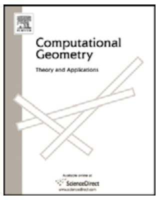
CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Elsevier - Publisher Connector

Computational Geometry 43 (2010) 68–72

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo



Computing the dilation of edge-augmented graphs in metric spaces

Christian Wulff-Nilsen

Department of Computer Science, University of Copenhagen, Copenhagen, Denmark

ARTICLE INFO

Article history:
Received 14 April 2008
Received in revised form 10 March 2009
Accepted 10 March 2009
Available online 23 June 2009
Communicated by S. Lazard

Keywords:
Dilation
Stretch factor
Best shortcut
Edge augmentation
Metric space

ABSTRACT

Let $G = (V, E)$ be an undirected graph with n vertices embedded in a metric space. We consider the problem of adding a shortcut edge in G that minimizes the dilation of the resulting graph. The fastest algorithm to date for this problem has $O(n^4)$ running time and uses $O(n^2)$ space. We show how to improve the running time to $O(n^3 \log n)$ while maintaining quadratic space requirement. In fact, our algorithm not only determines the best shortcut but computes the dilation of $G \cup \{(u, v)\}$ for every pair of distinct vertices u and v.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

In areas such as VLSI design, telecommunication, and distributed systems, a problem often arising is that of interconnecting a set of sites in a network of small cost. There are many different ways of measuring the cost of a network, such as its size, total length, minimum and maximum degree, diameter, and dilation (also known as stretch factor).

Spanners are sparse or economic representations of networks, making them important geometric structures in the areas mentioned above. They have received a great deal of attention in recent years, see e.g. [7,2,8].

A *t-spanner* is a graph embedded in a metric space such that, for any pair of vertices in this graph, the graph distance between them is at most t times their metric distance. The smallest t such that a graph is a t-spanner is called the *dilation* of the graph.

Most algorithms construct networks from scratch, but frequently one is interested in extending an already given network with a number of edges such that the dilation of the resulting network is minimized.

Farshi et al. [3] considered the following problem: given a graph $G = (V, E)$ with n vertices embedded in a metric space, find a vertex pair $(u, v) \in V \times V$ (called a shortcut) such that the dilation of $G \cup \{(u, v)\}$ is minimized. They gave a trivial $O(n^4)$ time and $O(n^2)$ space algorithm for this problem together with various approximation algorithms.

In this paper, we present an $O(n^3 \log n)$ time and $O(n^2)$ space algorithm for the above problem. This algorithm not only computes the best shortcut but returns a table T with a row and a column for every vertex in G such that for any pair of distinct vertices u and v, $T(u, v)$ is the dilation of $G \cup \{(u, v)\}$.

The organization of the paper is as follows. In Section 2, we give various basic definitions and assumptions. In Section 3, we present one of the key ideas of the paper which gives a more efficient way of obtaining the dilation of edge-augmented graphs. We present our algorithm and prove its correctness in Sections 4 and 5, we show that the above time and space bounds hold. In Section 6, we consider the case where the given graph is disconnected. Finally, we make some concluding remarks and pose open problems in Section 7.

E-mail address: koolooz@diku.dk.

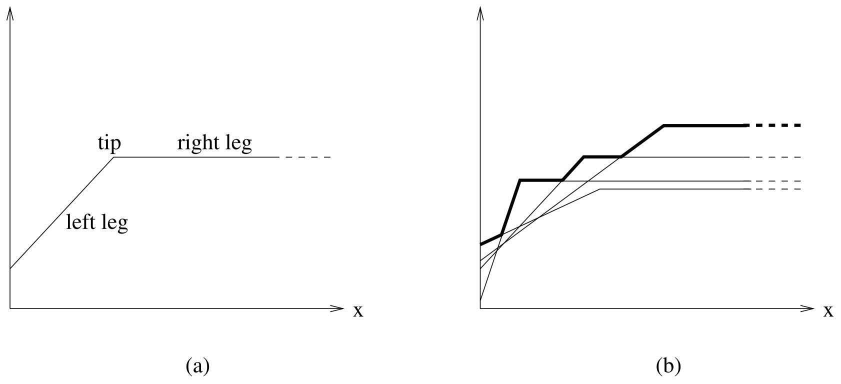


Fig. 1. (a) A staircase step and (b) the upper envelope (thick line segments) of a set of four staircase steps.

2. Basic definitions and assumptions

Let $G = (V, E)$ be an undirected graph embedded in metric space (V, d) and assume that G is connected.

Given two vertices $u, v \in V$, a *shortest path* between u and v is a path in G between u and v for which the sum of lengths of the edges of the path is minimal. We denote by $d_G(u, v)$ the length of such a path.

We define the *dilation* $\delta_G(u, v)$ of a pair of distinct vertices $u, v \in V$ as $d_G(u, v)/d(u, v)$. The dilation of G is defined as

$$\delta_G = \max_{u,v \in V,\, u \neq v} \delta_G(u, v).$$

In the following, G denotes an undirected, connected graph (V, E) embedded in metric space (V, d) and we define $n = |V|$.

3. Upper envelope of functions

In this section, we consider the upper envelope of certain functions which will help us to compute the dilation of graphs obtained from G by the addition of a shortcut (a single edge).

Let u, v, and w_1 be three fixed vertices of G such that $u \neq v$ and let w_2 be a fourth vertex of G (not fixed).

Let $G' = G \cup \{e\}$ be the graph obtained by adding shortcut $e = (w_1, w_2)$ to G. Suppose that $d_G(u, w_2) < d_G(u, w_1) + d(w_1, w_2)$. Then no shortest path in G' from u traverses e in the direction $w_1 \to w_2$. Let $x = d_G(u, w_2) + d(w_2, w_1)$. Since we have not fixed w_2, we can regard x as a non-negative variable depending on w_2.

Let us analyze how $\delta_{G'}(u, v)$ changes as a function of x. For $x = 0$, if $x + d_G(w_1, v) > d_G(u, v)$ then a shortest path between u and v in G' cannot traverse e in direction $w_2 \to w_1$. By the above, a shortest path between u and v in G is also a shortest path between u and v in G'. This will also hold when x increases so $\delta_{G'}(u, v)$ is just a constant function of x.

Now, suppose $x + d_G(w_1, v) \leqslant d_G(u, v)$ for $x = 0$. Then there will be a shortest path from u to v in G' with a subpath traversing e in direction $w_2 \to w_1$ (by the inequality $d_G(u, w_2) < d_G(u, w_1) + d(w_1, w_2)$ above). As x increases it gets more and more expensive to use this subpath and thus $\delta_{G'}(u, v)$ increases; the increase in $\delta_{G'}(u, v)$ is a linear function of the increase in x. Eventually, $x + d_G(w_1, v) \geqslant d_G(u, v)$ and it will be cheaper to use a shortest path from u to v in the original graph G so $\delta_{G'}(u, v)$ will no longer increase but stay constant (as a function of x).

To be more precise, define constants $a = 1/d(u, v)$, $b = d_G(w_1, v)/d(u, v)$, and $c = \delta_G(u, v)$. Then $x \geqslant (c - b)/a$ is equivalent to the inequality $x + d_G(w_1, v) \geqslant d_G(u, v)$ above, giving

$$\delta_{G'}(u, v) = \min\{c, ax + b\} = \begin{cases} c & \text{if } x \geqslant \frac{c-b}{a}, \\ ax + b & \text{if } x \leqslant \frac{c-b}{a}. \end{cases}$$

Hence, the dilation between u and v in G' may be expressed as a piecewise linear function $\delta(x)$ of the length $x \geqslant 0$ of a shortest path among those paths in G' from u to w_1 having e as their last edge.

We refer to the graph of $\delta(x)$ as a *staircase step*, see Fig. 1(a). Assuming $(c - b)/a > 0$, the part of the graph on interval $[0, (c - b)/a]$ is a line segment with slope a, which we refer to as the *left leg* of $\delta(x)$.

If $(c - b)/a \leqslant 0$, we define the left leg of $\delta(x)$ to be the degenerate line segment with slope a starting and ending in the point $(0, \delta(0))$.

The part of the graph on interval $[\max\{0, (c - b)/a\}, \infty)$ is a horizontal halfline, called the *right leg* of $\delta(x)$.

We define the *slope* of $\delta(x)$ to be the slope a of its left leg.

The left and right leg of $\delta(x)$ meet in a single point. We refer to this point as the *tip* of $\delta(x)$.

Now, suppose we fix only u and w_1. For each $v \in V \setminus \{u\}$, we obtain a staircase step expressing the dilation between u and v in G'. We define $s_{(u,w_1)}$ to be the *staircase function* representing the upper envelope of the union of all these staircase steps as a function of $x \geqslant 0$, see Fig. 1(b). Note that this function is piecewise linear and non-decreasing.

4. The algorithm and its correctness

In this section, we present our algorithm and prove its correctness.

Initially, $d_G(u,v)$ is computed and stored for each $(u,v) \in V \times V$, and a table T with an entry for each ordered pair of vertices of G is initialized.

Let (w_1,w_2) be a pair of distinct vertices of V. At termination, entry $T(w_1,w_2)$ will hold the maximum dilation in $G \cup \{(w_1,w_2)\}$ over a certain set of vertex pairs. Similarly, entry $T(w_2,w_1)$ will hold the maximum dilation in $G \cup \{(w_1,w_2)\}$ over another set of vertex pairs. As we shall see, the union of these two sets cover all pairs of distinct vertices, implying that

$$\max\{T(w_1,w_2), T(w_2,w_1)\} = \delta_{G\cup\{(w_1,w_2)\}}. \tag{1}$$

A subsequent step may update T in $\Theta(n^2)$ time such that $T(w_1,w_2) = \delta_{G\cup\{(w_1,w_2)\}}$ for all $w_1 \neq w_2$. A best shortcut is then a pair (w_1,w_2) minimizing $T(w_1,w_2)$.

The algorithm consists of a loop which iterates over all vertices of G. Let w_1 be the vertex in the current iteration. First, staircase functions $s_{(u,w_1)}$ are computed for each $u \in V$. Then for each vertex $w_2 \neq w_1$, entry (w_1,w_2) in T is set to

$$T(w_1,w_2) = \max\{s_{(u,w_1)}(x) \mid u \in V,\ d_G(u,w_2) < d_G(u,w_1) + d(w_1,w_2)\},$$

where $x = d_G(u,w_2) + d(w_2,w_1)$. This is well-defined since $u = w_2$ satisfies $d_G(u,w_2) < d_G(u,w_1) + d(w_1,w_2)$. Note that x and the inequality in the definition of $T(w_1,w_2)$ are the same as in the previous section.

The following theorem shows the correctness of our algorithm.

Theorem 1. *When the above algorithm terminates,* (1) *holds for each pair* (w_1,w_2) *of distinct vertices.*

Proof. Let (w_1,w_2) be any pair of distinct vertices of G and let $G' = G \cup \{(w_1,w_2)\}$. For any $u \in V$ for which $d_G(u,w_2) < d_G(u,w_1) + d(w_1,w_2)$ holds,

$$s_{(u,w_1)}\big(d_G(u,w_2) + d(w_2,w_1)\big) = \max_{v\in V\setminus\{u\}} \delta_{G'}(u,v).$$

Similarly, for any $u \in V$ for which $d_G(u,w_1) < d_G(u,w_2) + d(w_2,w_1)$ holds,

$$s_{(u,w_2)}\big(d_G(u,w_1) + d(w_1,w_2)\big) = \max_{v\in V\setminus\{u\}} \delta_{G'}(u,v).$$

Furthermore, for any $u \in V$, either $d_G(u,w_2) < d_G(u,w_1) + d(w_1,w_2)$ or $d_G(u,w_1) < d_G(u,w_2) + d(w_2,w_1)$ for otherwise we get the contradiction

$$\begin{aligned} d_G(u,w_2) + d(w_2,w_1) &\leqslant d_G(u,w_1) \\ &\leqslant d_G(u,w_2) - d(w_1,w_2) \\ &< d_G(u,w_2) + d(w_2,w_1), \end{aligned}$$

where the strict inequality follows from the assumption that $w_1 \neq w_2$. Thus, at termination,

$$\max\{T(w_1,w_2), T(w_2,w_1)\} = \max_{u,v\in V,\, u\neq v} \delta_{G'}(u,v) = \delta_{G'},$$

as requested. □

5. Running time and space requirement

In this section, we show that the algorithm of the previous section has $O(n^3 \log n)$ running time and $O(n^2)$ space requirement. We will need the following lemma.

Lemma 1. *Given vertices u and* w_1*, the graph of staircase function* $s_{(u,w_1)}$ *consists of* $O(n)$ *line segments and one halfline and can be computed in* $O(n\log n)$ *time when* $d_G(w_1,v)$ *and* $d_G(u,v)$ *are precomputed for all* $v \in V$*. Furthermore, when this graph is given,* $s_{(u,w_1)}(x)$ *can be computed in* $O(\log n)$ *time for any* $x \geqslant 0$.

Proof. Note that when $d_G(w_1,v)$ and $d_G(u,v)$ are precomputed for all vertices v, each staircase step may be computed in constant time.

We represent the graph of $s_{(u,w_1)}$ as a polygonal chain P. To construct P, we start by computing each staircase step and the tip with maximum x-coordinate, say $x_{\max}$. The upper envelope of P to the right of $x_{\max}$ is the upper envelope of $O(n)$ horizontal halflines and may be computed in $O(n)$ time. The upper envelope of P on interval $[0, x_{\max}]$ is the upper envelope of $O(n)$ line segments. We use the algorithm of Hershberger [5] to compute this upper envelope in $O(n \log n)$ time. It follows that P may be constructed in $O(n \log n)$ time.

Clearly, P consists of line segments and exactly one halfline. We need to show that the number of line segments is $O(n)$.

Consider constructing P by iteratively adding staircase steps in non-decreasing order of slope. Let P_i be the upper envelope of the first i staircase steps.

Upper envelope P_1 consists of exactly one line segment (and one halfline). For $i > 1$, the left leg of the ith staircase step s_i intersects P_{i-1} at most once due to the order of staircase steps. Since the right leg is horizontal, it cannot intersect P_{i-1} more than once. Hence, P_i has at most two more line segments than P_{i-1}.

One fine point: a degeneracy may occur if the left leg of s_i overlaps with a line segment of P_{i-1}. It is easy to see that in this case, P_i cannot contain more line segments than P_{i-1}, again due to the order of the staircase steps.

Since there are $O(n)$ staircase steps, the above shows that P consists of $O(n)$ line segments.

Since $s_{(u,w_1)}$ is a non-decreasing function of x, we may apply a binary search in P to compute $s_{(u,w_1)}(x)$ for any $x \geqslant 0$. Since P consists of $O(n)$ line segments, this takes $O(\log n)$ time. $\square$

We are now ready for the main result of this section.

Theorem 2. *The algorithm described in Section* 4 *has* $O(n^3 \log n)$ *running time and* $O(n^2)$ *space requirement.*

Proof. To prove the time bound, first observe that computing all-pairs shortest paths takes $O(n^3)$ time with the Floyd–Warshall algorithm [4] (faster algorithms exist, see e.g. [1], but they will not improve the asymptotic running time of our algorithm).

Furthermore, the graph of each staircase function is computed exactly once throughout the course of the algorithm. Hence, by Lemma 1, the total time spent on computing these functions is $O(n^3 \log n)$. Once the staircase functions have been computed, computing an entry of T takes $O(n \log n)$ time by Lemma 1. Since T has n^2 entries, computing T takes $O(n^3 \log n)$ time. When all entries in T have been computed, finding the best shortcut takes $O(n^2)$ time. Hence, the total running time of the algorithm is $O(n^3 \log n)$.

Space requirement is bounded by that of the Floyd–Warshall algorithm and the space for storing the staircase functions, the shortest path lengths, and the table T. The Floyd–Warshall algorithm requires $\Theta(n^2)$ space. Clearly, T and the shortest path lengths can be stored using a total of $\Theta(n^2)$ space. In each iteration of the algorithm, we only store n staircase functions. By Lemma 1, they take up a total of $O(n^2)$ space. $\square$

6. Disconnected graph

Recall our assumption that G is connected. In this section, we show that some simple modifications of our algorithm allow us to handle the case where G is disconnected without affecting the worst-case running time and space requirement of the algorithm.

Note that if G consists of more than two connected components, G has infinite dilation and no single edge can be added to G to reduce the dilation, making the problem we consider trivial. Since there are efficient algorithms for determining the connected components of a graph, we may therefore restrict our attention to the case where G consists of exactly two connected components and assume that these two components have been computed.

So let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be the subgraphs defining the two connected components of G. For all shortcuts $(w_1, w_2) \in V_1 \times V_1 \cup V_2 \times V_2$, we set $T(w_1, w_2) = \infty$ since they leave the graph disconnected and hence leave the dilation of the graph unchanged.

As for the other entries in T, consider a pair of vertices (w_1, w_2) in $V_1 \times V_2$ and let $G' = G \cup \{(w_1, w_2)\}$. Let $u \neq v$ be two vertices of V. If $u, v \in V_1$ or $u, v \in V_2$ then clearly $\delta_{G'}(u, v) = \delta_G(u, v)$.

Now assume that $v \in V_1$ and $u \in V_2$. Then

$$\delta_{G'}(u, v) = ax + b,$$

where $x = d_G(u, w_2) + d(w_2, w_1)$, $a = 1/d(u, v)$, and $b = d_G(w_1, v)/d(u, v)$. Comparing this with the results of Section 3, we see that we in effect obtain staircase steps with no right leg. We let $s_{(u,w_1)}$ denote the staircase function representing the upper envelope of the staircase steps obtained from each $v \in V_1$ as a function of x.

To determine all entries $T(w_1, w_2)$ of T where $(w_1, w_2) \in V_1 \times V_2$, we make the following small changes to the algorithm of Section 4. The loop only iterates over vertices $w_1 \in V_1$. Furthermore, we only compute staircase functions $s_{(u,w_1)}$ for $u \in V_2$ and we set

$$T(w_1, w_2) = \max\bigl\{\delta_{G_1}, \delta_{G_2}, \ \max\bigl\{s_{(u,w_1)}(x) \bigm| u \in V_2\bigr\}\bigr\}$$

for each $w_2 \in V_2$ with x defined as above.

By the above it follows that, at termination, the modified algorithm satisfies

$$\max\bigl\{T(w_1, w_2), T(w_2, w_1)\bigr\} = \delta_{G \cup \{(w_1, w_2)\}}$$

for all distinct pairs of vertices w_1 and w_2 in G.

Computing the graph of staircase function $s_{(u,w_1)}$ is done in $O(n\log n)$ time using the algorithm of Hershberger [5] (as in the proof of Lemma 1, we pick a maximum x-value in order to consider line segments instead of halflines. Pick, say, the largest x-value ever needed by the algorithm). The graph of $s_{(u,w_1)}$ has complexity $O(n)$ and when it is given, $s_{(u,w_1)}(x)$ can be computed in $O(\log n)$ time for any $x \geqslant 0$; the proof of these claims is similar to the proof of Lemma 1.

From the above and from the results of Section 5, it follows easily that all entries of T can be computed in $O(n^3\log n)$ time using $O(n^2)$ space when G is disconnected.

7. Concluding remarks

We presented an $O(n^3\log n)$ time and $O(n^2)$ space algorithm for the problem of computing the best shortcut of a graph $G = (V, E)$ with n vertices embedded in a metric space. This improves upon a previous bound of $O(n^4)$ time and $O(n^2)$ space [3]. Our algorithm in fact solves a harder problem, namely that of computing the dilation of $G \cup \{(u, v)\}$ for each pair of distinct vertices u and v.

Based on ideas of this paper, the open problem stated in [3] of whether there exists a linear space algorithm with $o(n^4)$ running time for finding the best shortcut is solved in [6]. We pose the following problems. Is our algorithm optimal in terms of running time? Is it possible to extend our results to the more general case of adding a constant number of edges to G?

Acknowledgements

I thank Pawel Winter, Martin Zachariasen, and the anonymous referees for their comments and remarks.

References

[1] T.M. Chan, More algorithms for all-pairs shortest paths in weighted graphs, in: Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing, 2007, pp. 590–598.

[2] D. Eppstein, Spanning trees and spanners, in: J.-R. Sack, J. Urrutia (Eds.), Handbook of Computational Geometry, Elsevier Science Publishers, Amsterdam, 2000, pp. 425–461.

[3] M. Farshi, P. Giannopoulos, J. Gudmundsson, Improving the stretch factor of a geometric network by edge augmentation, SIAM J. Comput. 38 (1) (2008) 226–240.

[4] R.W. Floyd, Algorithm 97 (Shortest Path), Comm. ACM 5 (6) (1962) 345.

[5] J. Hershberger, Finding the upper envelope of n line segments in $O(n\log n)$ time, Inform. Process. Lett. 33 (4) (1989) 169–174.

[6] J. Luo, C. Wulff-Nilsen, Computing best and worst shortcuts of graphs embedded in metric spaces, in: S.-H. Hong, H. Nagamochi, T. Fukunaga (Eds.), Proceedings of the 19th International Symposium on Algorithms and Computation, in: Lecture Notes in Computer Sience, vol. 5369, Springer-Verlag, Berlin, 2008, pp. 764–775.

[7] G. Narasimhan, M. Smid, Geometric Spanner Networks, Cambridge University Press, 2007.

[8] M. Smid, Closest point problems in computational geometry, in: J.-R. Sack, J. Urrutia (Eds.), Handbook of Computational Geometry, Elsevier Science Publishers, Amsterdam, 2000, pp. 877–935.