ELSEVIER

Information Processing Letters 79 (2001) 81–85

Information Processing Letters

www.elsevier.com/locate/ipl

A faster computation of the most vital edge of a shortest path ☆

Enrico Nardelli [a,b], Guido Proietti [a,b,*], Peter Widmayer [c]

[a] *Dipartimento di Matematica Pura ed Applicata, Università di L'Aquila, Via Vetoio, 67010 L'Aquila, Italy*
[b] *Istituto di Analisi dei Sistemi e Informatica, CNR, Viale Manzoni 30, 00185 Roma, Italy*
[c] *Institut für Theoretische Informatik, ETH Zentrum, 8092 Zürich, Switzerland*

Received 25 January 1999; received in revised form 16 June 2000
Communicated by S. Zaks

Abstract

Let $P_G(r,s)$ denote a shortest path between two nodes r and s in an undirected graph $G=(V,E)$ such that $|V|=n$ and $|E|=m$ and with a positive real length $w(e)$ associated with any $e \in E$. In this paper we focus on the problem of finding an edge $e^* \in P_G(r,s)$ whose removal is such that the length of $P_{G-e^*}(r,s)$ is maximum, where $G-e^*=(V,E\setminus\{e^*\})$. Such an edge is known as the *most vital edge* of the path $P_G(r,s)$. We will show that this problem can be solved in $O(m\cdot\alpha(m,n))$ time, where α is the functional inverse of the Ackermann function, thus improving on the previous $O(m+n\log n)$ time bound.

Keywords: Graph algorithms; Shortest path; Most vital edge; Transient failure; Replacement shortest path

1. Introduction

Let $G=(V,E)$ be an undirected graph with $|V|=n$ vertices and $|E|=m$ edges, with a positive real length $w(e)$ associated with each edge $e \in E$. Given a source node r and a destination node s in G, a *shortest path* $P_G(r,s)$ from r to s in G is defined as a path which minimizes the sum of the lengths of the edges along the path from r to s. The length of $P_G(r,s)$ is called the *distance* in G between r and s and will be in the following denoted as $d_G(r,s)$.

The removal of an edge $e \in P_G(r,s)$ from the graph G results in a different—and perhaps longer—shortest path from r to s: $d_{G-e}(r,s) \geqslant d_G(r,s)$, where $G-e=(V,E\setminus\{e\})$. We call $P_{G-e}(r,s)$ a *replacement shortest path* for edge e. In the past, the problem of finding an edge in $P_G(r,s)$ whose removal from G results in the largest increase of the distance between r and s has been studied. This edge is generally denoted as the *most vital edge* with respect to the shortest path $P_G(r,s)$. For the sake of brevity, we will refer to this problem in the following as the *most vital edge* (MVE) problem. More precisely, the MVE problem with respect to $P_G(r,s)$ asks for finding an edge $e^* \in P_G(r,s)$ such that $d_{G-e^*}(r,s) \geqslant d_{G-e}(r,s)$, for any edge $e \in P_G(r,s)$.

The MVE problem has been solved efficiently by Malik et al. [1], who gave an $O(m+n\log n)$ time algorithm to compute all the replacement shortest paths between the source and the destination node in the pres-

☆ Work supported by the EU TMR Grant CHOROCHRONOS and by grant "Combinatorics and Geometry" of the Swiss National Science Foundation.

* Corresponding author.
E-mail addresses: nardelli@univaq.it (E. Nardelli), proietti@univaq.it (G. Proietti), widmayer@inf.ethz.ch (P. Widmayer).

0020-0190/01/$ – see front matter
PII: S0020-0190(00)00175-7

ence of edge failures along the (original) shortest path. As a byproduct of their solution, the most vital edge along the path is immediately obtained.

In this paper we improve the above result to $O(m \cdot \alpha(m, n))$ time, where α is the well-known functional inverse of the Ackermann function [4]. The improvement comes from the use of a linear time algorithm for the undirected single source shortest paths tree [7], combined in a novel way with the use of a *transmuter* [6]. Namely, we build all the replacement paths for any edge $e \in P_G(r, s)$, and we select the shortest by using a transmuter. Moreover, we also show that our approach allows to solve with the same time complexity the *longest-detour* (LD) problem [2], which asks for finding an edge $e^* = (u^*, v^*) \in P_G(r, s)$ whose removal produces a detour at node u^* such that the length of $P_{G-e^*}(u^*, s)$ minus the length of $P_G(u,^* s)$ is maximum, for any edge in $P_G(r, s)$.

Solving efficiently the MVE problem is important for dealing with transient failures on a communication network. Suppose in fact that the given graph models a communication network, and the shortest path we are focusing on represents the communication line between a source and a target of a message (the two endpoints of the path). Assume that sudden (transient) failures of links (i.e., edges) are possible in such a network. When this happens along the communication line between r and s and the link joining u and v goes down, messages should then be routed from r to s on a shortest path that does not use edge (u, v). Of course, from the network management point of view, it is important to know "a priori" both the most vital edge and the replacing shortest paths for all the edges along the path. Our approach allows to solve efficiently both problems.

In what follows, r and s are assumed to be 2-edge connected, so that for each edge e on $P_G(r, s)$, at least one alternative path exists. Otherwise, the MVE problem can be easily solved in $O(m)$ time by applying Tarjan's algorithm for finding the bridges of G [3]. In fact, removing a bridge between r and s will increase the length of any replacement shortest path to infinity. The computation model we use is a RAM, where the memory is divided into addressable words of length ω [7]. The edge lengths are represented as floating point numbers, each contained in O(1) words.

The paper is organized as follows: in Section 2 we solve efficiently the MVE problem; in Section 3 we show how to use the same approach for solving the LD problem; finally, Section 4 contains concluding remarks and lists some open problems.

2. Solving efficiently the MVE problem

Let $P_G(r, s)$ be a shortest path joining r and s in G. We start by computing the shortest paths trees rooted at r and s, denoted as $S_G(r)$ and $S_G(s)$, respectively. This can be done in $O(m)$ time and space [7]. Let $e = (u, v)$ be an edge on $P_G(r, s)$, with u closer to r than v. Let $M_r(e)$ denote the set of nodes reachable in $S_G(r)$ from r without passing through edge e and let $N_r(e) = V \setminus M_r(e)$ be the remaining nodes (i.e., the subtree of $S_G(r)$ rooted at v). Fig. 1 shows $N_r(e)$ and $M_r(e)$. Symmetrically, we define the sets $M_s(e)$ and $N_s(e)$ with respect to $S_G(s)$. Note that for the nodes in $M_r(e)$ ($M_s(e)$), the distance from r (s) does not change after deleting the edge e, while for the nodes in $N_r(e)$ ($N_s(e)$) the distance from r (s) may increase as a consequence of deleting e.

$N_r(e)$ and $M_r(e)$ define a cut in G, and

$$C_r(e) = \big\{(x, y) \in E \setminus \{e\} \mid (x \in M_r(e)) \wedge (y \in N_r(e))\big\}$$

is the set of edges crossing the cut (*crossing edges*, for short). Since a replacement shortest path $P_{G-e}(r, s)$ joining r and s must contain an edge in $C_r(e)$, it follows that it corresponds to the set of edges whose lengths satisfy the condition

$$d_{G-e}(r, s) = \min_{f=(x,y)\in C_r(e)} \big\{d_{G-e}(r, x) + w(f) + d_{G-e}(y, s)\big\}.$$

Each individual term of the above expression is available in O(1) time for fixed (x, y), once $S_G(r)$

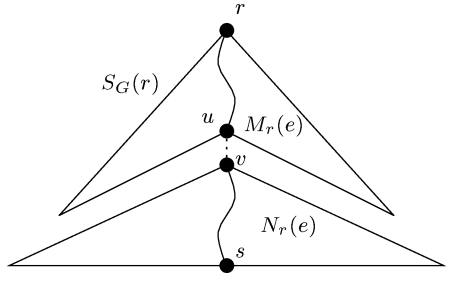

Fig. 1. $M_r(e)$ and $N_r(e)$.

and $S_G(s)$ have been computed. In fact, $d_{G-e}(r,x) = d_G(r,x)$, since $x \in M_r(e)$, and then can be obtained in O(1) time by looking at $S_G(r)$. For given f, $w(f)$ is available in O(1) time. Concerning $d_{G-e}(y,s)$, the following holds:

Lemma 1. *Let $f = (x,y) \in C_r(e)$. Then, we have that $y \in M_s(e)$.*

Proof. Suppose, for the sake of contradiction, that $y \notin M_s(e)$, i.e., $y \in N_s(e)$. Therefore, y is a descendant of u (and v) in $S_G(s)$. This means that $P_G(s,y)$ makes use of e, and then we have (since subpaths of shortest paths are shortest paths) that $P_G(v,y)$ is a subpath of $P_G(s,y)$. Therefore

$$d_G(v,y) = w(e) + d_G(u,y) > d_G(u,y).$$

On the other hand, since $y \in N_r(e)$, we have that $P_G(r,y)$ makes use of (u,v), and then we have that $P_G(u,y)$ is a subpath of $P_G(r,y)$. Hence

$$d_G(u,y) = w(e) + d_G(v,y) > d_G(v,y),$$

that is, we have a contradiction. □

Since $y \in M_s(e)$, we conclude that its distance from s remains unchanged after deleting the edge $e = (u,v)$, that is, $d_{G-e}(y,s) = d_G(y,s)$, and then it can be obtained in O(1) time by looking at $S_G(s)$. Fig. 2 illustrates the situation.

Therefore, it remains to establish the minimum over all the crossing edges. To do this efficiently, we make use of a *transmuter* [6]. A transmuter $D_G(T)$ is a directed acyclic graph that represents the set of fundamental cycles of a graph G with respect to a spanning tree $T = (V, E_T)$. The transmuter $D_G(T)$ contains for each tree edge e one source node v_e, for each non-tree edge f one sink node v_f, and a certain number of additional nodes. The fundamental property of a transmuter is that there is a directed path from a given source v_e to a given sink v_f if and only if e and f form a cycle in $T + f = (V, E_T \cup \{f\})$. It is clear that all and only edges belonging to $C_r(e)$ form a cycle with e. Therefore, we can build a transmuter having as source nodes all the edges belonging to $S_G(r)$ and as sink nodes all the remaining edges. This can be done in $O(m \cdot \alpha(m,n))$ time and space [6]. Given such a transmuter, we can solve the MVE problem by labeling in O(1) time a sink node v_f, associated with a non-tree edge $f = (x,y)$, with the cost

$$c(v_f) = d_G(r,x) + w(f) + d_G(y,s).$$

Afterwards, we process the nodes of the transmuter in reverse topological order, labeling each node with the minimum of the labels of its immediate successors. When the process is complete, a source node v_e, associated with a tree edge $e \in P_G(r,s)$, is labeled with a cost $c(v_e)$ corresponding to the length of a shortest path from r to s not using e. Finally, the most vital edge e^* of $P_G(r,s)$ can be easily obtained in O(n) time as the edge such that

$$c(v_{e^*}) = \max_{e \in P_G(r,s)} \{c(v_e)\}.$$

Therefore, the following result can be stated:

Theorem 1. *The most vital edge on a shortest path $P_G(r,s)$ between two nodes r and s in a graph $G = (V,E)$ with n vertices and m edges, with positive real edge lengths, can be determined in $O(m \cdot \alpha(m,n))$ time and space.*

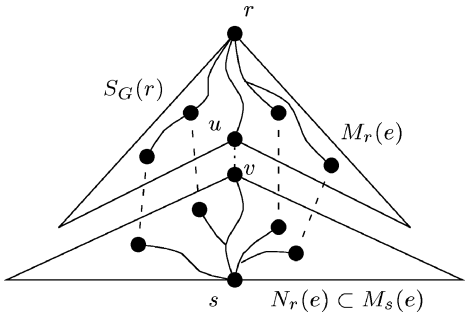

Fig. 2. Edge $e = (u,v) \in P_G(r,s)$ is removed from G. Dashed lines represent crossing edges.

3. Solving efficiently the LD problem

In this section we illustrate how to make use of the technique developed in the previous section to solve an interesting variation of the MVE problem: the *longest-detour* (LD) problem [2], which asks for finding an edge $e^* = (u^*, v^*) \in P_G(r,s)$ whose removal produces a detour at node u^* such that the length of $P_{G-e^*}(u^*,s)$ minus the length of $P_G(u^*,s)$

is maximum, for any edge in $P_G(r,s)$. Such an edge is called a *detour-critical edge*. The problem is interesting since in communication networks, when a message is routed along the path $P_G(r,s)$, if a sudden (transient) failure of a link $e=(u,v)$ in such a path occurs, then the message cannot continue on its path as intended, as the outgoing edge (u,v) to be taken is currently not operational. The message should then be routed from u to s on a shortest path that does not use edge (u,v). This problem has been solved in $O(m+n\log n)$ time [2].

However, by using a transmuter, we can solve the LD problem in $O(m\cdot\alpha(m,n))$ time, as follows:

Theorem 2. *The detour-critical edge on a shortest path $P_G(r,s)$ between two nodes r and s in a graph $G=(V,E)$ with n vertices and m edges, with positive real edge lengths, can be determined in $O(m\cdot\alpha(m,n))$ time and space.*

Proof. As for the MVE problem, we start by computing in $O(m)$ time and space the shortest paths trees rooted at r and s, denoted as $S_G(r)$ and $S_G(s)$, respectively. Maintaining the same notations as above, we now consider the cut $C_s(e)$ induced by $M_s(e)$ and $N_s(e)$, with the corresponding crossing edges. Since the detour $P_{G-e}(u,s)$ joins $u\in N_s(e)$ with $s\in M_s(e)$, it must contain an edge in $C_s(e)$. Then, it follows that it corresponds to the set of edges whose lengths satisfy the condition

$$d_{G-e}(u,s) = \min_{f=(x,y)\in C_s(e)} \{d_{G-e}(u,x)+w(f)+d_{G-e}(y,s)\}. \quad (1)$$

Any term of the above expression can be evaluated in $O(1)$ time for fixed (x,y), once $S_G(r)$ and $S_G(s)$ have been computed. In fact, since $x\in N_s(e)$, we have

$$d_{G-e}(u,x)=d_G(s,x)-d_G(s,u),$$

and since $y\in M_s(e)$, we have

$$d_{G-e}(y,s)=d_G(y,s).$$

Therefore, it remains to establish the minimum over all the crossing edges. To do this efficiently, once again we make use of a transmuter. It is clear that all and only edges belonging to $C_s(e)$ form a cycle with e. Therefore, as for the MVE problem, to select the edge minimizing (1), we could build a transmuter associating with the source nodes all the edges belonging to $S_G(s)$, and with the sink nodes all the non-tree edges. However, there is a difficulty this time in associating a cost with sink nodes: in fact, if v_f is a sink node associated with a non-tree edge $f=(x,y)$ forming a cycle in $S_G(s)+f$ with $e_1=(u_0,u_1), e_2=(u_1,u_2),\ldots,e_k=(u_{k-1},u_k)$, $e_i\in P_G(r,s)$, $i=1,\ldots,k$, then, according to (1), it will have different costs depending on which e_i is considered. Hence, the question is: which value $c(v_f)$ in the transmuter should be associated with f, such that $c(v_f)$ is *independent* of e_i? To solve this problem, we associate with v_f the following cost depending only on edge f

$$c(v_f)=d_G(y,s)+w(f)+d_G(s,x),$$

and corresponding to the length of the shortest (not necessarily simple) cycle in $S_G(s)+f$ starting from s and passing through f. In fact, for any crossing edge f that replaces e, we have that

$$d_{G-e}(u,s)=c(v_f)-d_G(s,u),$$

and therefore, a shortest cycle (i.e., a cycle minimizing $c(v_f)$ for any crossing edge f) is associated with a shortest detour, and vice-versa. Fig. 3 illustrates the situation.

Afterwards, we process the nodes of the transmuter in reverse topological order. When the process is complete, each source node v_e associated with an edge $e\in P_G(r,s)$ is labeled with a cost $c(v_e)$ corresponding to the length of a shortest cycle in $S_G(s)$ starting from s and making use of an edge in $C_s(e)$. Since the length of the detour induced by the failure of an edge $e=$

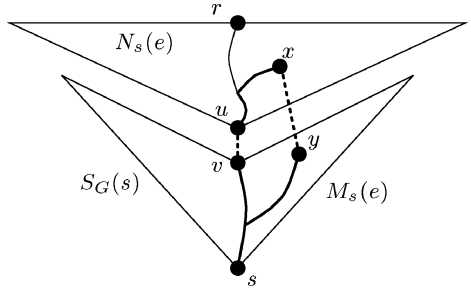

Fig. 3. Edge $e=(u,v)\in P_G(r,s)$ is removed from G. A crossing edge $f=(x,y)$ is associated with a (not necessarily simple) cycle starting from s and passing through f (in bold).

$(u, v) \in P_G(r, s)$ is $c(v_e) - d_G(s, u)$, it follows that the distance from u to s increases by $c(v_e) - 2d_G(s, u)$, and therefore the detour-critical edge of $P_G(r, s)$ can be obtained in $O(n)$ time as the edge $e^* = (u^*, v^*)$ such that

$$c(v_{e^*}) - 2d_G(s, u^*) = \max_{e=(u,v)\in P_G(r,s)} \{c(v_e) - 2d_G(s, u)\}. \quad \square$$

4. Conclusions

In this paper we have presented a faster solution to the problems of finding the most vital edge and the detour-critical edge along a shortest path $P_G(r, s)$ between two nodes r and s. Our solutions run in $O(m \cdot \alpha(m, n))$ time, where α is the functional inverse of the Ackermann function. The best previous bounds known in the literature were $O(m + n \log n)$ time [1,2].

Our solutions are efficient, but lower and upper bounds still do not match. However, a linear time algorithm is not allowed to use a transmuter over all the m edges, since the transmuter already has size $\Omega(m \cdot \alpha(m, n))$ [5], and therefore a different approach is needed.

Acknowledgements

The authors would like to thank the anonymous referees for their suggestions, which helped us in improving the paper.

References

[1] K. Malik, A.K. Mittal, S.K. Gupta, The k most vital arcs in the shortest path problem, Oper. Res. Lett. 8 (1989) 223–227.

[2] E. Nardelli, G. Proietti, P. Widmayer, Finding the detour-critical edge of a shortest path between two nodes, Inform. Process. Lett. 67 (1) (1998) 51–54.

[3] R.E. Tarjan, A note on finding the bridges of a graph, Inform. Process. Lett. 2 (1974) 160–161.

[4] R.E. Tarjan, Efficiency of a good but not linear set union algorithm, J. ACM 22 (1975) 215–225.

[5] R.E. Tarjan, Complexity of monotone networks of computing conjunctions, Ann. Discrete Math. 2 (1975) 121–133.

[6] R.E. Tarjan, Applications of path compression on balanced trees, J. ACM 26 (1979) 690–715.

[7] M. Thorup, Floats, integers, and single source shortest paths, in: Proc. 15th Annual Symposium on Theoretical Aspects of Computer Science (STACS'98), Paris, France, February 25–27, Lecture Notes in Comput. Sci., Vol. 1373, Springer, Berlin, 1998, pp. 14–24.