European Journal of Operational Research 53 (1991) 402–404
North-Holland

Theory and Methodology

A linear time algorithm for the maximum capacity path problem

Abraham P. Punnen *

Department of Mathematics, Indian Institute of Technology, Kanpur 208 016, India

Received August 1989; revised December 1989

Abstract: The maximum capacity path problem is to find a path joining two fixed vertices of an edge weighted graph such that the minimum edge weight is maximized. The best known algorithm to solve this problem has a worst-case complexity of $O(\min\{m + n\log n,\ m\log_n W\})$, where m is the number of edges, n is the number of vertices and W is the maximum edge capacity. We give a simple $O(m)$ algorithm to solve this problem. As a by-product of this result, we have an $O(m^2)$ algorithm for a bicriteria path problem which is an improvement over the available $O(m^2\log n)$ algorithm.

Keywords: Networks, shortest paths, maxmin paths, linear time algorithm, computational complexity

1. Introduction

Let G be an undirected graph with vertex set $V(G)$ and edge set $E(G)$ such that $|V(G)| = n$ and $|E(G)| = m$. Let e_{ij} be the edge joining vertices i and j. For each $e_{ij} \in E(G)$ a weight w_{ij} is prescribed. Let s and t be two fixed vertices of G. Then the maximum capacity path problem or the bottleneck path problem can be stated as

$$\text{Maximize}_{P(s,t)\in P} \min_{e_{ij}\in P(s,t)} (w_{ij}),$$

where P is the set of all paths joining vertices s and t and $P(s, t)$ is a path joining s and t. We denote this problem by MCP.

The problem MCP has been studied by several research workers in various contexts. Ichimori et al. [11] used MCP to solve minimax flow problems. Berman and Handler [4] used MCP to obtain optimal minimax path of a single service unit on a network. Hansen [10] used MCP to solve certain bicriteria path problems and Lawler [12] pointed out applications of MCP in reliability theory. However, the algorithm used by all these authors to solve MCP is a modification of the Dijkstra algorithm [6] which has a running time of order n^2. Several modified versions of the Dijkstra algorithm [2,3,7] are now available with better worst-case time bound and almost all of these algorithms can be easily modified to solve MCP also. The best known algorithm today to solve MCP is the one due to Gabow [8] with a complexity of $O(\min\{m + n\log n,\ m\log_n W\})$, where W is the maximum edge weight of G.

In this note we present an $O(m)$ algorithm to solve MCP. This improves all the existing algorithms for MCP and is the best possible. As a consequence of this result we have an $O(m^2)$ algorithm to solve the bicriteria minimax–maxmin path problem defined on an *undirected* graph which improves the $O(m^2\log n)$ bound given by

* present address: CORE, 34 Voie du Roman Pays, 1348 Louvain-La-Neuve, Belgium.

Hansen for this problem [10]. It may be noted that Hansen's paper deals with directed graphs. But all his results are applicable to the undirected graphs also and the improvement mentioned above is applicable only to undirected graphs.

2. Development of the algorithm

We first assume that the edge weights are distinct. However this is not a restriction on the algorithm and is only for the simplicity of presentation. If the edge weights are not distinct one can number the edges arbitrarily and the edge weights can be considered as ordered pairs (edge weight, edge number) with a lexicographic order relation among these modified weights.

For any real number k, let $G(k)$ be the graph with vertex set $V(G(k)) = V(G)$ and the edge set

$$E(G(k)) = \{e_{ij} \in E(G) \colon w_{ij} \geq k\}.$$

Let $G_1(k), G_2(k), \ldots, G_q(k)$ be the connected components of $G(k)$. For $1 \leq u \leq q$ and $1 \leq v \leq q$, let,

$$S_k(u, v) = \{e_{ij} \in E(G) \colon i \in V(G_u(k)) \text{ and } j \in V(G_v(k))\}.$$

Consider the graph $\bar{G}(k)$ with $V(\bar{G}(k)) = \{1, 2, \ldots, q\}$ and

$$E(\bar{G}(k)) = \{e_{uv} \colon S_k(u, v) \neq \emptyset\}.$$

The weight of the edge e_{uv} of the graph $\bar{G}(k)$ is

$$\operatorname*{Max}_{e_{ij} \in S_k(u,v)} \{w_{ij}\}.$$

Let the optimum objective function value of MCP on the graph G with s and t as fixed vertices be denoted by MAXMIN(G, s, t).

We now state a lemma which helps us to develop an algorithm to solve MCP.

Lemma 1. *For any real number k,*

(a) *If s and t are in the same connected component $G_x(k)$ of $G(k)$ then* MAXMIN$(G, s, t) =$ MAXMIN$(G_x(k), s, t)$.

(b) *If s and t are in different connected components of G, say in $G_x(k)$ and in $G_y(k)$ respectively, then* MAXMIN$(G, s, t) =$ MAXMIN$(\bar{g}(k), x, y)$.

The proof of the above lemma is simple and we omit it. We now present a recursive procedure PATH(G, s, t) which identifies MAXMIN(G, s, t). Without loss of generality we assume that G is connected.

```
Procedure PATH(G, s, t)
begin
  if $G$ contains a single edge only then return its
      weight
  else
  begin
    $W := \{w_{ij} \colon e_{ij} \in E(G)\}$;
    $k :=$ Median of $W$;
    Compute the graph $G(k)$;
    (* this can be done as mentioned earlier in
        text *)
      if $s$ and $t$ are in the same connected
          component of $G(k)$ then
        begin
          $\underline{G}$ := the unique component of $G(k)$
              containing $s$ and $t$;
          return PATH($\underline{G}, s, t$);
        end
      else
        begin
          Construct $\bar{G}(k)$;
          (* this can be done as mentioned
              earlier in text *)
          (* let $s \in G_x(k)$ and $t \in G_y(k)$ *)
          return PATH($\bar{G}(k), x, y$);
        end
  end;
end.
```

Lemma 1 insures that the procedure PATH (G, s, t) correctly identifies MAXMIN(G, s, t). In each recursive call of PATH(G, s, t) the computation of k needs $O(|E(G)|)$ time [1]. From the definition, it can be seen that the construction of $\bar{G}(k)$ and $G(k)$ takes $O(|E(G)|)$ time. Identification of $\underline{G}$ can be done in linear time. Further after each recursive call we obtain a new graph whose number of edges is half of that of the graph in the previous call. Thus the overall complexity of PATH(G, s, t) is $O(m + \frac{1}{2}m + \frac{1}{4}m + \cdots) = O(m)$.

Now after identifying MAXMIN(G, s, t) construct the graph $G($MAXMIN$(G, s, t))$. It is easy to see that any path joining vertices s and t in $G($MAXMIN$(G, s, t))$ is an optimal solution to MCP. Thus we have

Theorem. *In a graph with m edges, the maximum capacity path problem can be solved in* O(m) *time.*

Unfortunately, the above approach is not likely to be applicable to obtain an O(m) algorithm to find the maximum capacity path from s to all other vertices. However, it can be seen that a maximum spanning tree will give the maximum capacity path from any given vertex to all other vertices. The best known algorithm for the maximum spanning tree problem is the one suggested by Gabow et al. [9] and is not very likely to solve maximum spanning tree problem in O(m) time [9]. It may be noted that the spanning tree of maxmin paths rooted at s need not be a maximum spanning tree. Thus the following question is interesting: "Is it possible to identify the maximum capacity path from a fixed vertex to all other vertices more efficiently than solving a maximum spanning tree problem?"

Interestingly, the linear time algorithm proposed in this note can be used as a subroutine in the algorithm proposed by Hansen [10] to solve bicriteria minimax–maxmin path problem on an undirected graph and its equivalent versions. This will reduce the complexity of Hansen's algorithm (restricted to undirected graphs) to O(m^2) as compared to his O($m^2 \log n$) bound.

3. Conclusion

In this note we have given a best possible algorithm to solve MCP which improves all the existing algorithms for the problem. Our result shows that the complexity of MCP (which is a bottleneck problem) is the same as that of the corresponding feasibility (here the existence of a path joining s and t) problem. Camerini [5] proved a similar result for the case of bottleneck spanning tree problem. It may be pointed out that not all bottleneck problems, which are efficiently solvable, have this property. For example, the existence of a perfect matching in a bipartite graph G with n vertices and m edges can be tested in O($n^{0.5}m$) time [2]. But no O($n^{0.5}m$) algorithm is known to solve the bottleneck perfect matching problem on a bipartite graph. Thus it is worth to identify bottleneck problems which have the same complexity as that of the corresponding feasibility problem.

Acknowledgments

I wish to thank Prof. S.K. Gupta and Dr. R.K. Ahuja for their encouragement. Dr. Ahuja pointed out reference [8] to my notice. I am thankful to Prof. W.H. Cunningham for pointing out an error in an earlier version of this paper. Comments of the referees were also very helpful.

Note (by the author in proof). We have recently learned that in a paper by Gabow and Tarjan (*Journal of Algorithms* 9 (1988) 411–417) it is mentioned as a private communication that Lesly Matheson has a linear time algorithm for MCP based on [5]. Since our algorithm is also inspired by [5], there is a possibility of duplication, although we could not verify it. However both the works were independent.

References

[1] Aho, A.V., Hopcroft, J.E., and Ullman, J.D., *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[2] Ahuja, R.K., Magnanti, T.L., and Orlin, J.B., "Network flows", Working paper, Operations Research Center, MIT, Cambridge, MA, 1988.

[3] Ahuja, R.K., Mehlhorn, K., Orlin, J.B., and Tarjan, R.E., "Faster algorithms for the shortest path problem", Technical Report No. 193, Operations Research Center, MIT, Cambridge, MA, 1988.

[4] Berman, O., and Handler, G.Y., "Optimal minimax path of a single service unit on a network to nonservice destinations", *Transportation Science* 21 (1987), 115–122.

[5] Camerini, P.M., "The minimax spanning tree problem and some extensions", *Information Processing Letters* 7 (1978), 10–14.

[6] Dijkstra, E.W., "A note on two problems in connexion with graphs", *Numerische Mathematik* 1 (1959), 269–271.

[7] Fredman, M.L. and Tarjan, R.E., "Fibonacci heaps and their uses in improved network optimization algorithms", *Journal of the ACM* 34 (1987), 596–615.

[8] Gabow, H.N., "Scaling algorithms for network problems", *Journal of Computers and Systems Science* 31 (1985), 148–168.

[9] Gabow, H.N., Galil, Z., Spencer, T., and Tarjan, R.E., "Efficient algorithms for finding minimum spanning trees in directed and undirected graphs", *Combinatorica* 6 (1986), 109–122.

[10] Hansen, P., "Bicriterion path problems", *Lecture notes in Economics and Mathematical Systems,* No. 177, Springer Verlag, Berlin, (1980), 109–127.

[11] Ichimori, T., Ishi, H., and Nishida, T., "A minimax flow problem", *Mathematica Japonica* 24 (1979), 65–71.

[12] Lawler, E.L., "*Combinatorial Optimization: Networks and Matroids*", Holt, Rinehart and Winston, New York, 1976.