An Approximation Algorithm for the Tree t-Spanner Problem on Unweighted Graphs via Generalized Chordal Graphs

Feodor F. Dragan[1] and Ekkehard Köhler[2]

[1] Algorithmic Research Laboratory, Department of Computer Science, Kent State University, Kent, OH 44242, USA
dragan@cs.kent.edu

[2] Mathematisches Institut, Brandenburgische Technische Universität Cottbus, D-03013 Cottbus, Germany
ekoehler@math.tu-cottbus.de

Abstract. A spanning tree T of a graph G is called a *tree* t*-spanner* of G if the distance between every pair of vertices in T is at most t times their distance in G. In this paper, we present an algorithm which constructs for an n-vertex m-edge unweighted graph G: (1) a tree $(2\lfloor\log_2 n\rfloor)$-spanner in $O(m\log n)$ time, if G is a chordal graph; (2) a tree $(2\rho\lfloor\log_2 n\rfloor)$-spanner in $O(mn\log^2 n)$ time or a tree $(12\rho\lfloor\log_2 n\rfloor)$-spanner in $O(m\log n)$ time, if G is a graph admitting a Robertson-Seymour's tree-decomposition with bags of radius at most ρ in G; and (3) a tree $(2\lceil t/2\rceil\lfloor\log_2 n\rfloor)$-spanner in $O(mn\log^2 n)$ time or a tree $(6t\lfloor\log_2 n\rfloor)$-spanner in $O(m\log n)$ time, if G is an arbitrary graph admitting a tree t-spanner. For the latter result we use a new necessary condition for a graph to have a tree t-spanner: if a graph G has a tree t-spanner, then G admits a Robertson-Seymour's tree-decomposition with bags of radius at most $\lceil t/2\rceil$ in G.

1 Introduction

Given a connected graph G and a spanning tree T of G, we say that T is a *tree* t*-spanner* of G if the distance between every pair of vertices in T is at most t times their distance in G. The parameter t is called the *stretch* (or *stretch factor*) of T. The TREE t-SPANNER problem asks, given a graph G and a positive number t, whether G admits a tree t-spanner. Note that the problem of finding a tree t-spanner of G minimizing t is known in literature also as the Minimum Max-Stretch spanning Tree problem (see, e.g., [14] and literature cited therein). This paper concerns the TREE t-SPANNER problem on unweighted graphs. The problem is known to be NP-complete even for planar graphs and chordal graphs (see [5,8,15]), and the paper presents an efficient algorithm which produces a tree t-spanner with $t \le 2\log_2 n$ for every n-vertex chordal graph and a tree $(2\lceil t/2\rceil\lfloor\log_2 n\rfloor)$-spanner for an arbitrary n-vertex graph admitting a tree t-spanner. To obtain the latter result, we show that every graph having a tree t-spanner admits a Robertson-Seymour's tree-decomposition with bags of radius at most $\lceil t/2\rceil$ in G. This tree-decomposition is a generalization of the well-known

L.A. Goldberg et al. (Eds.): APPROX/RANDOM 2011, LNCS 6845, pp. 171–183, 2011.

notion of a clique-tree of a chordal graph, and allows us to extend our algorithm developed for chordal graphs to arbitrary graphs admitting tree t-spanners.

There are many applications of tree spanners in various areas. We refer to the survey paper of Peleg [21] for an overview on spanners and their applications.

Related work. Substantial work has been done on the TREE t-SPANNER problem on unweighted graphs. Cai and Corneil [8] have shown that, for a given graph G, the problem to decide whether G has a tree t-spanner is NP-complete for any fixed $t \geq 4$ and is linear time solvable for $t = 1, 2$ (the status of the case $t = 3$ is open for general graphs)[1]. The NP-completeness result was further strengthened in [5] and [6], where Branstädt et al. showed that the problem remains NP-complete even for the class of chordal graphs (i.e., for graphs where each induced cycle has length 3) and every fixed $t \geq 4$, and for the class of chordal bipartite graphs (i.e., for bipartite graphs where each induced cycle has length 4) and every fixed $t \geq 5$.

The TREE t-SPANNER problem on planar graphs was studied in [15,23]. In [23], Peleg and Tendler presented a polynomial time algorithm for the minimum value of t for the TREE t-SPANNER problem on outerplanar graphs. In [15], Fekete and Kremer proved that the TREE t-SPANNER problem on planar graphs is NP-complete (when t is part of the input) and polynomial time solvable for $t = 3$. They also gave a polynomial time algorithm that for every fixed t decides for planar graphs with bounded face length whether there is a tree t-spanner. For fixed $t \geq 4$, the complexity of the TREE t-SPANNER problem on arbitrary planar graphs was left as an open problem in [15]. This open problem was recently resolved in [12], where it was shown that the TREE t-SPANNER problem is linear time solvable for every fixed constant t on the class of apex-minor-free graphs which includes all planar graphs and all graphs of bounded genus.

An $O(\log n)$-approximation algorithm for the minimum value of t for the TREE t-SPANNER problem is due to Emek and Peleg [14], and until recently that was the only $O(\log n)$-approximation algorithm available for the problem. Let G be an n-vertex m-edge unweighted graph and t^* be the minimum value such that a tree t^*-spanner exists for G. Emek and Peleg gave an algorithm which produces for every G a tree $(6t^* \lceil \log_2 n \rceil)$-spanner in $O(mn \log^2 n)$ time. Furthermore, they established that unless P = NP, the problem cannot be approximated additively by any $o(n)$ term. Hardness of approximation is established also in [19], where it was shown that approximating the minimum value of t for the TREE t-SPANNER problem within factor better than 2 is NP-hard (see also [22] for an earlier result). Recently, another logarithmic approximation algorithm for the TREE t-SPANNER problem was announced in [3], but authors did not provide any details. A number of papers have studied the related but easier problem of finding a spanning tree with good *average* stretch factor (see [1,2,13] and papers cited therein).

Our contribution. In this paper, we present a new algorithm which constructs for an n-vertex m-edge unweighted graph G: (1) a tree $(2\lfloor \log_2 n \rfloor)$-spanner in $O(m \log n)$ time, if G is a chordal graph; (2) a tree $(2\rho\lfloor \log_2 n \rfloor)$-spanner in

[1] When G is an unweighted graph, t can be assumed to be an integer.

$O(mn\log^2 n)$ time or a tree $(12\rho\lfloor\log_2 n\rfloor)$-spanner in $O(m\log n)$ time, if G is a graph admitting a Robertson-Seymour's tree-decomposition with bags of radius at most ρ in G; and (3) a tree $(2\lceil t/2\rceil\lfloor\log_2 n\rfloor)$-spanner in $O(mn\log^2 n)$ time or a tree $(6t\lfloor\log_2 n\rfloor)$-spanner in $O(m\log n)$ time, if G is an arbitrary graph admitting a tree t-spanner. For the latter result we employ a new necessary condition for a graph to have a tree t-spanner: if a graph G has a tree t-spanner, then G admits a Robertson-Seymour's tree-decomposition with bags of radius at most $\lceil t/2\rceil$ and diameter at most t in G. The algorithm needs to know neither an appropriate Robertson-Seymour's tree-decomposition of G nor the true value of t. It works directly on an input graph G.

A high-level description of our method is similar to that of [14], although the details are very different. We find a "small radius" balanced disk-separator of a graph $G=(V,E)$, that is, a disk $D_r(v,G)$ of radius r and centered at vertex v such that removal of vertices of $D_r(v,G)$ from G leaves no connected component with more that $n/2$ vertices. We recursively build a spanning tree for each graph formed by a connected component G_i of $G[V\setminus D_r(v,G)]$ with one additional vertex v added to G_i to represent the disk $D_r(v,G)$ and its adjacency relation to G_i. The spanning trees generated by recursive invocations of the algorithm on each such graph are glued together at vertex v and then the vertex v of the resulting tree is substituted with a single source shortest path spanning tree of $D_r(v,G)$ to produce a spanning tree T of G. Analysis of the algorithm relies on an observation that the number of edges added to the unique path between vertices x and y in T, where xy is an edge of G, on each of $\lfloor\log_2 n\rfloor$ recursive levels is at most $2r$.

Comparing with the algorithm of Emek and Peleg ([14]), one variant of our algorithm has the same approximation ratio but a better run-time, other variant has the same run-time but a better constant term in the approximation ratio. Our algorithm and its analysis, in our opinion, are conceptually simpler due to a new necessary condition for a graph to have a tree t-spanner.

2 Preliminaries

All graphs occurring in this paper are connected, finite, unweighted, undirected, loopless and without multiple edges. We call $G=(V,E)$ an *n-vertex m-edge graph* if $|V|=n$ and $|E|=m$. A *clique* is a set of pairwise adjacent vertices of G. By $G[S]$ we denote a subgraph of G induced by vertices of $S\subseteq V$. Let also $G\setminus S$ be the graph $G[V\setminus S]$ (which is not necessarily connected). A set $S\subseteq V$ is called a *separator* of G if the graph $G[V\setminus S]$ has more than one connected component, and S is called a *balanced separator* of G if each connected component of $G[V\setminus S]$ has at most $|V|/2$ vertices. A set $C\subseteq V$ is called a *balanced clique-separator* of G if C is both a clique and a balanced separator of G. For a vertex v of G, the sets $N_G(v)=\{w\in V : vw\in E\}$ and $N_G[v]=N_G(v)\cup\{v\}$ are called the *open neighborhood* and the *closed neighborhood* of v, respectively.

In a graph G the *length* of a path from a vertex v to a vertex u is the number of edges in the path. The *distance* $d_G(u,v)$ between vertices u and v is the length of a shortest path connecting u and v in G. The *diameter* in G of a set $S \subseteq V$ is $\max_{x,y \in S} d_G(x,y)$ and its *radius* in G is $\min_{x\in V} \max_{y\in S} d_G(x,y)$ (in some papers they are called the *weak diameter* and the *weak radius* to indicate that the distances are measured in G not in $G[S]$). The *disk* of G of radius k centered at vertex v is the set of all vertices at distance at most k to v: $D_k(v,G) = \{w \in V : d_G(v,w) \leq k\}$. A disk $D_k(v,G)$ is called a *balanced disk-separator* of G if the set $D_k(v,G)$ is a balanced separator of G.

Let G be a connected graph and t be a positive number. A spanning tree T of G is called a *tree t-spanner* of G if the distance between every pair of vertices in T is at most t times their distance in G, i.e., $d_T(x,y) \leq t\, d_G(x,y)$ for every pair of vertices x and y of G. It is easy to see that the tree t-spanners can equivalently be defined as follows.

Proposition 1. *Let G be a connected graph and t be a positive number. A spanning tree T of G is a tree t-spanner of G if and only if for every edge xy of G, $d_T(x,y) \leq t$ holds.*

This proposition implies that the stretch of a spanning tree of a graph G is always obtained on a pair of vertices that form an edge in G. Consequently, throughout this paper t can be considered as an integer which is greater than 1.

3 Tree Spanners of Chordal Graphs

As we have mentioned earlier the TREE t-SPANNER problem is NP-complete for every $t \geq 4$ even for the class of chordal graphs [5]. Recall that a graph G is called *chordal* if each induced cycle of G has length 3. In this section, we show that every chordal graph with n vertices admits a tree t-spanner with $t \leq 2\log_2 n$. In the full version of the paper (see [26]), we show also that there are chordal graphs for which any tree t-spanner has to have $t \geq \log_2 \frac{n}{3} + 2$. All proofs omitted in this extended abstract can also be found in the full version.

We start with three lemmas that are crucial to our method. Let $G = (V,E)$ be an arbitrary connected graph with a clique-separator C, i.e., there is a clique C in G such that the removal of the vertices of C from G results in a graph with more than one connected component. Let $G_1,\ldots,G_k$ be those connected components of $G[V \setminus C]$. Denote by $S_i := \{x \in V(G_i) : d_G(x,C) = 1\}$ the neighborhood of C with respect to G_i. Let also G_i^+ be the graph obtained from component G_i by adding a vertex c_i (*representative* of C) and making it adjacent to all vertices of S_i, i.e., for a vertex $x \in V(G_i)$, $c_i x \in E(G_i^+)$ if and only if there is a vertex $x_C \in C$ with $xx_C \in E(G)$ (see Fig. 1). Clearly, given a connected m-edge graph G and a clique-separator C of G, the graphs $G_1^+,\ldots,G_k^+$ can be constructed in total time $O(m)$. Note also that the total number of edges in graphs $G_1^+,\ldots,G_k^+$ does not exceed the number of edges in G.

Denote by $G_{/e}$ the graph obtained from G by contracting its edge e. Recall that edge e contraction is an operation which removes e from G while simultaneously merging together the two vertices e previously connected. If a contraction results in multiple edges, we delete duplicates of an edge to stay within the

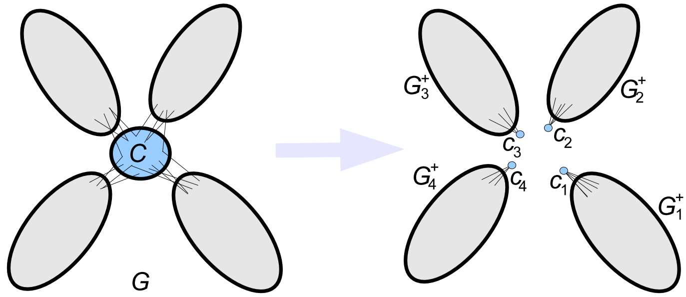

Fig. 1. A graph G with a clique-separator C and the corresponding graphs $G_1^+, \dots, G_4^+$ obtained from G

class of simple graphs. The operation may be performed on a set of edges by contracting each edge (in any order).

Lemma 1. *If a graph G is chordal then $G_{/e}$ is chordal as well, for any edge $e \in E(G)$. Consequently, if a graph G is chordal then G_i^+ is chordal as well, for each $i = 1, \dots, k$.*

Let T_i $(i = 1, \dots, k)$ be a spanning tree of G_i^+ such that for any edge $xy \in E(G_i^+)$, $d_{T_i}(x, y) \le \alpha$ holds, where α is some positive integer independent of i. We can form a spanning tree T of G from trees $T_1, \dots, T_k$ and the vertices of the clique C in the following way. For each $i = 1, \dots, k$, rename vertex c_i in T_i to c. Glue trees $T_1, \dots, T_k$ together at vertex c obtaining a tree T' (see Fig. 2). For the original clique C of G, pick an arbitrary vertex r_C of C and create a spanning star ST_C for C centered at r_C. Substitute vertex c in T' by that star ST_C. For each former edge xc of T', create an edge xx_C in T where x_C is a vertex of C adjacent to x in G. We can show that for any edge $xy \in E(G)$, $d_T(x, y) \le \alpha + 2$ holds. Evidently, the tree T of G can be constructed from trees $T_1, \dots, T_k$ and the vertices of the clique C in $O(m)$ time.

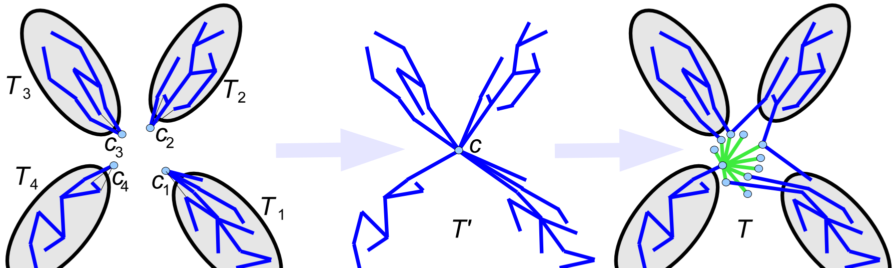

Fig. 2. Spanning trees $T_1, \dots, T_4$ of $G_1^+, \dots, G_4^+$, resulting tree T', and a corresponding spanning tree T of G

Lemma 2. *Let G be an arbitrary graph with a clique-separator C and $G_1^+,\dots,G_k^+$ be the graphs obtained from G as described above. Let also T_i ($i \in \{1,\dots,k\}$) be a spanning tree of the graph G_i^+, and T be a spanning tree of G constructed from $T_1,\dots,T_k$ and the clique C as described above. Assume also that there is a positive integer α such that, for each $i \in \{1,\dots,k\}$ and every edge $xy \in E(G_i^+)$, $d_{T_i}(x,y) \le \alpha$ holds. Then, for every edge $xy \in E(G)$, $d_T(x,y) \le \alpha + 2$ holds.*

Proof. Consider an arbitrary edge xy of G. If both x and y belong to C, then evidently $d_T(x,y) \le 2 < \alpha + 2$. Assume now that xy is an edge of G_i for some $i \in \{1,\dots,k\}$. Then, xy is an edge of G_i^+ and therefore $d_{T_i}(x,y) \le \alpha$. If the path P of T_i connecting x and y does not contain vertex c_i, then $d_T(x,y) = d_{T_i}(x,y) \le \alpha$ must hold. If c_i is between x and y in T_i (i.e., $c_i \in P$), then the distance in T between x and y is at most $d_{T_i}(x,y) + 2$ (the path of T between x and y is obtained from P by substituting the vertex $c = c_i$ by a path of star ST_C with at most 2 edges). It remains to consider the case when $x \in C$ and $y \in V(G_i)$. By construction of G_i^+, there must exist an edge $c_i y$ in G_i^+. We have $d_{T_i}(c_i,y) \le \alpha$. Let z be the neighbor of c_i in the path of T_i connecting vertices y and c_i ($y = z$ is possible). Evidently, $z \in V(G_i)$. By construction, in T we must have an edge zz_c where z_C is a vertex of C adjacent to z in G. Vertices x and z_C both are in C and the distance in T between them is at most 2. Thus, $d_T(x,y) \le d_T(z_C,y) + 2 = d_{T_i}(c_i,y) + 2 \le \alpha + 2$. □

The third important ingredient to our method is the famous chordal balanced separator result of Gilbert, Rose, and Edenbrandt [18].

Lemma 3. [18] *Every connected chordal graph G with n vertices and m edges contains a balanced clique-separator which can be found in $O(m)$ time.*

Now let $G = (V,E)$ be a connected chordal graph with n vertices and m edges. Using Lemma 1 and Lemma 3, we can build a (rooted) *hierarchical-tree* $\mathcal{H}(G)$ for G, which can be constructed as follows. If G is a connected graph with at most 5 vertices or is a clique of size greater than 5, then $\mathcal{H}(G)$ is a one node tree with root node (G, nil). Otherwise, find a balanced clique-separator C of G (which exists by Lemma 3 and which can be found in $O(m)$ time) and construct the associated graphs $G_1^+,\dots,G_k^+$. For each graph G_i^+, $i \in \{1,\dots,k\}$, which is chordal by Lemma 1, construct a hierarchical-tree $\mathcal{H}(G_i^+)$ recursively and build $\mathcal{H}(G)$ by taking the pair (G,C) to be the root and connecting the root of each tree $\mathcal{H}(G_i^+)$ as a child of (G,C). The depth of this tree $\mathcal{H}(G)$ is the smallest integer k such that $\frac{n}{2^k} + \frac{1}{2^{k-1}} + \ldots + \frac{1}{2} + 1 \le 5$, that is, the depth is at most $\log_2 n - 1$.

To build a tree t-spanner T of G, we use the hierarchical-tree $\mathcal{H}(G)$ and a bottom-up construction. We know from Proposition 1 that a spanning tree T is a tree t-spanner of a graph G if and only if for any edge xy of G, $d_T(x,y) \le t$ holds. For each leaf (L,nil) of $\mathcal{H}(G)$ (we know that graph L is a clique or a connected chordal graph with at most 5 vertices), we construct a tree 2-spanner T_L of L. It is easy to see that L admits such a tree 2-spanner. Hence, for any edge xy of L, we have $d_{T_L}(x,y) \le 2$. Consider now an inner node (H,K) of

$\mathcal{H}(G)$, and assume that all its children $H_1^+, \ldots, H_l^+$ in $\mathcal{H}(G)$ have tree α-spanners $T_1, \ldots, T_l$ for some positive integer α. Then, a tree $(\alpha+2)$-spanner of H can be constructed from $T_1, \ldots, T_l$ and clique K of H as described above (see Lemma 2 and paragraph before it). Since the depth of the hierarchical-tree $\mathcal{H}(G)$ is at most $\log_2 n - 1$ and all leaves of $\mathcal{H}(G)$ admit tree 2-spanners, applying Lemma 2 repeatedly, we will move from leaves to the root of $\mathcal{H}(G)$ and get a tree t-spanner T of G with t being no more than $2\log_2 n$.

It is also easy to see that, given a chordal graph G with n vertices and m edges, a hierarchical-tree $\mathcal{H}(G)$ as well as a tree t-spanner T of G with $t \leq 2\log_2 n$ can be constructed in $O(m \log n)$ total time since there are at most $O(\log n)$ levels in $\mathcal{H}(G)$ and one needs to do at most $O(m)$ operations per level. Thus, we have the following result for the class of chordal graphs.

Theorem 1. *Any connected chordal graph G with n vertices and m edges admits a tree $(2\lfloor \log_2 n \rfloor)$-spanner constructible in $O(m \log n)$ time.*

4 Tree Spanners of Generalized Chordal Graphs

It is known that the class of chordal graphs can be characterized in terms of existence of so-called *clique-trees*. Let $\mathcal{C}(G)$ denote the family of maximal (by inclusion) cliques of a graph G. A *clique-tree* $\mathcal{CT}(G)$ of G has the maximal cliques of G as its nodes, and for every vertex v of G, the maximal cliques containing v form a subtree of $\mathcal{CT}(G)$.

Theorem 2. **[7,17]** *A graph is chordal if and only if it has a clique-tree.*

In their work on graph minors [25], Robertson and Seymour introduced the notion of tree-decomposition which generalizes the notion of clique-tree. A *tree-decomposition* of a graph G is a tree $\mathcal{T}(G)$ whose nodes, called *bags*, are subsets of $V(G)$ such that: (1) $\bigcup_{X \in V(\mathcal{T}(G))} X = V(G)$, (2) for each edge $vw \in E(G)$, there is a bag $X \in V(\mathcal{T}(G))$ such that $v, w \in X$, and (3) for each $v \in V(G)$ the set of bags $\{X : X \in V(\mathcal{T}(G)), v \in X\}$ forms a subtree $\mathcal{T}_v(G)$ of $\mathcal{T}(G)$.

Tree-decompositions were used in defining at least two graph parameters. The *tree-width* of a graph G is defined as minimum of $\max_{X \in V(\mathcal{T}(G))} |X| - 1$ over all tree-decompositions $\mathcal{T}(G)$ of G and is denoted by $\mathsf{tw}(G)$ [25]. The *length* of a tree-decomposition $\mathcal{T}(G)$ of a graph G is $\max_{X \in V(\mathcal{T}(G))} \max_{u,v \in X} d_G(u, v)$, and the *tree-length* of G, denoted by $\mathsf{tl}(G)$, is the minimum of the length, over all tree-decompositions of G [11]. These two graph parameters are not related to each other. Interestingly, the tree-length of a graph can be approximated in polynomial time within a constant factor [11] whereas such an approximation factor is unknown for the tree-width.

For the purpose of this paper, we introduce yet another graph parameter based on the notion of tree-decomposition. It is very similar to the notion of tree-length but is more appropriate for our discussions, and moreover it will lead to a better constant in our approximation ratio presented in Section 5 for the TREE t-SPANNER problem on general graphs.

Definition 1. The *breadth* of a tree-decomposition $\mathcal{T}(G)$ of a graph G is the minimum integer k such that for every $X \in V(\mathcal{T}(G))$ there is a vertex $v_X \in V(G)$ with $X \subseteq D_k(v_X, G)$ (i.e., each bag X has radius at most k in G). Note that vertex v_X does not need to belong to X. The *tree-breadth* of G, denoted by $\mathsf{tb}(G)$, is the minimum of the breadth, over all tree-decompositions of G. We say that a family of graphs $\mathcal{G}$ is *of bounded tree-breadth,* if there is a constant c such that for each graph G from $\mathcal{G}$, $\mathsf{tb}(G) \leq c$.

Evidently, for any graph G, $1 \leq \mathsf{tb}(G) \leq \mathsf{tl}(G) \leq 2\mathsf{tb}(G)$ holds. Hence, if one parameter is bounded by a constant for a graph G then the other parameter is bounded for G as well.

In what follows, we will show that any graph G with tree-breadth $\mathsf{tb}(G) \leq \rho$ admits a tree $(2\rho\lfloor\log_2 n\rfloor)$-spanner, thus generalizing the result for chordal graphs of Section 3 (if G is chordal then $\mathsf{tl}(G) = \mathsf{tb}(G) = 1$). It is interesting to note that the TREE t-SPANNER problem is NP-complete for graphs of bounded tree-breadth (even for chordal graphs for every fixed $t > 3$; see [5]), while it is polynomial time solvable for all graphs of bounded tree-width (see [24]).

First we present a balanced separator result.

Lemma 4. *Every graph G with n vertices, m edges and with tree-breadth at most ρ contains a vertex v such that $D_\rho(v, G)$ is a balanced disk-separator of G.*

Proof. The proof of this lemma follows from *acyclic hypergraph* theory. First we review some necessary definitions and an important result characterizing acyclic hypergraphs. Recall that a *hypergraph* H is a pair $H = (V, \mathcal{E})$ where V is a set of vertices and $\mathcal{E}$ is a set of non-empty subsets of V called *hyperedges*. For these and other hypergraph notions see [4].

Let $H = (V, \mathcal{E})$ be a *hypergraph* with the vertex set V and the *hyperedge* set $\mathcal{E}$. For every vertex $v \in V$, let $\mathcal{E}(v) = \{e \in \mathcal{E} : v \in e\}$. The *2–section graph* $2SEC(H)$ of a hypergraph H has V as its vertex set and two distinct vertices are adjacent in $2SEC(H)$ if and only if they are contained in a common hyperedge of H. A hypergraph H is called *conformal* if every clique of $2SEC(H)$ is contained in a hyperedge $e \in \mathcal{E}$, and a hypergraph H is called *acyclic* if there is a tree T with node set $\mathcal{E}$ such that for all vertices $v \in V$, $\mathcal{E}(v)$ induces a subtree T_v of T. It is a well-known fact (see, e.g., [4]) that a hypergraph H is acyclic if and only if H is conformal and $2SEC(H)$ of H is a chordal graph.

Let now G be a graph with $\mathsf{tb}(G) = \rho$ and $\mathcal{T}(G)$ be its tree-decomposition of breadth ρ. Evidently, property (3) in the definition of tree-decomposition can be restated as follows: the hypergraph $H = (V(G), \{X : X \in V(\mathcal{T}(G))\})$ is an acyclic hypergraph. Since each edge of G is contained in at least one bag of $\mathcal{T}(G)$, the 2–section graph $G^* := 2SEC(H)$ of H is a chordal *supergraph* of the graph G (each edge of G is an edge of G^*, but G^* may have some extra edges between non-adjacent vertices of G contained in a common bag of $\mathcal{T}(G)$). By Lemma 3, the chordal graph G^* contains a balanced clique-separator $C \subseteq V(G)$. By conformality of H, C must be contained in a bag of $\mathcal{T}(G)$. Hence, there must exist a vertex $v \in V(G)$ with $C \subseteq D_\rho(v, G)$. As the removal of the vertices of C from G^* leaves no connected component in $G^*[V \setminus C]$ with more that $n/2$

vertices and since G^* is a supergraph of G, clearly, the removal of the vertices of $D_\rho(v, G)$ from G leaves no connected component in $G[V \setminus D_\rho(v, G)]$ with more that $n/2$ vertices. □

We do not need to know a tree-decomposition $\mathcal{T}(G)$ of breadth ρ to find such a balanced disk-separator $D_\rho(v, G)$ of G. For a given graph G and an integer ρ, checking whether G has a tree-decomposition of breadth ρ could be a hard problem. For example, while graphs with tree-length 1 (as they are exactly the chordal graphs) can be recognized in linear time, the problem of determining whether a given graph has tree-length at most λ is NP-complete for every fixed $\lambda > 1$ (see [20]). Instead, we can use the following result.

Proposition 2. *For an arbitrary graph G with n vertices and m edges a balanced disk-separator $D_r(v, G)$ with minimum r can be found in $O(nm)$ time.*

Now let $G = (V, E)$ be an arbitrary connected n-vertex m-edge graph with a disk-separator $D_r(v, G)$. As in the case of chordal graphs, let $G_1, \dots, G_k$ be the connected components of $G[V \setminus D_r(v, G)]$. Denote by $S_i := \{x \in V(G_i) : d_G(x, D_r(v, G)) = 1\}$ the neighborhood of $D_r(v, G)$ with respect to G_i. Let also G_i^+ be the graph obtained from component G_i by adding a vertex v_i (*representative* of $D_r(v, G)$) and making it adjacent to all vertices of S_i, i.e., for a vertex $x \in V(G_i)$, $v_i x \in E(G_i^+)$ if and only if there is a vertex $x_D \in D_r(v, G)$ with $x x_D \in E(G)$. Given graph G and its disk-separator $D_r(v, G)$, the graphs $G_1^+, \dots, G_k^+$ can be constructed in total time $O(m)$. Furthermore, the total number of edges in the graphs $G_1^+, \dots, G_k^+$ does not exceed the number of edges in G, and the total number of vertices in those graphs does not exceed the number of vertices in $G[V \setminus D_r(v, G)]$ plus k. Let again $G_{/e}$ be the graph obtained from G by contracting its edge e.

Lemma 5. *For any graph G and its edge e, $\mathtt{tb}(G) \leq \rho$ implies $\mathtt{tb}(G_{/e}) \leq \rho$. Consequently, for any graph G with $\mathtt{tb}(G) \leq \rho$, $\mathtt{tb}(G_i^+) \leq \rho$ holds for each i.*

As in Section 3, let T_i $(i = 1, \dots, k)$ be a spanning tree of G_i^+ such that for any edge $xy \in E(G_i^+)$, $d_{T_i}(x, y) \leq \alpha$ holds, where α is some positive integer independent of i. For the disk $D_r(v, G)$ of G, construct a shortest path tree SPT_D rooted at vertex v (and spanning all and only the vertices of the disk). We can form a spanning tree T of G from trees $T_1, \dots, T_k$ and SPT_D in the following way. For each $i = 1, \dots, k$, rename vertex v_i in T_i to v. Glue trees $T_1, \dots, T_k$ together at vertex v obtaining a tree T' (consult with Fig. 2). Substitute vertex v in T' by the tree SPT_D. For each former edge xv of T', create an edge $x x_D$ in T where x_D is a vertex of $D_r(v, G)$ adjacent to x in G. We can show that for any edge $xy \in E(G)$, $d_T(x, y) \leq \alpha + 2r$ holds. Evidently, the tree T of G can be constructed from trees $T_1, \dots, T_k$ and SPT_D in $O(m)$ time.

Lemma 6. *Let G be an arbitrary graph with a disk-separator $D_r(v, G)$ and $G_1^+, \dots, G_k^+$ be the graphs obtained from G as described above. Let also T_i $(i \in \{1, \dots, k\})$ be a spanning tree of the graph G_i^+, and T be a spanning tree of G*

constructed from $T_1, \ldots, T_k$ and a shortest path tree SPT_D of the disk $D_r(v, G)$ as described above. Assume also that there is a positive integer α such that, for each $i \in \{1, \ldots, k\}$ and every edge $xy \in E(G_i^+)$, $d_{T_i}(x, y) \leq \alpha$ holds. Then, for every edge $xy \in E(G)$, $d_T(x, y) \leq \alpha + 2r$ must hold.

Now we have all necessary ingredients to apply the technique used in Section 3 and show that each graph G admits a tree $(2\mathsf{tb}(G)\lfloor\log_2 n\rfloor)$-spanner.

Let $G = (V, E)$ be a connected n-vertex, m-edge graph and assume that $\mathsf{tb}(G) \leq \rho$. Lemma 4 guaranties that G has a balanced disk-separator $D_r(v, G)$ with $r \leq \rho$. Proposition 2 says that such a balanced disk-separator $D_r(v, G)$ of G can be found in $O(nm)$ time by an algorithm that works directly on graph G and does not require the construction of a tree-decomposition of G of breadth $\leq \rho$. Using this and Lemma 5, we can build as before a (rooted) *hierarchical-tree* $\mathcal{H}(G)$ for G. Only now, the leaves of $\mathcal{H}(G)$ are connected graphs with at most 9 vertices. It is not hard to see that any leaf of $\mathcal{H}(G)$ has a tree t-spanner with $t \leq 4\rho$. Furthermore, a simple analysis shows that the depth of this tree $\mathcal{H}(G)$ is at most $\log_2 n - 2$.

To build a tree t-spanner T of G, we again use the hierarchical-tree $\mathcal{H}(G)$ and a bottom-up construction. Each leaf (L, nil) of $\mathcal{H}(G)$ has a tree (4ρ)-spanner. A tree t-spanner with minimum t of such a small graph L can be computed directly. Consider now an inner node $(H, D_r(v, G))$ of $\mathcal{H}(G)$ (where $D_r(v, G)$ is a balanced disk-separator of H), and assume that all its children $H_1^+, \ldots, H_l^+$ in $\mathcal{H}(G)$ have tree α-spanners $T_1, \ldots, T_l$ for some positive integer α. Then, a tree $(\alpha + 2r)$-spanner of H can be constructed from $T_1, \ldots, T_l$ and a shortest path tree SPT_D of the disk $D_r(v, G)$ as described above (see Lemma 6 and paragraph before it). Since the depth of the hierarchical-tree $\mathcal{H}(G)$ is at most $\log_2 n - 2$ and all leaves of $\mathcal{H}(G)$ admit tree (4ρ)-spanners, applying Lemma 6 repeatedly, we move from leaves to the root of $\mathcal{H}(G)$ and get a tree t-spanner T of G with t being no more than $2\rho\log_2 n$. It is also easy to see that, given a graph G with n vertices and m edges, a hierarchical-tree $\mathcal{H}(G)$ as well as a tree t-spanner T of G with $t \leq 2\mathsf{tb}(G)\log_2 n$ can be constructed in $O(nm\log^2 n)$ total time. There are at most $O(\log n)$ levels in $\mathcal{H}(G)$, and one needs to do at most $O(nm\log n)$ operations per level since the total number of edges in the graphs of each level is at most m and the total number of vertices in those graphs can not exceed $O(n\log n)$.

Note that our algorithm does not need to know the value of $\mathsf{tb}(G)$, neither it needs to know any appropriate Robertson-Seymour's tree-decomposition of G. It works directly on an input graph. To indicate this, we say that the algorithm constructs an appropriate tree spanner *from scratch*.

Thus, we have the following results.

Theorem 3. *There is an algorithm that for an arbitrary connected graph G with n vertices and m edges constructs a tree $(2\mathsf{tb}(G)\lfloor\log_2 n\rfloor)$-spanner of G in $O(nm\log^2 n)$ total time.*

Corollary 1. *Any connected n-vertex, m-edge graph G with $\mathsf{tb}(G) \leq \rho$ admits a tree $(2\rho\lfloor\log_2 n\rfloor)$-spanner constructible in $O(nm\log^2 n)$ time from scratch.*

Corollary 2. *Any connected n-vertex, m-edge graph G with $\mathsf{tl}(G) \leq \lambda$ admits a tree $(2\lambda\lfloor\log_2 n\rfloor)$-spanner constructible in $O(nm\log^2 n)$ time from scratch.*

There is another natural generalization of chordal graphs. A graph G is called *k-chordal* if its largest induced cycle has length at most k. Chordal graphs are exactly 3-chordal graphs. It was shown in [16] that every k-chordal graph has tree-length at most $k/2$. Thus, we have one more corollary.

Corollary 3. *Any connected n-vertex, m-edge k-chordal graph G admits a tree $(2\lfloor k/2\rfloor\lfloor\log_2 n\rfloor)$-spanner constructible in $O(nm\log^2 n)$ time from scratch.*

5 Approximating Tree t-Spanners of General Graphs

In this section, we show that the results obtained for tree t-spanners of generalized chordal graphs lead to an approximation algorithm for the TREE t-SPANNER problem on general (unweighted) graphs. We show that every graph G admitting a tree t-spanner has tree-breadth at most $\lceil t/2\rceil$. From this and Theorem 3 it follows that there is an algorithm which produces for every n-vertex and m-edge graph G a tree $(2\lceil t/2\rceil\lfloor\log_2 n\rfloor)$-spanner in $O(nm\log^2 n)$ time, whenever G admits a tree t-spanner. The algorithm does not even need to know the true value of t.

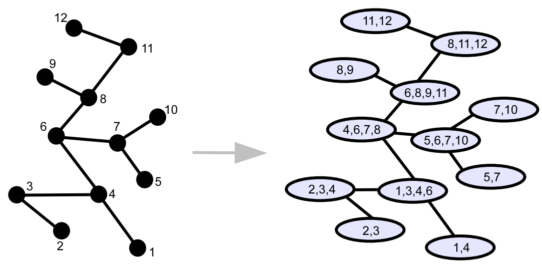

Fig. 3. From tree T to tree-decomposition $\mathcal{T}$ with $t = 2$

Lemma 7. *If a graph G admits a tree t-spanner then $\mathsf{tb}(G) \leq \lceil t/2\rceil$.*

Proof. Let T be a tree t-spanner of G. We can transform this tree T to a tree-decomposition $\mathcal{T}$ of G by expanding each vertex x in T to a bag X and putting all vertices of disk $D_{\lceil t/2\rceil}(x, T)$ into that bag (note that the disk here is considered in T; see Fig. 3 for an illustration). The edges of T and of $\mathcal{T}$ are identical: XY is an edge in $\mathcal{T}$ if and only if $xy \in E(T)$, where X is a bag that replaced vertex x in T and Y is a bag that replaced vertex y in T. Since $d_G(u,v) \leq d_T(u,v)$ for every pair of vertices u and v of G, we know that every bag $X := D_{\lceil t/2\rceil}(x,T)$ is contained in a disk $D_{\lceil t/2\rceil}(x,G)$ of G. It is easy to see that all three properties of tree-decomposition are fulfilled for $\mathcal{T}$.

Combining Lemma 7 with Theorem 3 we get our main result.

Theorem 4. *There is an algorithm that for an arbitrary connected graph G with n vertices and m edges constructs a tree $(2\lceil t/2\rceil\lfloor\log_2 n\rfloor)$-spanner in $O(nm\log^2 n)$ time, whenever G admits a tree t-spanner.*

The complexity of our algorithm is dominated by the complexity of finding a balanced disk-separator $D_r(v,G)$ of a graph G with minimum r. Proposition 2 says that for an n-vertex, m-edge graph such a balanced disk-separator can be found in $O(nm)$ time. In the full version of the paper, we show that a balanced disk-separator of a graph G with radius $r \leq 6\cdot \mathtt{tb}(G)$ can be found in linear $O(m)$ time. This immediately leads to the following result.

Theorem 5. *There is an algorithm that for an arbitrary connected graph G with n vertices and m edges constructs a tree $(6t\lfloor\log_2 n\rfloor)$-spanner in $O(m\log n)$ time, whenever G admits a tree t-spanner.*

References

1. Abraham, I., Bartal, Y., Neiman, O.: Nearly Tight Low Stretch Spanning Trees. In: FOCS, pp. 781–790 (2008)
2. Alon, N., Karp, R.M., Peleg, D., West, D.B.: A Graph-Theoretic Game and Its Application to the k-Server Problem. SIAM J. Comput. 24, 78–100 (1995)
3. Bădoiu, M., Indyk, P., Sidiropoulos, A.: Approximation algorithms for embedding general metrics into trees. In: SODA 2007, pp. 512–521. SIAM, Philadelphia (2007)
4. Berge, C.: Hypergraphs. North-Holland, Amsterdam (1989)
5. Brandstädt, A., Dragan, F.F., Le, H.-O., Le Tree Spanners, V.B.: on Chordal Graphs: Complexity and Algorithms. Theoret. Comp. Science 310, 329–354 (2004)
6. Brandstädt, A., Dragan, F.F., Le, H.-O., Le, V.B., Uehara, R.: Tree spanners for bipartite graphs and probe interval graphs. Algorithmica 47, 27–51 (2007)
7. Buneman, A.: A characterization of rigid circuit graphs. Discrete Math. 9, 205–212 (1974)
8. Cai, L., Corneil, D.G.: Tree spanners. SIAM J. Discr. Math. 8, 359–387 (1995)
9. Chepoi, V.D., Dragan, F.F., Newman, I., Rabinovich, Y., Vaxes, Y.: Constant Approximation Algorithms for Embedding Graph Metrics into Trees and Outerplanar Graphs. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX 2010, LNCS, vol. 6302, pp. 95–109. Springer, Heidelberg (2010)
10. Dourisboure, Y., Dragan, F.F., Gavoille, C., Yan, C.: Spanners for bounded tree-length graphs. Theor. Comput. Sci. 383, 34–44 (2007)
11. Dourisboure, Y., Gavoille, C.: Tree-decompositions with bags of small diameter. Discrete Mathematics 307, 2008–2029 (2007)
12. Dragan, F.F., Fomin, F., Golovach, P.: Spanners in sparse graphs. J. Computer and System Sciences (2010), doi: 10.1016/j.jcss.2010.10.002
13. Elkin, M., Emek, Y., Spielman, D.A., Teng, S.-H.: Lower-Stretch Spanning Trees. SIAM J. Comput. 38, 608–628 (2008)
14. Emek, Y., Peleg, D.: Approximating minimum max-stretch spanning trees on unweighted graphs. SIAM J. Comput. 38, 1761–1781 (2008)
15. Fekete, S.P., Kremer, J.: Tree spanners in planar graphs. Discrete Appl. Math. 108, 85–103 (2001)
16. Gavoille, C., Katz, M., Katz, N.A., Paul, C., Peleg, D.: Approximate Distance Labeling Schemes. In: Meyer auf der Heide, F. (ed.) ESA 2001. LNCS, vol. 2161, pp. 476–488. Springer, Heidelberg (2001)

17. Gavril, F.: The intersection graphs of subtrees in trees are exactly the chordal graphs. J. Comb. Theory (B) 16, 47–56 (1974)
18. Gilbert, J.R., Rose, D.J., Edenbrandt, A.: A separator theorem for chordal graphs. SIAM J. Algebraic Discrete Methods 5, 306–313 (1984)
19. Liebchen, C., Wünsch, G.: The zoo of tree spanner problems. Discrete Appl. Math. 156, 569–587 (2008)
20. Lokshtanov, D.: On the complexity of computing tree-length. Discrete Appl. Math. 158, 820–827 (2010)
21. Peleg, D.: Low Stretch Spanning Trees. In: Diks, K., Rytter, W. (eds.) MFCS 2002. LNCS, vol. 2420, pp. 68–80. Springer, Heidelberg (2002)
22. Peleg, D., Reshef, E.: Low complexity variants of the arrow distributed directory. J. Comput. System Sci. 63, 474–485 (2001)
23. Peleg, D., Tendler, D.: Low stretch spanning trees for planar graphs, Tech. Report MCS01-14, Weizmann Science Press of Israel, Israel (2001)
24. Makowsky, J.A., Rotics, U.: Optimal spanners in partial k-trees, manuscript
25. Robertson, N., Seymour, P.D.: Graph minors. II. Algorithmic aspects of tree-width. Journal of Algorithms 7, 309–322 (1986)
26. `http://www.cs.kent.edu/~dragan/papers/Approx-tree-spanner-FullVers.pdf`