Математички Билтен
16 (XLII)
1992 (99-104)
Скопје, Македонија

FINDING ALL MINIMUM-HOP PATHS IN NETWORKS

Iskra K. Djonova-Popova, Oliver B. Popov

Abstract. The problem of finding all minimum-hop paths from one node to another arises in several contexts for adaptive routing in computer communication networks. This paper presents an efficient algorithm for determining all paths with minimum number of links between two nodes in a network. Polynomial bound are established for the worst case time complexity of the algorithm. Directions for further research are also proposed.

1. Introduction. The functionality of computer communication networks vitally depends on routing, i.e. on finding a proper path between two nodes in a network. Hence, there has been extensive research interest in algorithms for single path routing. This follows from the fact that routing schemes in now-a-days networks all turn out to be variants, in one form or another, of shortest path algorithms that route packets from source to destination over a path of least cost, [1].

The development of more sophisticated adaptive routing techniques, [2], [3], [4], arose the interest for algorithms that find multiple paths between pairs of nodes in a network. Multiple disjoint paths can increase the effective bandwith between pairs of nodes, reduce the congestion in a network and decrease the probability of dropped packets. However, having in mind the reliability and the efficiency in transportation, it is desirable to route packets over minimum-hop paths, namely paths with a minimum number of links, [3], [4].

A considerable effort has been devoted to the design of algorithms for finding multipe paths from every node to a destination node in a network. For instance, one solution, [5], finds multiple paths that are initial-disjoint (i.e. disjoint-in the first link), another scheme, [6], finds a pair of disjoint paths of minimum total cost from every node to a destination, while the third, [7] obtains multiple disjoint paths.

In the paper, a simple extension of Dijkstra's algorithm for single source shortest path problem is given, [8], in order to obtain all minimum-hop paths from a given source to all other nodes in a network. The paper is organized as follows. In section 2 we present the basic design of the algorithm, while section 3 deals with the running time of the algorithm. Finally, in section 4 the conclusion and directions for further research are put forward.

2. Constructing Minimum-Hop Paths. In this section, an approach for constructing all minimum-hop paths from source node to every other node in a network is proposed.

2.1. Preliminaires. A network is modeled as a graph, $G(V,E)$, with a vertex set, $V=\{1,2,\ldots,n\}$, and a link set E. The graph contains neither paralel links nor loops. There is an arbitrary and distinguished node, s, in V called the source node. Without loss of generallity, node 1 is refered to be the source. A link in E connects a pair of nodes, u and v, in V and is denoted by (u,v). The lenght associated with each link (u,v), $l(u,v)$ is the distance between links u and v, $l(u,v)=\delta(u,v)=1$. If there is no link between nodes u and v we assume $l(u,v)$ is ∞, some value much larger than any actual distance. The number of links incident with a vertex v represents the degree of the vertex and is denoted $d(v)$.

Unlike Dijkstra's algorithm that is designed for directed graphs, the proposed scheme works on both directed and undirected graphs. A directed graph $G'(V,E')$, $E'=2E$, can be always associated with the graph G, such that each link (u,v) is presented with two directed arcs (u,v) and (v,u) each with length 1, $l(u,v)=l(v,u)=1$.

2.2. Algorithm. The algorithm actually performs a breadth-first search upon a given graph. It works by maintaining a set S of vertices with minimum number of links in the path from the source s. Since all links have length 1 we can always find all nodes that do not belong to S and are by one link away from the

```
procedure Minimum-Hop Paths
    { The procedure computes the distance of the minimum-hop
    path from vertex 1 to every vertex of a graph and finds all
    minimum-hop paths}
begin
(1)   S:={1};
(2)   for i:=2 to n do
(3)       D(i):=l(1,i); {initialize D}
(4)       P(i,j):=0, j=1,...,d(i); {initialize P}
(5)       δ:=0 {initilize the distance from the source to the most distante
              nodes in S}
(6)   do while S ≠ V
(7)       δ:=δ+1
(8)       find all vertices w_j in V-S such that D(w_j)=δ
(9)       add all w_j to S
(10)      for each vertex v in V-S do
(11)          for all w_j in S for which D(w_j)=δ do
(12)              D_j(v):=min[D(v), D(w_j)+l(w_j,v)]
(13)              do if D_j(v)<D(v)
(14)                  P(v,j):=w_j
(15)              end
              end
(16)          D(v):=min_j[D_j(v)]
          end
      end
end
```

Fig. 1

maximum distant nodes that belong to S. That is, at the k-th step of the algorithm it discovers all vertices that are at distance k from s. We use an array D to record the length of the minimum-hop path to each vertex, and a multidimensional array P to record the predecessors of the vertices in all minimum-hop paths. An element of P, P(v,j), v=1,...,n, j=1,...,d(v), contains the vertices immediately before vertex v in the j-th minimum-hop path or 0, when the j-th minimum-hop path does not exist. The number of minimum-hop paths from s to v is determi-

ned by the number of the nonzero elements P(v,j) and the number of minimum-hop paths from s to each predecessor of v. The algorithm itself is given in Fig. 1.

Upon termination of the algorithm the paths to each vertex can be found by tracing backward the predecessors in the P-array.

2.3. Example. Let us apply the algorithm to the graph on Fig. 2. Initially, set S={1}, D(2)=D(4)=1, D(3)=D(5)=D(6)=∞. In

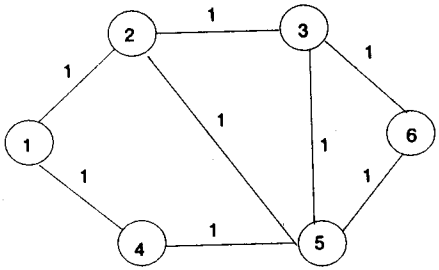

Fig. 2

the first iteration of the do while loop of lines (5) to (9) $w_1=2$ and $w_2=4$ are selected as the vertices that have distance $\delta=1$ from the source. Then, we set:

$D_2(3) = \text{Min}[\infty,1+1]=2$, $D_4(3) = \text{Min}[\infty,\infty]=\infty$,
$D(3) = \text{Min}[D_2(3),D_4(3)] = \text{Min}[2,\infty]=2$,
$D_2(5) = \text{Min}[\infty,1+1]=2$, $D_4(5) = \text{Min}[\infty,1+1]=2$,
$D(5) = \text{Min}[D_2(5),D_4(5)] = \text{Min}[2,2]=2$
D(6) do not change, because it has no link directly connected neither to 2 nor to 4.

In the second iteration $w_1=3$ and $w_2=4$ are selected as the vertices that have distance $\delta=2$ from the source. Then we set:

$$D_3(6) = \text{Min}[\infty,2+1] = 3,\ D_5(6) = \text{Min}[\infty,2+1) = 3$$

$$D(6) = \text{Min}[D_3(6),D_2(6)] = \text{Min}[3,3] = 3$$

In the third iteration $w_1=6$ is selected as the vertex that have distance $\delta=3$ from the source, and the algorithm terminates.

The nonzero elements of P-array would have the values P(2,1)=1, P(3,1)=2, P(4,1)=1, P(5,1)=2, P(5,2)=4, P(6,1)=3, P(6,2)=5. To find all minimum-hop paths from vertex 1 to vertex 6 for example, we would trace the predecessors in reverse order beginning at vertex 6. From the P-array we determine 3 and 5 as the predecessors of 6, 2 as the predecessor of 3 and 5 and 4 as the predecessor of 5, then 1 as the predecessor of 2 and 4. Thus, there are 3 minimum-hop paths from vertex 1 to vertex 6, all of equal distance δ=3. They are: 1,2,3,6; 1,2,5,6; 1,4,5,6. The first and the third path are disjoint because they have no intermediate common node.

3. The running time of the algorithm. The time complexity of the algorithm can be derived from the following notions:

1. In the case when the graph with n vertices is represented by an adjacency matrix, the loops of lines (11) till (14) take O(n) time, and it is executed at most n-1 times for the total running time of $O(n^2)$.

2. If the graph is sparce, i.e. the number of links, e, is much less then n^2, the use of adjacency list representation is recomended. For then, the loop of lines (11) till (14) can be implemented by going down the adjacency list for w_j, and it takes time which is proportional to the degree of w_j. When this quantity is summed over all w_j, at most O(e) time is spent in complete scanning of the adjacency list. Lines (1) to (5) take O(n) times, as do lines (6) to (9). The result yields the total time to be O(e+n). This running time is considerably better then $O(n^2)$ if e is very small compared with n^2.

4. Conclusions and directions for further research. In this paper, basicaly a breadth-first search technique for finding all minimum-hop paths is described. It is actually an extension of Dijkstra's algorithm for single source shortest path problem. The algorithm might prove to be extremely useful when implementing adaptive multipath routing that uses minimum-hop paths in a wide area network. The upper bound of the time complexity of the algorithm is polynomial. Practical realization and imple-

mentaion upon randomly chosen networks will be the next step towards the verification of its efficiency. The average running time is also the subject to further investigations.

Some of the routing algorithms, [4], use the least-hop and the least-hop+1 paths as the most appropriate paths for routing. Hence, it is also interesting to find out if the algorithm can be modified in the way that all minimum-hop paths and all minimum-hop+1 paths are found.

R E F E R E N C E S

[1] Schwartz M., Stern T.E.: Routing Techniques Used in Computer Communication Networks. IEEE Trans. Commun. COM-28:539-552, 1980

[2] Rudin H.: On Routing and "Delta Routing": A taxonomy and Performance Comparision of Techniques for Packet-Switched Networks. IEEE Trans. Commun. COM-24:43-59, 1976

[3] Thaker G.H., Cain J.B.: Interactions Between Routing and Flow Control Algorithms. IEEE Trans. Commun. COM-34:269-277, 1986

[4] Nelson D.J, Sayood K., Chang H.: An Extended Least-Hop Distributed Routing Algorithm. IEEE Trans. Commun. COM-38: 520-528, 1990

[5] Topkis D.M.: A k Shortest Path Algorithm for Adaptive Routing in Communications Networks. IEEE Trans. Commun., COM-36: 855-859, 1988

[6] Surballe J.W., Tarjan R.E.: A quick method of finding shortest pairs of disjoint paths. Networks, 14, 1984

[7] Sidhu D.P., Nair R., Abdallah S.: Finding Disjoint Paths in Networks. ACM Computer Comm. Review. 3:43-51, 1991

[8] Dijkstra E.: A note on two problems in connection with graphs. Numer. Math. vol. 1:269-271, 1959

ОДРЕДУВАЊЕ НА СИТЕ ПАТИШТА СО МИНИМАЛЕН БРОЈ НА ЛИНИИ ВО МРЕЖИ

И. Џонова-Попова, О. Попов

Р е з и м е

Во мрежите за пренос на податоци честопати се јавува проблемот на одредување на сите патишта од еден јазол до друг кои содржат минимален број линии. Во овој труд е претставен едноставен алгоритам за одредување на таквите патишта. Одредена е комплексноста на алгоритмот која во најлош случај е полиномијална. Предложени се и насоки за понатамошно истражување.