ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XIII, 2 (1972)

ALGORITHMS 17-18

M. SYSŁO (Wrocław)

THE SHORTEST PATHS FROM A FIXED NODE IN A NETWORK

1. Procedure declaration of Algorithm 17. The procedure *paths* calculates for a given n-nodes network

(i) the shortest path between two nodes s and t if $t > 0$,

(ii) the shortest paths between node s and all other nodes if $t < 0$.

Data:

n	–	the number of nodes of the network,
$d[1:n, 1:n]$	–	the array of non-negative integer distances between all nodes of a network; the elements $d[i,j]$ for which no arc exists between nodes i and j should be not less than $M = n \times \max d[k,l]$, i.e., n times the maximum connected link value,
M	–	integer number not less than $n \times \max d[k,l]$,
s	–	start node number,
t	–	in case (i): end node number ($t \neq s$), or in case (ii): a negative integer number.

Results:

path$[1:n]$	–	in case (i): *path*$[t]$ is the value of shortest path between two nodes s and t, in case (ii): the path array containing the shortest paths from node s to all other nodes (*path*$[s] = 0$),
precede$[1:n]$	–	array of node numbers such that *precede*$[k]$ contains the number of the node preceding node k on the shortest path (*precede*$[s] = 0$).

Remark. The list of nodes on the shortest path can be determined by procedure *shortpath* [1].

```
procedure paths(n,d,M,s,t,path,precede);
 value n,M;
 integer n,M,s,t;
 integer array d,path,precede;
 begin
  integer count,i,k,l,l1,min,s1;
  integer array scan[1:n];
  for i:=1 step 1 until n do
   begin
    path[i]:=M;
    scan[i]:=precede[i]:=0
   end i;
  k:=path[s]:=0;
  scan[s]:=count:=1;
  s1:=s;
  for min:=M while if t < 0 then count < n else k ≠ t do
   begin
    l:=path[s1];
    for i:=1 step 1 until n do
     if scan[i]=0
      then
      begin
       l1:=l+d[s1,i];
       if l1 < path[i]
        then
        begin
         path[i]:=l1;
         precede[i]:=s1
        end l1<path[i]
        else l1:=path[i];
```

```
              if l1 < min
               then
               begin
                min:=l1;
                k:=i
               end l1<min
             end scan[i]=0, i;
           if min=M
            then go to end;
           count:=count+1;
           s1:=k;
           scan[s1]:=1
         end min;
       end:
        end paths
```

2. Method used. The algorithm of Dijkstra (cf. [3]) has been used in the procedure *paths*. This algorithm and its modification, given below, are inductive methods and they base on the following well-known theorem (cf. [2]).

The set S_k consisting of the fixed node and $k-1$ other nodes, and the minimum distances from the fixed node to nodes $i \in S_k$ using arcs of the whole network are assumed known and have the values π_i $(i \in S_k)$.

THEOREM. *If* $d_{ij} \geqslant 0$ *and* π_i *of* S_k *are known, then* $\pi_q = \pi_p + d_{pq}$ *is the minimal distance for* $q \notin S_k$, *where*

$$\pi_p + d_{pq} = \min_{\substack{i \in S_k \\ j \notin S_k}} (\pi_i + d_{ij}), \qquad p \in S_k.$$

3. Modification (Algorithm 18). The Dijkstra algorithm can be modified for large networks with distances which are of the same rank.

The idea of modification is that in the fundamental iterative step all (not one, as in the Dijkstra algorithm) smallest tentative node labels are declared permanently.

The modification of the Dijkstra algorithm may be described most shortly as follows:

I. Notation:

n — the number of nodes of the network,
d_{ij} — the distance between two nodes i and j $(i, j = 1, 2, \ldots, n)$,
s — start node number,
t — (i) end node number, (ii) a negative integer number,
l_i — the label of node i that changes from 0 to 1 when a node is assigned permanently,
p_i — the current minimum distance from s to i,
v_i — node number of the node preceding node i on the shortest path.

II. The algorithm:

Initialization: $l_i \leftarrow 0$, $p_i \leftarrow \infty$, $v_i \leftarrow 0$ $(i = 1, 2, \ldots, n)$, $i \neq s$; $l_s \leftarrow 1$, $p_s \leftarrow 0$, $v_s \leftarrow 0$, $k \leftarrow 0$.

Step 1. For each m for which $p_m = k$ set $p_i \leftarrow p_m + d_{mi}$, $v_i \leftarrow m$, for all i for which $l_i = 0$ and $p_i > p_m + d_{mi}$.

Step 2.

$$k \leftarrow \min_{l_i = 0} p_i.$$

If $k = \infty$, then go to step 4 else set $l_i \leftarrow 1$ for all i for which $p_i = k$.

Step 3. (i) If $l_t = 0$, then go to step 1. (ii) In there exists any $l_i = 0$, then go to step 1.

Step 4. (i) p_t is the minimum distance from s to t, (ii) p_i is the minimum distance from s to i $(i = 1, 2, \ldots, n,\ i \neq s)$.

If $p_i \neq \infty$ and $l_i = 1$, then the shortest path from s to i is

$$s, a_{j-1}, a_{j-2}, \ldots, a_2, i,$$

where $a_1 = i$, $a_i = v_{a_{i-1}}$, $i = 2, 3, \ldots, j$, $a_j = s$.

4. Certification. The procedures *paths* and *modpaths* have been verified on the computer ODRA 1204 for many examples. For examples with random distance matrices the following results have been obtained (time in secs):

	$t < 0$			$t > 0$		
n	procedure *paths*	procedure *modpaths*	procedure *netpath* [1]	procedure *paths*	procedure *modpaths*	procedure *minpath* [1]
10	0.3	0.5	0.6	0.1-0.3	0.1-0.5	0.3-0.6
20	1.2	1.2	2.5	0.1-1.2	0.2-1.2	0.4-1.2
40	4.4	3.0	12.0	0.2-4.4	0.3-3.0	1.0-2.0
60	10.0	6.0	20.0	1.0-10.0	1.0-6.0	2.0-5.0
80	17.0	9.0	30.0	1.0-17.0	1.0-9.0	2.0-7.0
90	20.0	10.0	40.0	3.0-20.0	2.0-10.0	3.0-9.0

```
procedure modpaths(n,d,M,s,t,path,precede);
 value n,M;
 integer n,M,s,t;
 integer array d,path,precede;
 begin
  integer count,fi,fii,i,i1,i2,i3,k1,k11,k2,l,l̄1,l2,mīn,M1;
  integer array f,scan[1:n];
  for i:=1 step 1 until n do
   begin
    path[i]:=M;
    scan[i]:=precede[i]:=0
   end i;
  path[s]:=0;
  scan[s]:=count:=fi:=k1:=k2:=1;
  M1:=M-1;
  k11:=n;
  f[1]:=s;
  for min:=M1 while if t < 0 then count < n else true do
   begin
    for i:=1 step 1 until n do
     if scan[i]=0
      then
      begin
       l1:=path[i];
       for i2:=k1 step k2 until fi do
        begin
         i1:=f[i2];
         l2:=path[i1]+d[i1,i];
         if l2 < l1
          then
          begin
           l1:=l2;
           i3:=i1
          end l2<l1
        end i2;
      if l1 < path[i]
        then
        begin
         path[i]:=l1;
         precede[i]:=i3
        end l1<path[i];
      if l1 < min
        then
        begin
         fii:=k11;
         min:=l1;
         f[k11]:=i
        end l1<min
        else
         if l1=min
          then
          begin
           fii:=fii-k2;
           f[fii]:=i
          end l1=min;
    end scan[i]=0,i;
  if min=M1
   then go to end;
  for i2:=k11 step -k2 until fii do
   begin
```

```
          i1:=f[i2];
          if i1=t
           then go to end;
          scan[i1]:=1
         end i2;
        count:=count+(if k11=1 then fii else n-fii+1);
        fi:=k1;
        k1:=k11;
        k11:=fi;
        k2:=-k2;
        fi:=fii
       end min;
    end:
     end modpaths
```

These results state the following facts:

(a) For networks in which the minimal distances from the fixed node to all other nodes are different, the Dijkstra method is better than its modification (e.g., for $n = 10, 20$); for networks with nodes equidistant from node s the contrary holds (e.g., for $n > 20$).

(b) The Dijkstra method and its modification are more effective than the Minty method (cf. [3] and [4]) (procedure *netpath* [1]).

(c) For greater networks ($n > 20$) the mean computer running time for the Nicholson method (cf. [5]) (procedure *minpath* [1]) is smaller than for all other methods.

References

[1] J. Boothroyd, *Algorithms 22, 23, 24*, Computer J. 10 (1967), p. 306-308.

[2] G. B. Dantzig, W. O. Blattner and M. R. Rao, *All shortest routes from a fixed origin in a graph*, in *Théorie des Graphes*, Proceedings of the International Symposium, Rome, Italy, July 1966, Dunod, Paris 1967, p. 85-90.

[3] S. E. Dreyfus, *An appraisal of some shortest-path algorithms*, Opns. Res. 17 (1969), p. 396-412.

[4] L. R. Ford, Jr., and D. R. Fulkerson, *Flows in networks*, Princeton, N. J., 1962 (Polish edition: *Przeplywy w sieciach*, PWN, Warszawa 1969).

[5] T. A. J. Nicholson, *Finding the shortest route between two points in a network*, Computer J. 9 (1966), p. 175-180.

DEPT. OF NUMERICAL METHODS
UNIVERSITY OF WROCŁAW

Received on 10. 11. 1971

M. SYSŁO (Wrocław)

WYZNACZANIE NAJKRÓTSZYCH DRÓG Z USTALONEGO WIERZCHOŁKA W SIECI

STRESZCZENIE

Procedura *paths*, oparta na metodzie Dijkstry (patrz [3]), i *modpaths*, realizująca przedstawioną modyfikację metody Dijkstry, wyznaczają

(i) najkrótszą drogę między wierzchołkami sieci s i t, jeśli $t > 0$,

(ii) najkrótsze drogi między wierzchołkiem s i wszystkimi pozostałymi wierzchołkami sieci, jeśli $t < 0$.

Dane:

n	– liczba wierzchołków sieci,
$d\,[1:n,\ 1:n]$	– tablica zawierająca długości całkowite i nieujemne połączeń między wierzchołkami sieci; elementy $d[i,j]$, dla których połączenie od i-tego wierzchołka do j-tego wierzchołka nie istnieje, powinny być nie mniejsze od M,
M	– wyrażenie o wartości nie mniejszej od $n \times \max\ d[k,l]$, gdzie maksimum wzięto po istniejących połączeniach,
s	– numer początkowego wierzchołka szukanych dróg,
t	– w przypadku (i): numer wierzchołka końcowego ($t \neq s$), w przypadku (ii): ujemna liczba całkowita.

Wyniki:

path$[1:n]$	– w przypadku (i) : *path*$[t]$ jest długością najkrótszej drogi między wierzchołkami s i t, w przypadku (ii) : *path* jest tablicą długości najkrótszych dróg z wierzchołka s-tego do wszystkich pozostałych wierzchołków sieci (*path*$[s] = 0$),
precede$[1:n]$	– *precede*$[k]$ jest numerem przedostatniego wierzchołka na najkrótszej drodze prowadzącej do k-tego wierzchołka (*precede*$[s] = 0$).

Uwaga. Wierzchołki tworzące najkrótszą drogę można wyznaczyć za pomocą procedury *shortpath* [1].

Obliczenia kontrolne, wykonane na maszynie cyfrowej ODRA 1204, wykazały poprawność procedur *paths* i *modpaths*.