

Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Information Processing Letters ••• (••••) •••–•••

**Information
Processing
Letters**

www.elsevier.com/locate/ipl

Improved algorithm for finding next-to-shortest paths [☆]

Shisheng Li ^{*}, Guangzhong Sun, Guoliang Chen

National High Performance Computing Center at Hefei, Department of Computer Science, University of Science and Technology of China,
Anhui, Hefei 230027, PR China

Received 21 October 2005; received in revised form 3 March 2006; accepted 7 April 2006

Communicated by K. Iwama

Abstract

We study the problem of finding the next-to-shortest paths in a weighted undirected graph. A next-to-shortest (u, v) -path is a shortest (u, v) -path amongst (u, v) -paths with length strictly greater than the length of the shortest (u, v) -path. The first polynomial algorithm for this problem was presented in [I. Krasikov, S.D. Noble, Finding next-to-shortest paths in a graph, Inform. Process. Lett. 92 (2004) 117–119]. We improve the upper bound from $\mathcal{O}(n^3m)$ to $\mathcal{O}(n^3)$.

© 2006 Elsevier B.V. All rights reserved.

Keywords: Graph algorithms; Computational complexity; Shortest paths

1. Introduction

In the problem of finding the next-to-shortest paths, we are given a weighted undirected graph G and a pair of vertices (u, v) , we need to figure out a (u, v) -path (if exist) which is a shortest path amongst all (u, v) -paths with length strictly greater than the shortest (u, v) -path. It has been shown in [1] that this problem has efficient algorithms. In fact, [1] has given an algorithm with time complexity $\mathcal{O}(n^3m)$, where n is the number of vertices and m is the number of edges. In this paper, our main contribution is to present an algorithm with lower upper bound $\mathcal{O}(n^3)$.

Now, we firstly show our graph theoretical notations just as [1]. Graphs in this paper are restricted to being

simple, that is, having no loops or multiple edges. Formally, a graph is denoted with $G = (V, E)$ (G for short), where V is the vertex set of G , and E is the edge set. For every G , there is a function $l: E \rightarrow \mathcal{Q}^+$ giving the length of each edge of G . Note that by \mathcal{Q}^+ we mean the set $\{x: x \in \mathcal{Q}, x > 0\}$ so we do not allow edges with length zero. For each edge e of G , the length of e is denoted with $l(e)$. Symbols for a directed weighted graph $D(V, E)$ (D for short) is defined in similar way. To prevent confusion an edge of D is denoted with a .

A walk of G is an ordered list of vertices and edges $W = \langle v_0, e_1, v_1, \dots, e_k, v_k \rangle$ such that for $1 \leq i \leq k$, the endpoints of e_i are v_{i-1} and v_i . A path of G is a walk for which the vertices v_0, \dots, v_k are distinct. A (u, v) -walk (or path) is a walk (or path) for which $v_0 = u$ and $v_k = v$. The length of a walk W , which is denoted by $l(W)$, is summation of lengths of each edge e_i ($1 \leq i \leq k$). It is $l(W) = \sum_{i=1}^k l(e_i)$. The length of a path P is defined in similar way. Sometimes we will deal with a directed

[☆] This work is supported by National Science Foundation of China [No. 60533020].

^{*} Corresponding author.

E-mail address: shisheng@mail.ustc.edu.cn (S. Li).

graph D in which we require the edges are directed (so denoted with a , not e) and $a_i = (v_{i-1}, v_i)$.

Given a weighted undirected graph G , the *next-to-shortest* (u, v) -path, is the shortest (u, v) -path amongst those (u, v) -paths having length strictly greater than the length of the shortest (u, v) -path. We also need to identify the case in which no such path exists.

This paper is organized in four sections. In Section 2, we introduce some previous results on this problem. In Section 3, we show our main results. In the last section we give some conclusions.

2. Related works

To make our paper integral, in this section we list some related works before us.

In [2], the next-to-shortest path problem for a directed graph where we allow edges of length zero, has been shown to be NP-hard.

In [1], a new directed graph $D_G(u, v)$ for the input graph G has been introduced. $D_G(u, v)$ and G share the same vertices and a directed edge $a = (x, y)$ is in $D_G(u, v)$ if and only if there is at least one shortest (u, v) -path P with form of $\langle \dots, v_i = x, e_{i+1}, v_{i+1} = y, \dots \rangle$. The following property of $D_G(u, v)$ was proved in [1].

Lemma 1. *For all vertex w , if $\langle v_0 = u, a_1, v_1, \dots, v_{k-1}, a_k, v_k = w \rangle$ is a (u, w) -walk in $D_G(u, v)$ which is a directed acyclic graph, then $\langle v_0 = u, e_1, v_1, \dots, v_{k-1}, e_k, v_k = w \rangle$ is a shortest (u, w) -path in G .*

[1] also show the following lemma.

Lemma 2. *Given a weighted, undirected graph, G , and specified vertices u, v, w , there is a polynomial time algorithm to find the shortest length (u, v) -path passing through w .*

When proving Lemma 2 in [1], an algorithm with time complexity $\mathcal{O}(n^2)$ was given. So in fact the authors of [1] proved the following result.

Lemma 3. *Given a weighted, undirected graph, G , and specified vertices u, v, w , there is an $\mathcal{O}(n^2)$ time algorithm to find the shortest length (u, v) -path passing through w .*

Then an algorithm for the original problem with time complexity $\mathcal{O}(n^3m)$ was presented in [1].

3. Main results

This section is devoted to a proof of our main results.

Theorem 1. *There is an algorithm with time complexity $\mathcal{O}(n^3)$ solving the problem of finding the next-to-shortest path between a given pair of vertices in a weighted undirected graph.*

Suppose we have constructed the directed graph $D_G(u, v)$ for the input graph $G = (V, E)$, and there is a (u, v) -path $P = \langle v_0, e_1, v_1, \dots, v_k \rangle$ of G which is not a shortest one. From Lemma 1, there must exist an i ($0 \leq i \leq k-1$) for which a directed edge $a = (v_i, v_{i+1})$ is not in $D_G(u, v)$. Denote the smallest one with i_0 . Let us check the edge set $B(v_{i_0}) = \{e = (v_{i_0}, x), \text{ where } a = (v_{i_0}, x) \in D_G(u, v)\}$. It is trivial to show that P cannot pass any edge in $B(v_{i_0})$.

The above discussion on any (u, v) -path can be applied on a next-to-shortest path P . If we know what v_{i_0} of P is, we can delete any edge in $B(v_{i_0})$ from G without changing the existence of P .

We list our algorithm first, then prove its correctness and finally analysis its time complexity.

Algorithm 1 (NTSP).

- (1) Construct directed graph $D_G(u, v)$ for the input graph G , source u and destination v .
- (2) For each vertex w of G except v , delete all edges in $B(w)$, then find a shortest (u, v) -path P passing through w .
- (3) If we have not found any paths, return that there is no solution.
- (4) Return the shortest one amongst all paths we find in step 2.

Correctness. Firstly, we prove that G has a path with strictly greater length than the shortest one if and only if at least one path has been found in step 2. On one direction, if G has at least one path, called P' , with strictly greater length than the shortest one. Suppose $P' = \langle v_0, e_1, v_1, \dots, e_k, v_k \rangle$, and the smallest i for which directed edge $a = (v_i, v_{i+1})$ is not in $D_G(u, v)$ is i_0 (from Lemma 1, the existence of i_0 is obvious). From previous discussion, deleting all edges in $B(v_{i_0})$ does not change the existence of P' . So, in step 2, when we consider v_{i_0} as w , we can find a (u, v) -path. On the other direction, suppose we has found one (u, v) -path P'' in step 2 when we are checking vertex w . P'' must have the form $\langle \dots, v_i = w, e, v_{i+1}, \dots \rangle$ and directed edge

$a = (v_i, v_{i+1})$ cannot be in $D_G(u, v)$. From our definition of $D_G(u, v)$, we know that P'' cannot be a shortest path.

Secondly, we prove that the path returned in step 4 is a next-to-shortest path. From above we know that there is a next-to-shortest path in G if and only if a path is returned in step 4. Suppose $P = \langle v_0, e_1, v_1, \dots, e_k, v_k \rangle$ is a next-to-shortest (u, v) -path in G , and the smallest i for which directed edge $a = (v_i, v_{i+1})$ is not in $D_G(u, v)$ is i_0 . P cannot include any edge in $B(v_{i_0})$, so deleting all edges in $B(v_{i_0})$ from G does not change the existence of P . Suppose in step 2, when we check vertex v_{i_0} , the shortest (u, v) -path we find is Q , so $l(Q) \leq l(P)$. From the definition of a next-to-shortest path, we know that $l(P) \leq l(Q)$. So, $l(P) = l(Q)$ and Q is also a next-to-shortest path. That is, the path the algorithm return in step 4 must be a next-to-shortest (u, v) -path.

Time complexity analysis. From [1], in step 1 of Algorithm NTSP, the construction can be done in $\mathcal{O}(n^2)$ time. In each iteration of step 2, we need to find a shortest (u, v) -path passing through a given vertex x . From Lemma 3, this can be done in $\mathcal{O}(n^2)$ time. So the total

time complexity of our algorithm is $\mathcal{O}(n^2 + n * n^2) = \mathcal{O}(n^3)$.

4. Conclusion

In this paper we improve the upper bound of the problem of finding the next-to-shortest paths from $\mathcal{O}(n^3m)$ to $\mathcal{O}(n^3)$.

It is interesting to know if there is an algorithm with even lower bound. We can show that to find the next-to-shortest path is at least as harder as to find the shortest path. We guess that the two problems have efficient algorithms sharing the same time complexity.

Acknowledgements

We thank the anonymous referees for their useful suggestions to improve the presentation of this paper.

References

- [1] I. Krasikov, S.D. Noble, Finding next-to-shortest paths in a graph, Inform. Process. Lett. 92 (2004) 117–119.
- [2] K.N. Lalgudi, M.C. Papaefthymiou, Computing strictly-second shortest paths, Inform. Process. Lett. 63 (1997) 177–181.