37-th Anuall Iranian Mathematics Conference
11-14 Shahrivar 1385, 2-6 September 2006 , pp. 1–4

On Algorithms for Computing the Diameter of a t-Spanner

Mohammad Farshi† and Joachim Gudmundsson
Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands. m.farshi@tue.nl
and
NICTA‡ , Sydney, Australia. Joachim.Gudmundsson@nicta.com.au
(Submitted May 2006)

A t-spanner of a point set V is a undirected weighted graph G such that for each pair of points $p, q \in V$, there exist a path between p and q in G of length at most t times the distance between p and q (t-path). The diameter (t-diameter) of a t-spanner is the smallest integer d such that for any pair of vertices, there is a t-path in the graph between them containing at most d edges. As far as we know there is no known algorithm on how to compute the diameter of a t-spanner. In this paper we will give some algorithms for computing the diameter of a t-spanner. The time complexity of the most efficient algorithm is $\mathcal{O}(dmn)$, where n is the number of vertices, m is the number of edges and d is the diameter of the input graph, and it requires $\mathcal{O}(n)$ space.

Keywords: diameter, t-spanner, algorithm.
AMS subject classifications: 05C85, 05C12, 68W01.

1 Introduction

Consider a set V of n points. A network on V can be modeled as an undirected graph G with vertex set V and an edge set E of size m where every edge $e = (u, v)$ has a weight $|uv|$. Let P be a path in G between u and v. The weight of P is defined as the sum of the weight of the edges of P. Let $t > 1$ be a real number. We say that G is a *t-spanner* for V, if for each pair of points $u, v \in V$, there exists a path in G of weight at most t times the distance between u and v. We call such a path a *t-path* between u and v.

Complete graphs represent ideal communication networks, but they are expensive to build; sparse spanners represent low-cost alternatives. Spanners find applications in robotics, network topology design, distributed systems, design of parallel machines, and many other areas and have been a subject of considerable research. Recently spanners found interesting practical applications in

† Supported by Ministry of Science, Research and Technology of I. R. Iran.
‡ National ICT Australia is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

areas such as metric space searching [4] and broadcasting in communication networks [2]. For wireless ad hoc networks it is often desirable to have small diameter since it determines the maximum number of times a message has to be transmitted in a network. If a graph has diameter d then it is said to be a d-hop network. The problem of constructing spanners has received considerable attention from a theoretical perspective, see the surveys [3, 5].

Definition 1.1 The diameter (t-diameter) of a t-spanner is the smallest integer d such that for any pair of vertices u and v in V, there is a t-path in the graph between u and v containing at most d edges.

The definition of the diameter of a t-spanner is slightly different from its definition in general graphs. In general graphs the diameter of a graph $G(V, E)$ is $D = \max_{x,y \in V} d(x, y)$, where $d(x, y)$ is the number of edges of a shortest path between x and y. Here the path length is the number of edges in the path. The graph diameter can be determined by computing all-pairs shortest path (APSP) and it appears that this is the only way to solve the diameter problem, see [1]. But for computing t-diameter, APSP does not help because a t-path is not necessarily the shortest path. In other words, t-diameter of a graph can be smaller than its graph diameter because we can choose between wider range of paths, t-paths, between pairs.

2 Main Results

Let $G(V, E)$ be a t-spanner. The problem at hand is to compute the diameter of G. As far as we know there is no known algorithm on how to compute the diameter of a t-spanner, below we present a dynamic programming approach for the problem.

Notation. For each p and q in V, let $\delta_k(p, q)$ be the shortest path between p and q with at most k edges. In the case when we have more than one such a path we consider the path with minimum number of edges. Also assume that $L_k[p, q]$ is the length of $\delta_k(p, q)$. If there is no such path, we set $L_k[p, q]$ to ∞.

So the diameter of G is the smallest integer k such that $L_k[p, q] \leq t \cdot |pq|$, for all points p and q in V. Obviously

$$L_k(p, q) = \min \left\{ L_{k-1}(p, q), \min_{\binom{i+j=k}{r \in V \setminus \{p,q\}}} \{L_i(p, r) + L_j(r, q)\} \right\}.$$

LEMMA 2.1 *Let $G = (V, E)$ be a graph, p and q are two vertices of G and $k \geq 2$ be an integer. If i' and j' be two arbitrary positive integer such that $i' + j' = k$ then*

$$\min_{\binom{i+j=k}{r \in V \setminus \{p,q\}}} \{L_i(p, r) + L_j(r, q)\} = \min_{r \in V \setminus \{p,q\}} \{L_{i'}(p, r) + L_{j'}(r, q)\}.$$

Proof Obviously the left side of the equality is at most equal to the right side. To prove the opposite inequality, assume i and j are two arbitrary positive integer such that $i + j = k$. Our claim is that for each summation $L_i(p, r) + L_j(r, q)$ there exist a summation like $L_{i'}(p, r') + L_{j'}(r', q)$ such that $L_i(p, r) + L_j(r, q) \geq L_{i'}(p, r') + L_{j'}(r', q)$.

Without loss of generality we assume $i' < i$. So $j' > j$ and for each pair of points (x, y), $L_{j'}(x, y) \leq L_j(x, y)$.

Case 1: $\delta_i(p, r)$ contains at most i' edges.

In this case we are done because $L_i(p, r) = L_{i'}(p, r)$ and $L_j(r, q) \geq L_{j'}(r, q)$.

Case 2: $\delta_i(p, r)$ contains more than i' edges.

Let r' be the i'th node in $\delta_i(p, r)$ starting from p. Now

$$\begin{aligned} L_i(p, r) + L_j(r, q) &= L_{i'}(p, r') + L_{i-i'}(r', r) + L_j(r, q) \\ &\geq L_{i'}(p, r') + L_{i-i'+j}(r', q) \\ &= L_{i'}(p, r') + L_{j'}(r', q). \end{aligned}$$

□

By replacing i' by $k - 1$ and j' by 1 we have following corollary.

Corollary 2.2 *Let $G = (V, E)$ be a graph and p and q are two vertices of G. For each integer $k \geq 2$,*

$$L_k[p, q] = \min\{L_{k-1}[p, q], \min_{r \in V \setminus \{p,q\}} \{L_{k-1}[p, r] + L_1[r, q]\}\}.$$

Now for computing the diameter we start with $k = 1$. Obviously $L_1[p, q] = |pq|$ for all edges (p, q) in the graph and $L_1[p, q] = \infty$ otherwise. We can compute $L_k[p, q]$ for each $k \geq 1$ inductively using Corollary 2.2.

Algorithm. Diameter(G(V,E),t)

```
for all pair (p,q) ∈ V^2
    if (p,q) ∈ E then L_1[p,q] := |pq| else L_1[p,q] := ∞
k := 2; flag:=true
while flag =true
    flag:=false
    for all (p,q) ∈ V^2
        L_k[p,q] := min{L_{k-1}[p,q], min_{r∈V\{p,q}}{L_{k-1}[p,r] + L_1[r,q]}}
        if L_k[p,q] > t · |pq| then flag:=true
    k := k + 1
return k − 1
```

The following corollary follows immediately from the algorithm.

Corollary 2.3 *The diameter of a t-spanner G with n vertices can be computed in $\mathcal{O}(dn^3)$ time using $\mathcal{O}(n^2)$ space, where d is the diameter of G.*

With some modifications we can reduce the running time to $\mathcal{O}(dmn)$ which is much faster for t-spanners with $\mathcal{O}(n)$ edges. In the line 7 of DIAMETER, we compute the minimum over all $r \in V \setminus \{p, q\}$, but if r is not adjacent to q, then $L_1[r, q] = \infty$ and obviously we do not need to check this r. In the modification, we only check the adjacent nodes of one of the end points of the path instead of checking all nodes.

The algorithm uses $\mathcal{O}(n^2)$ space since we use a matrix to save the path lengths. However, by processing one node at a time, we can replace the matrix by an array. For each point $p \in V$ compute the radius of G from p. The radius of G from a node p is defined as the minimum integer d_p such that for any node $q \in V$, there is a t-path between p and q in G which contains at most d_p edges. After all the points have been processed, the largest radius will be the diameter of G.

THEOREM 2.4 *The diameter of a t-spanner G with n nodes and m edges can be computed in $\mathcal{O}(dmn)$ time using $\mathcal{O}(n)$ space, where d is the diameter of G.*

2.1 *Another Strategy*

In the previous algorithm, we computed $L_k[p, q]$ for every pair of points p and q in V. This was done by computing $L_{k-1}[p, r] + L_1[r, q]$ for every point $r \in V$ adjacent to q. However, the shortest path between p and r of *at most* $k - 1$ edges, $\delta_{k-1}(p, r)$, may contains less than $k - 1$ edges. We here observe that we do not need to check the path in this case since this will give us the shortest path between p and q containing at most $k - 1$ edges – which already has been computed. Thus to compute L_k from L_{k-1} it suffices to process all pairs (p, q) such that $\delta_{k-1}(p, q)$ contains exactly $k - 1$ edges and look at them instead of checking all pairs. Let S_{k-1} be the set of all such a pairs. For each pair $(p, q) \in S_{k-1}$ and for all adjacent nodes of p, say r, we check if there is a shorter path between r and q via p. We do the same for all adjacent nodes of q, as well. Using an obvious upper bound for S_{k-1} will give us an algorithm with time complexity $\mathcal{O}(dmn)$, which probably would be even faster in practice. However it would be interesting if one could prove that there exist a better bound for the number of pairs in S_k.

Reference

[1] D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999.

[2] K. Alzoubi, X.-Y. Li, Y. Wang, P.-J. Wan, and O. Frieder. Geometric spanners for wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(4):408–421, 2003.

[3] D. Eppstein. Spanning trees and spanners. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 425–461. Elsevier Science Publishers, Amsterdam, 2000.

[4] G. Navarro and R. Paredes. Practical construction of metric t-spanners. In *Proc. 5th Workshop on Algorithm Engineering and Experiments*, pages 69–81. SIAM Press, 2003.

[5] M. Smid. Closest point problems in computational geometry. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 877–935. Elsevier Science Publishers, Amsterdam, 2000.