Discrete Applied Mathematics 236 (2018) 395–407

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Tree spanners of bounded degree graphs

Ioannis Papoutsakis

Kastelli Pediados, Heraklion, Crete, 700 06, Greece

ARTICLE INFO

Article history:
Received 23 March 2015
Received in revised form 24 October 2016
Accepted 20 October 2017
Available online 20 November 2017

Keywords:
Tree spanner
Distance
Spanning tree
Efficient graph algorithm
Bounded degree graph

ABSTRACT

A tree t-spanner of a graph G is a spanning tree of G such that the distance between pairs of vertices in the tree is at most t times their distance in G. Deciding tree t-spanner admissible graphs has been proved to be tractable for $t < 3$ and NP-complete for $t > 3$, while the complexity status of this problem is unresolved when $t = 3$. For every $t > 2$ and $b > 0$, an efficient dynamic programming algorithm to decide tree t-spanner admissibility of graphs with vertex degrees less than b is presented. Only for $t = 3$, the algorithm remains efficient, when graphs G with degrees less than $b \log|V(G)|$ are examined.

1. Introduction

A t-spanner of a graph G is a spanning subgraph of G, such that the distance between pairs of vertices in the t-spanner is at most t times their distance in G. Spanners, when they have a few edges, approximate the distances in the graph, while they are sparse. Spanners of a graph that are trees attain the minimum number of edges a spanner of the graph can have. There are applications of spanners in a variety of areas, such as distributed computing [2,27], communication networks [28,26], motion planning and robotics [1,9], phylogenetic analysis [3] and in embedding finite metric spaces in graphs approximately [30]. In [29] it is mentioned that spanners have applications in approximation algorithms for geometric spaces [19], various approximation algorithms [12] and solving diagonally dominant linear systems [31].

On one hand, in [4,8,7] an efficient algorithm to decide tree 2-spanner admissible graphs is presented, where a method to construct all the tree 2-spanners of a graph is also given. On the other hand, in [8,7] it is proven that for each $t \geq 4$ the problem to decide graphs that admit a tree t-spanner is an NP-complete problem. The complexity status of the tree 3-spanner problem is unresolved. In [13], for every t, an efficient algorithm to determine whether a planar graph with bounded face length admits a tree t-spanner is presented. In [14] the existence of an efficient (actually linear) algorithm for the tree spanner problem on bounded degree graphs is shown, using a theorem of Logic; while it is mentioned that: “It would be interesting to show that one could use tools that do not rely on Courcelle’s theorem or Bodlaender’s algorithm to speed up practical implementations”. In this article, for every t, an efficient dynamic programming algorithm to decide tree t-spanner admissibility of bounded degree graphs is presented (Theorem 1).

Tree t-spanners ($t \geq 3$) have been studied for various families of graphs. If a connected graph is a cograph or a split graph or the complement of a bipartite graph, then it admits a tree 3-spanner [7]. Also, all convex bipartite graphs have a tree 3-spanner, which can be constructed in linear time [32]. Efficient algorithms to recognize graphs that admit a tree 3-spanner have been developed for interval, permutation and regular bipartite graphs [17], planar graphs [13], directed path graphs [16], very strongly chordal graphs, 1-split graphs and chordal graphs of diameter at most 2 [6]. In [23] an efficient algorithm to decide if a graph admits a tree 3-spanner of diameter at most 5 is presented. Moreover, every strongly chordal graph admits a tree 4-spanner, which can be constructed in linear time [5]; note that, for each t, there is a connected chordal

E-mail address: papoutsakis.ioannis@alumni.utoronto.ca.

https://doi.org/10.1016/j.dam.2017.10.025
0166-218X/© 2017 Elsevier B.V. All rights reserved.

graph that does not admit any tree t-spanner. The tree t-spanner problem has been studied for small diameter chordal graphs [6], diametrically uniform graphs [18], and outerplanar graphs [20]. Approximation algorithms for the tree t-spanner problem are presented in [11,26], where as in [11] a new necessary condition for a graph to have a tree t-spanner in terms of decomposition is also presented.

There are NP-completeness results for the tree t-spanner problem for families of graphs. In [13], it is shown that it is NP-hard to determine the minimum t for which a planar graph admits a tree t-spanner. In [10], it is proved that, for every $t \geq 4$, the problem of finding a tree t-spanner is NP-complete on K_6-minor-free graphs. For any $t \geq 4$, the tree t-spanner problem is NP-complete on chordal graphs of diameter at most $t + 1$, when t is even, and of diameter at most $t + 2$, when t is odd [6]; note that this refers to the diameter of the graph not to the diameter of the spanner. In [24] it is shown that the problem to determine whether a graph admits a tree t-spanner of diameter at most $t + 1$ is tractable, when $t \leq 3$, while it is an NP-complete problem, when $t \geq 4$. This last result is used in [25] to hint at the difficulty to approximate the minimum t for which a graph admits a tree t-spanner.

The tree 3-spanner problem is very interesting, since its complexity status is unresolved. In [22] it is shown that only for $t = 3$ the union of any two tree t-spanners of any given graph may contain big induced cycles but never an odd induced cycle (other than a triangle); such unions are proved to be perfect graphs. The algorithm presented in this article is efficient only for $t \leq 3$, when graphs with maximum degree $O(\log n)$ are considered, where $n(G)$ is the number of vertices of each graph G (Section 5). The tree 3-spanner problem can be formulated as an integer programming optimization problem. Constraints for such a formulation appear in [22], providing certificates of tree 3-spanner inadmissibility for some graphs.

2. Definitions

In general, terminology of [33] is used. If G is a graph, then $V(G)$ is its *vertex set* and $E(G)$ its *edge set*. An *edge* between vertices $u, v \in G$ is denoted as uv. Also, $G \setminus \{uv\}$ is the graph that remains when edge uv is removed from G. Let v be a vertex of G, then $N_G(v)$ is the set of G neighbors of v, while $N_G[v]$ is $N_G(v) \cup \{v\}$; in this article, graphs do not have loop edges. The *degree* of a vertex v in G is the number of edges of G incident to v. Here, $\Delta(G)$ is the maximum degree over the vertices of G.

Let G and H be two graphs. Then, $G \setminus H$ is graph G without the vertices of H, i.e. $V(G \setminus H) = V(G) \setminus V(H)$ and $E(G \setminus H) = \{uv \in E(G) : u \notin V(H) \text{ and } v \notin V(H)\}$. The *union* of G and H, denoted as $G \cup H$, is the graph with vertex set $V(G) \cup V(H)$ and edge set $E(G) \cup E(H)$. Similarly, the *intersection* of G and H, denoted as $G \cap H$, is the graph with vertex set $V(G) \cap V(H)$ and edge set $E(G) \cap E(H)$. Additionally, $G[H]$ is the subgraph of G *induced* by the vertices of H, i.e. $G[H]$ contains all vertices in $V(G) \cap V(H)$ and all the edges of G between vertices in $V(G) \cap V(H)$. Note that the usual definition of induced subgraph refers to H being a subgraph of G.

The G distance between two vertices $u, v \in G$ is the length of a u, v shortest path in G, while it is infinity, when u and v are not connected with a path in G. The definition of a tree t-spanner follows.

Definition 1. A graph T is a t-**spanner** of a graph G if and only if T is a subgraph of G and, for every pair u and v of vertices of G, if u and v are at distance d from each other in G, then u and v are at distance at most $t \cdot d$ from each other in T. If T is also a tree, then T is a **tree** t-**spanner** of G.

Note that in order to check that a spanning tree of a graph G is a tree t-spanner of G, it suffices to examine pairs of vertices that are adjacent in G [7]. There is an additive version of a spanner as well [15,29], which is not studied in this article. In the algorithm and in the proofs, r-centers are frequently used.

Definition 2. Let r be an integer. Vertex v of a graph G is an r-**center** of G if and only if for all vertices u in G, the distance from v to u in G is less than or equal to r.

To refer to all the vertices near a central vertex, the notion of a sphere is used.

Definition 3. Let r be an integer and v a vertex of a graph G. Then, the subgraph of G induced by the vertices of G at distance less than or equal to r from v is the **sphere** of G with center v and radius r; it is denoted as $(v, r)_G$-**sphere**.

Obviously, v is an r-center of a graph G, if and only if the $(v, r)_G$-sphere is equal to G.

An algorithm that runs in polynomial time is called *efficient*.

3. Description of the algorithm

In [21], characterization of tree t-spanner admissible graphs in terms of decomposition states, generally speaking, that if a tree t-spanner admissible graph G does not have small diameter then it is the union of two tree t-spanner admissible graphs whose intersection is a small diameter subgraph of G (this result requires further definitions to be stated exactly and it is not used in the proofs of this article). So, it may be the case that, starting with small diameter subgraphs and adding on them partial solutions of the remaining graph, a tree t-spanner of the whole graph is built.

Algorithm `Find_Tree_spanner` in Table 1 has as input a graph G and an integer $t > 1$. Its output is a tree t-spanner of G or a message that G does not admit any tree t-spanner. Being a dynamic programming algorithm, it grows partial solutions into final solutions starting from small subtrees of G. Obviously, each such subtree must be a tree t-spanner of the subgraph