if for every arc $i-j$, $H \leq a_{ij}(t) \leq MH$, where $a_{ij}(t)$ denotes the value of a_{ij} at time t and H is a constant for the network in question. In essence M-bounded variation means that there are no very short arcs and that no arc becomes very congested. Since the problem is linear, cost units may be chosen so that $H=1$. For simplicity therefore M-bounded variation will be assumed to mean $1 \leq a_{ij}(t) \leq M$.

THEOREM 5: *Let N be a network satisfying*

(1) *no node has more than* Δ *outgoing arcs* (i.e. *the outdegree of no node exceeds* Δ)

(2) *the set of arc lengths is of* M-bounded variation

(3) *for each arc* $i-j$ *there is a path with no more than e arcs from i to j and which does* not *include* $i-j$

(4) Δ, *e and M are independent of n and constant.*

Then phase 2 of the all-to-all alternative route algorithm can be executed in $O(n)$ *time.*

Proof: The length of the second shortest path is at most eM in length. Consequently, this may be found by scanning no more than Δ^{eM} nodes even if Dijkstra's algorithm is used. Since Dijkstra's algorithm will require at least as many nodes to be scanned as Nemhauser's (theorem 2) it follows that the

second shortest path from node i to node j may be found in $O(1)$ time (if Perko's device [11] is used for label initialisation). A node has at most Δ neighbours and so it follows that all applications of Nemhauser's algorithm can be carried out in $O(1)$ time. Finally, the application of theorem 4 requires the comparison of $O(1)$ terms for $O(n)$ nodes and so requires $O(n)$ time. This completes the proof.

The essence of the above result is that provided the stated conditions are satisfied, the application of Nemhauser's algorithm involves only a *local* search whatever the size of the network. Thus, if calculations are distributed it is only necessary to have knowledge of *local* arc lengths and, for test (4), distances from local nodes only are needed. It follows that at any node only $O(nQ)$ items need be stored where Q is independent of n. The magnitude of Q is indicated by the proof of theorem 5, but in order to get a better feel for the amount of computation involved the special case of a square grid network will be considered for $M=1, 2, 3$. The worst case occurs when all the arcs on the second shortest path are at maximum length and all others are at minimal length. Results are given in the table below in which ν is the number of permanently labelled (*i.e.* scanned) nodes using (3) as termination criterion and B is the number of temporarily labelled (*i.e.* labelled but unscanned) nodes.

TABLE 1
(Square grid: $\Delta=4$, $e=3$).

M	1	2	3
ν	4	7	12
B	7	9	11

Corresponding results for an hexagonal grid are shown in table 2.

TABLE 2
(Square grid: $\Delta=3$, $e=5$).

M	1	2	3
ν	7	10	10
B	7	9	9

It thus appears that the amount of computational effort required to find all alternative paths by the proposed algorithm is very modest.

The use of the zero-intensity function in connection with non-linear network flow can result in a 'weak' consistent function if the network is fairly congested; perhaps we can do better. If the optimal flow pattern *for the system* is being considered, we may expect that the removal of an arc will lead to some arc flows being increased but none being decreased. This is likely to be the case for computer network networks and for dynamic guidance systems with a strong directive element. If this supposition is justified then b_{ij} may be taken as the value of a_{ij} *before* arc failure. Moreover, the values of $h(x)=d^a(x, t)$ are already available from phase (1).

In any case, since the system is dynamic the a_{ij} will be changing so that information is always a little out of date. Also, it is likely that few alternative paths will come into operation between successive shortest path calculations. Consequently it is better to perform the alternative path calculation only as necessary. This is quite feasible as only one application of Nemhauser's algorithm is necessary together with some minimisations (*cf.* example 1).

Finally, is the restriction to $K=2$ essential? The answer is no, since theorem 4 generalises readily. This time Nemhauser's algorithm is applied $m-1$ times with at application q, the first arcs of the first q restricted paths barred.

THEOREM 6: *Let CAND be the set of temporarily labelled nodes when application of Nemhauser's algorithm is terminated by condition (3). Then, when the q-th application of Nemhauser's algorithm terminates*

$$d_{q+1}(s, x) = \min_{\nu \in CAND} \{dist(\nu) + d_q(\nu, x)\} \quad \text{for all } x \in F(j).$$

REFERENCES

1. M. BEN-AKIVA, A. D. PALMA and I. KAYSI, Dynamic Network Models and Driver Information Systems, *Transpn. Res. A.*, 1991, *25A*, pp. 251-266.
2. D. P. BERTSEKAS and R. GALLAGER, Data Networks, 1987, Prentice-Hall, Englewood Cliffs, NJ.
3. T. B. BOFFEY, Distributed Computing: Associated Combinatorial Problems, 1992, Blackwell Scientific Publications, Oxford.
4. M. FLORIAN, S. NGUYEN and S. PALLOTTINO, A Dual Simplex Algorithm for Finding all Shortest Paths, *Networks*, 1981, *11*, pp. 367-378.
5. G. GALLO and S. PALLOTTINO, Shortest Path Methods: a Unifying Approach, *Math. Prog. Study*, 1986, *26*, pp. 38-64.
6. P. HART, N. NILSSON and B. RAPHAEL, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Trans. Syst. Man Cybernet.*, 1968, *4*, pp. 100-107.
7. G. J. HORNE, Finding the *K* Least Cost Paths in an Acyclic Activity Network, *J. Opl Res. Soc.* 1980, *31*, pp. 443-448.

8. C. L. MONMA and D. F. SHALLCROSS, Methods for Designing Communications Networks with Two-Connected Survivability Constraints, *Oper. Res.*, 1989, *37*, pp. 531-541.
9. G. L. NEMHAUSER, A Generalized Permanent Label Setting Algorithm for the Shortest Path Between all Nodes, *J. Math. Analysis & Applic.*, 1972, *38*, pp. 328-334.
10. N. J. NILSSON, Problem Solving Methods in Artificial Intelligence, 1971, McGraw-Hill, New York.
11. A. PERKO, A Representation of Disjoint Sets with Fast Initialization, *Information Processing Lett.*, 1983, *16*, p. 21.
12. A. PERKO, Implementation of Algorithms for *K* Shortest Loopless Paths, *Networks*, 1986, *16*, pp. 149-160.
13. H. RUDIN, On Routing and Delta Routing: a Taxonomy and Performance Comparison of Techniques for Packet-Switched Networks, *IEEE Trans. Commun.*, 1976, COM-24, pp. 43-59.
14. D. M. TOPKIS, A *k* Shortest Path Algorithm for Adaptive Routing in Communication Networks, *IEEE Trans. Commun.*, 1988, COM-36, pp. 855-859.