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:

- a tree $(2\lfloor \log_2 n \rfloor)$-spanner in $O(m \log n)$ time, if G is a chordal graph (i.e., every induced cycle of G has length 3);
- 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
- 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 $\lceil t/2 \rceil$ and diameter at most t 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., [27] 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 [10, 14, 28]), and the paper presents an efficient algorithm which produces a tree t-spanner with $t \leq 2 \log_2 n$ 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 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. Tree spanners are useful in designing approximation algorithms for combinatorial and algorithmic problems that are concerned with distances in a finite metric space induced by a graph. An arbitrary metric space (in particular a finite metric defined by a graph) might not have enough structure to exploit algorithmically. If we approximate the graph distances by the distances in a tree, we can solve the problem on

the tree and interpret the solution on the original graph. Tree spanners find applications also in network design and, in particular, in the context of distributed systems. One such application is the *arrow distributed directory protocol* introduced in [20]. This protocol supports the location of mobile objects in a distributed network. It is implemented over a spanning tree T that spans the network, and, as shown in [38], the worst case overhead ratio of the protocol is proportional to the stretch of T. Therefore, a good candidate for the backbone of the arrow protocol is a spanning tree with low stretch (see also [33]). Another application of tree spanners is in message routing in communication networks. In order to maintain succinct routing tables, efficient routing schemes can use only the edges of a tree spanner. A very efficient routing scheme is available for trees [42]. We refer to the survey paper of Peleg [37] 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 [14] 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 [10] and [11], 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 [28, 39]. In [39], Peleg and Tendler presented a polynomial time algorithm for the minimum value of t for the TREE t-SPANNER problem on outerplanar graphs. In [28], 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 [28]. This open problem was recently resolved in [24], 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. Note also that a number of particular graph classes (like interval graphs, permutation graphs, asteroidal-triple–free graphs, strongly chordal graphs, dually chordal graphs, and others) admit tree t-spanners for small values of t (we refer reader to [9–11, 14, 24, 27, 28, 34, 35, 37–40] and papers cited therein).

An $O(\log n)$-approximation algorithm for the minimum value of t for the TREE t-SPANNER problem is due to Emek and Peleg [27], 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 [35], 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 [38] for an earlier result). Recently, another logarithmic approximation algorithm for the TREE t-SPANNER problem was announced in [5], 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, 25] and papers cited therein). One should also mention

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

paper of Elkin and Peleg [26] which describes an algorithm that, given a graph G admitting a tree t-spanner, constructs a t-spanner of G with $O(n \log n)$ edges.

Our contribution. In this paper, we present a new algorithm which constructs for an n-vertex m-edge unweighted graph G:

- a tree $(2\lfloor \log_2 n \rfloor)$-spanner in $O(m \log n)$ time, if G is a chordal graph;
- 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
- 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 [27], 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 ([27]), 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.[2] 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.

Outline of the paper. In Section 2, we present the basic notions and notations used throughout the paper. As a warm up, in Section 3, we demonstrate our method on a simpler case, on the class of chordal graphs. The result of this section is of independent interest as it demonstrates that every chordal graph admits a logarithmic tree spanner, i.e., a tree t-spanner with $t \leq 2\lfloor \log_2 n \rfloor$. We complement this with a lower bound result which says that there are chordal graphs for which any tree t-spanner has to have $t \geq \log_2 \frac{n}{3} + 2$. In Section 4, we extend our method to all graphs admitting a Robertson-Seymour's tree-decomposition with bags of radius at most ρ. Section 5 is devoted to general (unweighted) graphs. We show there that any graph having a tree t-spanner admits a Robertson-Seymour's tree-decomposition with bags of radius at most $\lceil t/2 \rceil$ and diameter at most t. Combining this with the result of Section 4, we obtain our approximation algorithms for general graphs. Section 6 concludes the paper.

[2] We realize that it is perfectly possible that the authors of [27] did not try to optimize the constants in their analysis, and it may be the case that a more careful analysis of their algorithm may lead to an improved leading constant.

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 (if a graph G admits a tree t-spanner with $t = 1$ then G itself must be a tree).

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 [10]. 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 $t \leq 2 \log_2 n$ and that there are chordal graphs for which any tree t-spanner has to have $t \geq \log_2 \frac{n}{3} + 2$.

3.1 Lower bound

We use the construction of "bad" chordal graphs (so called *snowflakes*) presented in [34]. Let C_l denote a simple cycle with l edges. A *snowflake with one layer* (denoted by SF_1) is just a triangle C_3, i.e., an outerplanar graph whose outer face is C_3. A *snowflake with two layers* (denoted by SF_2) is obtained from SF_1 by adding, for each edge of the outer face C_3 of SF_1, a new vertex

adjacent in SF_2 only to endvertices of that edge. New vertices are placed in the outer face of SF_1 to have a nice layered embedding of SF_2 on the plane (see Fig. 1 for an illustration). Clearly, SF_2 is an outerplanar graph with the outer face (second layer) being C_6. Generally, a *snowflake with k layers*, $k \geq 2$, (denoted by SF_k) is obtained from the outerplanar graph SF_{k-1} by adding, for each edge of the outer face $C_{3\cdot 2^{k-2}}$ of SF_{k-1}, a new vertex adjacent in SF_k only to endvertices of that edge. Again, all new vertices are placed in the outer face of SF_{k-1}. Clearly, SF_k is an outerplanar graph with the outer face being $C_{3\cdot 2^{k-1}}$.

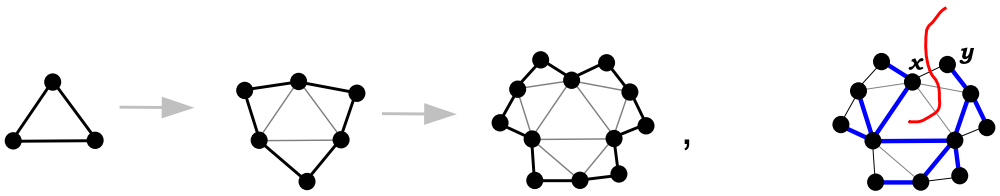

Fig. 1. Left picture: snowflakes SF_1, SF_2 and SF_3. Right picture: any tree t-spanner of SF_3 has to have $t \geq 4$.

Proposition 2. *For every integer $k \geq 1$, graph SF_k is outerplanar and chordal, it has $n = 3\cdot 2^{k-1}$ vertices and has no tree t-spanners with $t < k+1 = \log_2 \frac{n}{3} + 2$.*

Proof. Clearly, SF_k is an outerplanar graph with $n = 3 \cdot 2^{k-1}$ vertices (the number of vertices in SF_k is twice the number of vertices in SF_{k-1}). SF_k is a chordal graph (even a 2-tree) since it can be constructed from an edge by repeatedly adding a new vertex and making it adjacent to two old adjacent vertices (see survey [12] for details). Assume that SF_k is naturally embedded on the plane (see the construction above and Fig. 1). To show that the outerplanar graph SF_k has no tree t-spanners with $t < k+1$, consider an arbitrary spanning tree T of SF_k. Since T is a planar graph with only the outer face, we can connect by a Jordan curve $\mathcal{C}$ a point in the outer face of SF_k with a point of the plane inside the central triangle of SF_k (i.e., the triangle of SF_1; see Fig. 1) without intersecting the tree T. Let $\triangle$ be the first non-outer face of SF_k crossed by $\mathcal{C}$ and xy be the edge of $\triangle$ crossed first. Clearly, for x and y, $d_T(x,y) \geq k+1$ holds, while $d_G(x,y) = 1$. Thus, the stretch factor of T is at least $k+1$. For an alternative proof of the fact that SF_k has no tree t-spanners with $t < k+1$, see [34]. □

3.2 Upper bound

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 $C \subseteq V$ 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. 2 for an illustration). 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.

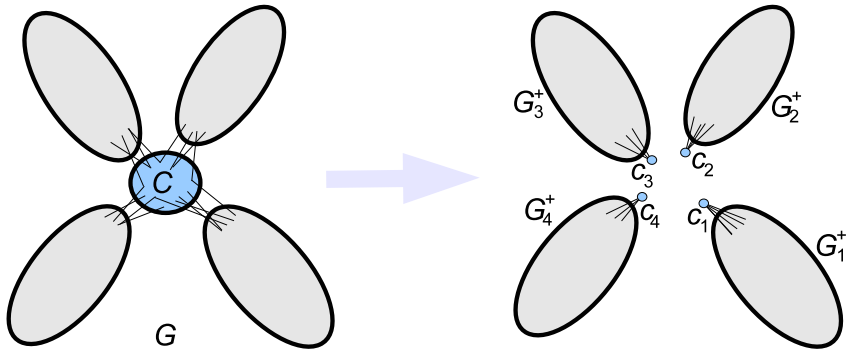

Fig. 2. A graph G with a clique-separator C and the corresponding graphs $G_1^+, \ldots, G_4^+$ obtained from G.

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 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 $i = 1, \ldots, k$.*

Proof. Clearly, contracting any edge e in a chordal graph G cannot result in creating in $G_{/e}$ an induced cycle with more than 3 vertices (otherwise, a similar induced cycle must be present in G as well, which is impossible). We can get G_i^+ from G by repeatedly contracting (in any order) edges of G that are not incident to vertices of G_i. □

Let T_i $(i = 1, \ldots, k)$ 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. We can form a spanning tree T of G from trees $T_1, \ldots, T_k$ and the vertices of the clique C in the following way. For each $i = 1, \ldots, k$, rename vertex c_i in T_i to c. Glue trees $T_1, \ldots, T_k$ together at vertex c obtaining a tree T' (see Fig. 3). 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 $xy \in E(G)$, $d_T(x, y) \leq \alpha + 2$ holds. Evidently, the tree T of G can be constructed from trees $T_1, \ldots, T_k$ and the vertices of the clique C in $O(m)$ time.

Lemma 2. *Let G be an arbitrary graph with a clique-separator C and $G_1^+, \ldots, G_k^+$ be the graphs obtained from G as described above. Let also T_i $(i \in \{1, \ldots, k\})$ 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 the clique C 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 $xy \in E(G)$, $d_T(x, y) \leq \alpha + 2$ must hold.*

Proof. Consider an arbitrary edge xy of G. If both x and y belong to C, then evidently $d_T(x, y) \leq 2 < \alpha + 2$. Assume now that xy is an edge of G_i for some $i \in \{1, \ldots, k\}$. Then, xy is an edge of G_i^+ and therefore $d_{T_i}(x, y) \leq \alpha$. If the path P of T_i connecting x and y does not contain vertex c_i, then $d_T(x, y) = d_{T_i}(x, y) \leq \alpha$ must hold. If c_i is between x and y in T_i (i.e., $c_i \in P$), then the

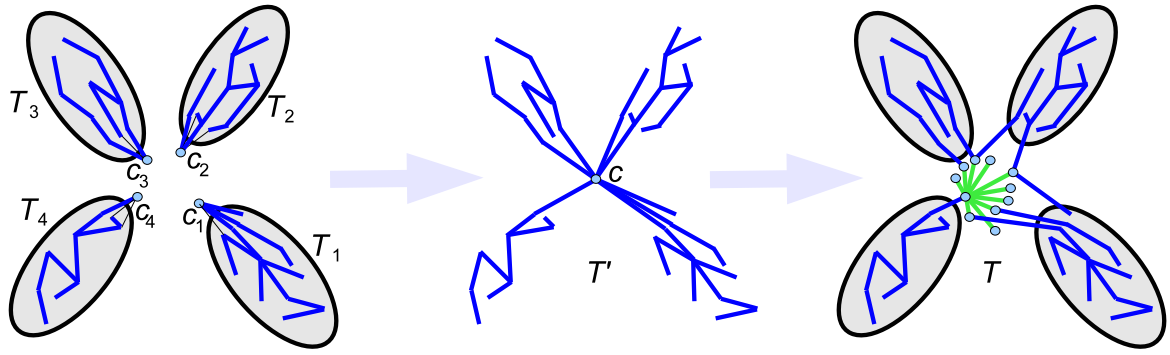

Fig. 3. Spanning trees $T_1, \ldots, T_4$ of $G_1^+, \ldots, G_4^+$, resulting tree T', and a corresponding spanning tree T of G.

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 $d_{T_i}(c_i, y) \leq \alpha$. 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, $d_T(x,y) \leq d_T(z_C, y) + 2 = d_{T_i}(c_i, y) + 2 \leq \alpha + 2$. □

The third important ingredient to our method is the famous chordal balanced separator result of Gilbert, Rose, and Edenbrandt [31].

Lemma 3. [31] *Every chordal graph G with n vertices and m edges contains a maximal clique C such that if the vertices in C are deleted from G, every connected component in the graph induced by any remaining vertices is of size at most $n/2$. Such a balanced clique-separator C of a connected chordal graph G 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 $G_1^+, \ldots, G_k^+$. For each graph G_i^+, $i \in \{1, \ldots, k\}$, 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

$$\frac{n}{2^k} + \frac{1}{2^{k-1}} + \ldots + \frac{1}{2} + 1 \leq 5,$$

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, $d_T(x,y) \leq t$ 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 $d_{T_L}(x,y) \leq 2$. 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. [13, 30, 43] *A graph is chordal if and only if it has a clique-tree.*

In their work on graph minors [41], 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 $\mathsf{tw}(G)$ [41]. 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 [23]. These two graph parameters are not related to each other. For instance, cliques (or, generally, all chordal graphs) have unbounded tree-width and tree-length 1, whereas cycles have tree-width 2 and unbounded tree-length.

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 5.1 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 [10]), while it is polynomial time solvable for all graphs of bounded tree-width (see [40]).

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 if the vertices of disk $D_\rho(v, G)$ are deleted from G, every connected component in the graph induced by any remaining vertices is of size at most $n/2$.*

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 [7].

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., [3, 6, 7]) 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 [36]).

Instead, we can use the following result.

Proposition 3. *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.*

Proof. We need to show that for each vertex v of G a balanced disk-separator $D_r(v,G)$ with minimum r can be found in $O(m)$ time. Then we can iterate over all vertices of G and find a vertex v^* whose $D_r(v^*,G)$ has the smallest radius r.

Construct a *layering* $L_0, L_1, \ldots, L_q$ of G with respect to v, where $L_i = \{u \in V : d_G(v,u) = i\}$, $i \in \{0,1,2,\ldots,q\}$ and $q := \max_{x\in V} d_G(v,x)$. Let $G_i^1, \ldots, G_i^{l_i}$ be the connected components of the graph $G[V \setminus D_{i-1}(v,G)]$, $i \in \{1,2,\ldots,q\}$. Let also $n_i^j := |V(G_i^j)|$, $i \in \{1,2,\ldots,q\}$ and $j \in \{1,2,\ldots,l_i\}$. We need to find the smallest r such that all n_{r+1}^j, $j \in \{1,2,\ldots,l_{r+1}\}$ are less than or equal to $n/2$, and for some $j \in \{1,2,\ldots,l_r\}$, n_r^j is larger than $n/2$. All we need is to iteratively compute values n_i^j starting with $i = q$ and ending with $i = r$. For this we can use the following simple procedure (which needs to work on a copy of the original graph G since the input graph gets modified during the procedure).

Initially, $mult(x) := 1$ for each vertex x of G; /* multiplicity of a vertex is set to 1 */
Compute connected components $\widehat{G}_q^1, \ldots, \widehat{G}_q^{l_q}$ of the graph $G[L_q] = G[V \setminus D_{q-1}(v,G)]$;
Compute the corresponding cardinalities $n_q^j := |V(\widehat{G}_q^j)| = |V(G_q^j)|$, $j \in \{1,2,\ldots,l_q\}$;
For $i = q$ downto 1 do
  If for all $j \in \{1,2,\ldots,l_i\}$, $n_i^j \le n/2$
    Change G by contracting in G each component $\widehat{G}_i^j$ to a vertex a_i^j, $j = 1,2,\ldots,l_i$;
    Define the multiplicity of a_i^j as $mult(a_i^j) := n_i^j$;
    Set $\widehat{L}_{i-1} := L_{i-1} \cup \{a_i^1, \ldots, a_i^{l_i}\}$;
    Compute the connected components $\widehat{G}_{i-1}^1, \ldots, \widehat{G}_{i-1}^{l_{i-1}}$ of the graph $G[\widehat{L}_{i-1}]$;
    Set $n_{i-1}^j := \sum_{x \in V(\widehat{G}_{i-1}^j)} mult(x)$, $j \in \{1,2,\ldots,l_{i-1}\}$;
  Else output $i-1$.

Clearly, for each vertex v of G, this method finds in linear $O(m)$ time a balanced disk-separator $D_r(v,G)$ with minimum r. □

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 $G_1, \ldots, G_k$ 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 $xx_D \in E(G)$. Given graph G and its disk-separator $D_r(v,G)$, the graphs $G_1^+, \ldots, G_k^+$ can be constructed in total time $O(m)$. Furthermore, the total number of edges in the graphs $G_1^+, \ldots, G_k^+$ 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, $\mathsf{tb}(G) \le \rho$ implies $\mathsf{tb}(G_{/e}) \le \rho$. Consequently, for any graph G with $\mathsf{tb}(G) \le \rho$, $\mathsf{tb}(G_i^+) \le \rho$ holds for each $i = 1,\ldots,k$.*

Proof. Let G be a graph with $\mathsf{tb}(G) = \rho$ and $\mathcal{T}(G)$ be its tree-decomposition of breadth ρ. Let $e = xy$ be an arbitrary edge of G. We can obtain a tree-decomposition $\mathcal{T}(G_{/e})$ of graph $G_{/e}$ by

replacing in each bag $X \in V(\mathcal{T}(G))$ vertices x and y with a new vertex x' representing them (if some bag A contained both x and y, only one copy of x' is kept). Evidently, properties (1) and (2) in the definition of tree-decomposition are fulfilled for $\mathcal{T}(G_{/e})$. Furthermore, the topology of the tree-decomposition did not really change. Still, for any vertex $v \neq x'$ of $G_{/e}$, the bags of $\mathcal{T}(G_{/e})$ containing v form a subtree in $\mathcal{T}(G_{/e})$. Since vertices x and y were adjacent in G, there was a bag A of $\mathcal{T}(G)$ containing both those vertices. Hence, a subtree of $\mathcal{T}(G_{/e})$ formed by bags of $\mathcal{T}(G_{/e})$ containing vertex x' is nothing else but the union of two subtrees (one for x and one for y) of $\mathcal{T}(G)$ sharing at least one common bag A. Also, contracting an edge can only reduce the distances in a graph. Hence, still, for each bag B of $\mathcal{T}(G_{/e})$, there must exists a corresponding vertex b in $G_{/e}$ with $B \subseteq D_\rho(b, G_{/e})$.

We can get G_i^+ from G again by repeatedly contracting (in any order) edges of G that are not incident to vertices of G_i. □

As in Section 3, let T_i $(i = 1, \ldots, k)$ 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 $T_1, \ldots, T_k$ and SPT_D in the following way. For each $i = 1, \ldots, k$, rename vertex v_i in T_i to v. Glue trees $T_1, \ldots, T_k$ together at vertex v obtaining a tree T' (consult with Fig. 3). Substitute vertex v in T' by the tree SPT_D. For each former edge xv of T', create an edge xx_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 $T_1, \ldots, T_k$ and SPT_D in $O(m)$ time.

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

Proof. The proof is done along the lines of the proof of Lemma 2. Consider an arbitrary edge xy of G. If both x and y belong to $D_r(v, G)$, then evidently $d_T(x, y) \leq 2r < \alpha + 2r$. Assume now that xy is an edge of G_i for some $i \in \{1, \ldots, k\}$. Then, xy is an edge of G_i^+ and therefore $d_{T_i}(x, y) \leq \alpha$. If the path P of T_i connecting x and y does not contain vertex v_i, then $d_T(x, y) = d_{T_i}(x, y) \leq \alpha$ must hold. If v_i is between x and y in T_i (i.e., $v_i \in P$), then the distance in T between x and y is at most $d_{T_i}(x, y) + 2r$ (the path of T between x and y is obtained from P by substituting the vertex $v = v_i$ by a path of tree SPT_D with at most $2r$ edges). It remains to consider the case when $x \in D_r(v, G)$ and $y \in V(G_i)$ (see Fig. 4 for an illustration). By construction of G_i^+, there must exist an edge $v_i y$ in G_i^+. We have $d_{T_i}(v_i, y) \leq \alpha$. Let z be the neighbor of v_i in the path of T_i connecting vertices y and v_i ($y = z$ is possible). Evidently, $z \in V(G_i)$. By construction, we must have in T an edge zz_D where z_D is a vertex of $D_r(v, G)$ adjacent to z in G. Vertices x and z_D both are in $D_r(v, G)$ and the distance in T between them is at most $2r$. Thus, $d_T(x, y) \leq d_T(z_D, y) + 2r = d_{T_i}(v_i, y) + 2r \leq \alpha + 2r$. □

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 3 says that

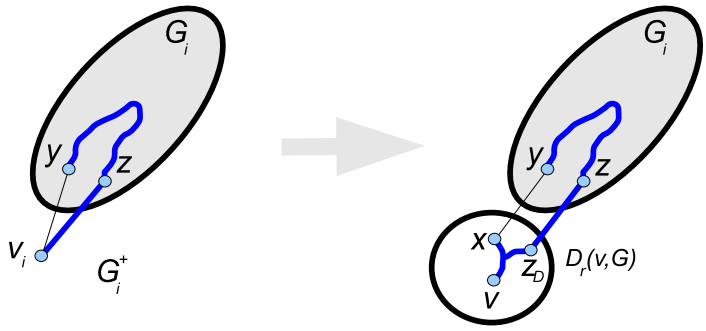

Fig. 4. Illustration to the proof of Lemma 6.

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 $O(n \log n)$.

Here is a summarized recursive version of our algorithm. G is an arbitrary input graph.

```
Tree_Spanner(G)
    If G has at most 9 vertices
        Find a tree t-spanner T of G with minimum t directly;
        Output T.
    Else
        Find a balanced disk-separator D_r(v,G) of G with minimum r (see Proposition 3);
        Find connected components G_1, ..., G_k of graph G[V \ D_r(v,G)];
        Build graphs G_1^+, ..., G_k^+ as described before Lemma 5;
        Set T_i := Tree_Spanner(G_i^+), for each i = 1, ..., k;
        Construct a shortest path tree SPT_D of G[D_r(v,G)] rooted at vertex v;
        Construct a spanning tree T of G from trees T_1, ..., T_k and SPT_D as described before Lemma 6;
        Output T.
```

Note that the 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 [29] 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.

5.1 Graphs admitting tree t-spanners have tree-breadth at most $\lceil t/2\rceil$

Here, 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 will follow 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.

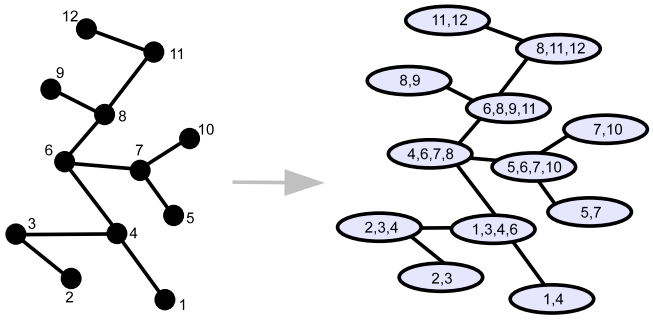

Fig. 5. From tree T to tree-decomposition $\mathcal{T}$ with $t = 2$.

Lemma 7. *If a graph G admits a tree t-spanner then its tree-breadth is at most $\lceil t/2 \rceil$.*

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. 5 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. Thus, it remains to show that all three properties of tree-decomposition are fulfilled for $\mathcal{T}$.

Evidently, every vertex x of G is in at least one bag of $\mathcal{T}$. Consider an arbitrary edge uv of G. Since T is a tree t-spanner of G, $d_T(u,v) \leq t$ holds. Let x be a middle vertex of the path connecting vertices u and v in T. Then, both u and v belong to the disk $D_{\lceil t/2 \rceil}(x,T)$, i.e, there is a bag $X = D_{\lceil t/2 \rceil}(x,T)$ containing u and v. For a vertex $v \in V(G)$ consider the set of all bags $\{X_i = D_{\lceil t/2 \rceil}(x_i,T) : i = 1,2,\ldots,l\}$ of $\mathcal{T}$ containing vertex v. This set of bags induces a subtree in $\mathcal{T}$ since the corresponding vertices $x_1,\ldots,x_l$ induce a subtree in T. Note that $v \in X_i$ if and only if $d_T(v,x_i) \leq \lceil t/2 \rceil$. □

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.*

5.2 Improving run-time on expense of approximation ratio

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 3 says that for an n-vertex, m-edge graph such a balanced disk-separator can be found in $O(nm)$ time. In this subsection, we show that a balanced disk-separator with a slightly larger radius can be found in linear $O(m)$ time. This will immediately lead to an $O(m \log n)$ algorithm (as in the case of chordal graphs; see Section 3), which produces for every graph G a tree $(6t\lfloor \log_2 n \rfloor)$-spanner, whenever G admits a tree t-spanner.

We will need the notion of *layering partition* introduced in papers [9, 15] and recently used in a slightly more general form in both approximation algorithms for embedding graph metric into trees [4, 5, 18] as well as in some other similar contexts [16–19, 22, 23].

Let $G = (V,E)$ be a connected graph with n vertices and m edges and with a distinguished vertex s. Consider a *layering* $L_0, L_1, \ldots, L_q$ of G with respect to s, where $L_i = \{u \in V : d_G(s,u) = i\}$, $i \in \{0,1,2,\ldots,q\}$ and $q := \max_{x \in V} d_G(s,x)$. A *layering partition* $\mathcal{L}P(s) = \{L_i^1, \ldots, L_i^{p_i} : i = 0,1,2,\ldots,q\}$ of G is a partition of each L_i into *clusters* $L_i^1, \ldots, L_i^{p_i}$ such that two vertices $u, v \in L_i$ belong to the same cluster L_i^j if and only if they can be connected by a path outside the disk $D_{i-1}(s,G)$. It was shown in [15] that for a given graph G such a layering partition can be found in $O(m)$ time. Let Γ be a graph whose vertex set is the set of all clusters L_i^j in a layering partition $\mathcal{L}P$ of G. Two vertices $C = L_i^j$ and $C' = L_{i'}^{j'}$ are adjacent in Γ if and only if there exist $u \in L_i^j$ and $v \in L_{i'}^{j'}$ such that u and v are adjacent in G (see Fig. 6). It was shown in [15] that Γ is a tree, called the *layering tree* of G, and that Γ is computable in linear time in the size of G.

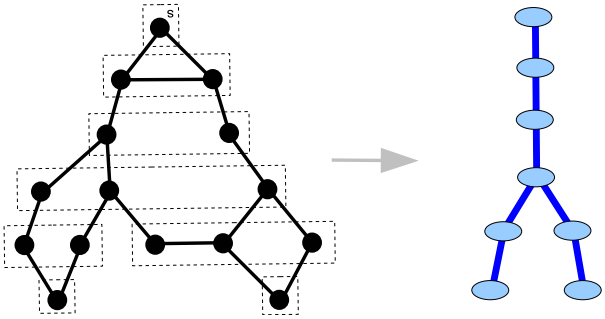

Fig. 6. A layering partition of G and the tree Γ associated with this layering partition.

Let assign to each vertex L_i^j of Γ a weight $w_i^j := |L_i^j|$. Clearly, $W := \sum_{i=0,1,2,\ldots,q,j=1,2,\ldots,p_i} w_i^j$ is equal to n. It is known that every vertex-weighted tree T with the total weight of vertices equal to W has a vertex x, called a *median* of T, such that the total weight of vertices in each subtree of $T \setminus \{x\}$ does not exceed $W/2$. Furthermore, such a vertex x of T can be found in $O(|V(T)|)$ time [32]. Let $C = L_i^j$ $(i \in \{0,1,2,\ldots,q\}, j \in \{1,2,\ldots,p_i\})$ be a median vertex of weighted tree Γ. Then, each subtree of $\Gamma \setminus \{C\}$ has total weight of vertices not exceeding $n/2$. It is clear from the construction of tree Γ that the set C separates in G any two vertices that belong to clusters from different subtrees of $\Gamma \setminus \{C\}$. Consequently, C is a balanced separator of G as any connected component of $G[V \setminus C]$ has no more than $n/2$ vertices. Note that, given a graph G, such a cluster C of a layering partition $\mathcal{L}P$ of G can be found in linear time in the size of G.

We show now that if graph G has tree-breadth ρ then there is a vertex v in G such that $C \subseteq D_{3\rho}(v, G)$.

Lemma 8. *If a graph G has tree-breadth ρ then for any cluster C of a layering partition $\mathcal{L}P$ of G there exists a vertex $v_C \in V(G)$ such that $C \subseteq D_{3\rho}(v_C, G)$.*

Proof. The proof is analogous to the proof of a similar result for graphs with tree-length λ (see [22], Lemma 5). Let $\mathcal{T}$ be a tree-decomposition of G with breadth ρ. It is known [21] that if X_1X_2 is an edge of a tree-decomposition $\mathcal{T}$ of G, and $\mathcal{T}_1$, $\mathcal{T}_2$ are the subtrees of $\mathcal{T}$ obtained after removing edge X_1X_2 from $\mathcal{T}$, then $I = X_1 \cap X_2$ separates in G vertices belonging to bags of $\mathcal{T}_1$ but not to I from vertices belonging to bags of $\mathcal{T}_2$ but not to I. We will need this property of a tree-decomposition below.

Assume that $\mathcal{T}$ is rooted at a bag containing vertex s, the source of layering partition $\mathcal{L}P$. Let C be a cluster from layer L_i (i.e, $C = L_i^j$ for some $j \in \{1,2,\ldots,p_i\}$). We have $d_G(x,s) = i$ for every $x \in C$. Let Z be the nearest common ancestor of all bags of $\mathcal{T}$ containing vertices of C. Let z be a vertex of G such that $Z \subseteq D_\rho(z, G)$.

Consider an arbitrary vertex x of C. It is easy to see that there is a vertex $y \in C$ and two bags X and Y of $\mathcal{T}$ containing vertices x and y, respectively, such that $Z = NCA_{\mathcal{T}}(X, Y)$ (i.e., Z is the nearest common ancestor of X and Y in $\mathcal{T}$; see Fig. 7). Let P be a shortest path of G from s to x. Necessarily, P intersects Z. Let a be a vertex of $P \cap Z$ closest to s in G. Since x and y both are in C, there exists a path Q from x to y in G using only intermediate vertices w with $d_G(w,s) \geq i$. Assume Q intersects Z at vertex b. We have $d_G(s,x) = i = d_G(s,a) + d_G(a,x)$ and $i \leq d_G(s,b) \leq d_G(s,a) + d_G(a,z) + d_G(z,b) \leq d_G(s,a) + 2\rho$. Hence, $d_G(a,x) = i - d_G(s,a) \leq 2\rho$ and therefore $d_G(x,z) \leq d_G(x,a) + d_G(a,z) \leq 2\rho + \rho = 3\rho$.

Thus, any vertex x of C is at distance at most 3ρ from z in G, i.e., $C \subseteq D_{3\rho}(z, G)$. □

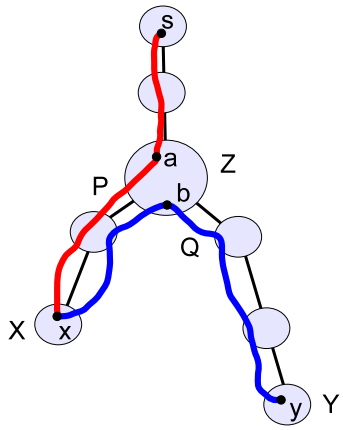

Fig. 7. Illustration to the proof of Lemma 8.

Given a cluster C of a layering partition of G, although we know that a vertex $v_C \in V(G)$ exists such that $C \subseteq D_{3\rho}(v_C, G)$, we do not currently know how to find v_C in $O(m)$ time. Instead, we can consider any vertex v of C and use $D_{6\rho}(v, G)$ as a "small radius" balanced disk-separator of G. Clearly, $C \subseteq D_{6\rho}(v, G)$ and therefore $D_{6\rho}(v, G)$ is a balanced disk-separator of G.

Corollary 4. *There is an algorithm that for an arbitrary connected graph G with n vertices and m edges finds a balanced disk-separator $D_{6\rho}(v, G)$ of G in linear $O(m)$ time, where $\rho = \mathsf{tb}(G)$.*

As a consequence, we have the following variant of Theorem 3.

Theorem 5. *There is an algorithm that for an arbitrary connected graph G with n vertices and m edges constructs a tree $(12\mathsf{tb}(G)\lfloor \log_2 n \rfloor)$-spanner of G in $O(m \log n)$ total time.*

This theorem already implies that there is an algorithm that for an arbitrary connected graph G with n vertices and m edges constructs a tree $(12\lceil t/2 \rceil \lfloor \log_2 n \rfloor)$-spanner in $O(m \log n)$ time, whenever G admits a tree t-spanner. But, if we decided to use an arbitrary vertex v of C as the center of a balanced disk-separator, we can get a similar (even slightly better for odd ts) result using just notions of tree-length and t-powers of trees. We can prove the following result.

Theorem 6. *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.*

Proof. First, we can show that if $G = (V, E)$ admits a tree t-spanner, then the tree-length of G is at most t. Indeed, let T be a tree t-spanner of G. The *t-power* T^t of T is a graph obtained from T by adding to T all new edges between vertices at distance at most t in T, i.e., for each $x, y \in V$, $xy \in E(T^t)$ if and only if $d_T(x, y) \leq t$. Since T is a tree t-spanner of G, i.e., $d_T(x, y) \leq t$ for every edge xy of G, G is a spanning subgraph of T^t. It is known that any power of a tree is a chordal graph (see, e.g., [12]). Consequently, T^t is chordal and has a clique-tree (see Theorem 2). This clique-tree of T^t gives a tree-decomposition of G. Each bag of that tree-decomposition forms a clique in T^t and, therefore, is a set of vertices $S \subseteq V$ such that $d_G(x, y) \leq d_T(x, y) \leq t$ for every $x, y \in S$. Thus, $\mathsf{tl}(G) \leq t$.

Second, it was proven in [22] that if $\mathsf{tl}(G) \leq t$, then for any cluster C of a layering partition $\mathcal{L}P$ of G and every two vertices $x, y \in C$, $d_G(x, y) \leq 3t$ holds. Hence, for any vertex $v \in C$,

$C \subseteq D_{3t}(v, G)$. Moreover, given G, we can find a balanced disk-separator $D_{3t}(v, G)$ of G in total $O(m)$ time (see the discussion before Lemma 8).

Third, it is easy to see that if $\mathsf{tl}(G) \leq t$ then $\mathsf{tl}(G_{/e}) \leq t$ for any edge e of G (see the proof of Lemma 5).

Consequently, these facts together with Lemma 6, plugged into our method, prove the theorem. □

6 Concluding remarks

In this paper, we examined the TREE t-SPANNER problem on chordal graphs, generalized chordal graphs and general graphs. Using a graph decomposition technique based on balanced disk-separators, we developed an algorithm which produces for any input unweighted graph a tree t-spanner with t close to minimum.

We conclude the paper with some problems for future research.

1. Investigate graphs with bounded tree-breadth. A better structural understanding may lead to a better approximation algorithm for the TREE t-SPANNER problem. Is $o(\log n)$-approximation algorithm for the TREE t-SPANNER problem achievable?
2. Characterize and recognize graphs with tree-breadth 1 (with tree-breadth at most ρ).
3. Get faster algorithms for finding a "small radius" balanced disk-separator of a graph. Can a balanced disk-separator with minimum radius be found in $o(nm)$ time (in $O(m)$ time)?

References

1. I. Abraham, Y. Bartal, O. Neiman, Nearly Tight Low Stretch Spanning Trees, In *FOCS* 2008, pp. 781–790.
2. N. Alon, R.M. Karp, D. Peleg, D.B. West, A Graph-Theoretic Game and Its Application to the k-Server Problem, *SIAM J. Comput.* 24 (1995), 78–100.
3. G. Ausiello, A. D'Arti, and M. Moscarini, Chordality properties on graphs and minimal conceptual connections in sematic data models, *J. Comput. System Sci.*, 33 (1986), 179–202.
4. M. Bădoiu, E.D. Demaine, M.T. Hajiaghayi, A. Sidiropoulos, and M. Zadimoghaddam, Ordinal embedding: approximation algorithms and dimensionality reduction, In *APPROX-RANDOM* 2008, pp. 21–34.
5. M. Bădoiu, P. Indyk, and A. Sidiropoulos, Approximation algorithms for embedding general metrics into trees, In *SODA* 2007, SIAM, pp. 512–521.
6. C. Beeri, R. Fagin, D. Maier and M. Yannakakis, On the desirability of acyclic database schemes, *J. ACM,* 30 (1983), 479–513.
7. C. Berge, Hypergraphs, *North Holland*, 1989.
8. H.L. Bodlaender, A linear time algorithm for finding tree-decompositions of small treewidth, *SIAM Journal on Computing*, 25 (1996), 1305–1317.
9. A. Brandstädt, V. Chepoi, and F. Dragan, Distance approximating trees for chordal and dually chordal graphs. *J. Algorithms*, 30 (1999), 166–184.
10. A. Brandstädt, F.F. Dragan, H.-O. Le, and V.B. Le, Tree Spanners on Chordal Graphs: Complexity and Algorithms, *Theoretical Computer Science*, 310 (2004), 329-354.
11. A. Brandstädt, F.F. Dragan, H.-O. Le, V.B. Le, and R. Uehara, Tree spanners for bipartite graphs and probe interval graphs, *Algorithmica,* 47 (2007), 27–51.
12. A. Brandstädt, V. Bang Le, J.P. Spinrad, *Graph Classes: A Survey*, SIAM Monographs on Discrete Mathematics and Applications. Philadelphia, 1999.
13. A. Buneman, A characterization of rigid circuit graphs, *Discrete Math.*, 9 (1974), 205-212.
14. L. Cai and D.G. Corneil, Tree spanners, *SIAM J. Discrete. Math.*, 8 (1995), 359–387.
15. V. Chepoi and F. Dragan, A note on distance approximating trees in graphs, *Europ. J. Combin.*, 21 (2000), 761–766.
16. V.D. Chepoi, F.F. Dragan, B. Estellon, M. Habib, Y. Vaxes and Y. Xiang, Additive Spanners and Distance and Routing Labeling Schemes for δ-Hyperbolic Graphs, *Algorithmica*, DOI: 10.1007/s00453-010-9478-x, 2010.

17. V. Chepoi, F.F. Dragan, B. Estellon, M. Habib and Y. Vaxès, Diameters, centers, and approximating trees of δ-hyperbolic geodesic spaces and graphs, In *SoCG* 2008, pp. 59–68.
18. V.D. Chepoi, F.F. Dragan, I. Newman, Y. Rabinovich, Y. Vaxes, Constant Approximation Algorithms for Embedding Graph Metrics into Trees and Outerplanar Graphs, *13th Intl. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2010)*, Barcelona, Spain, 1-3 September, 2010, Springer, Lecture Notes in Computer Science 6302, pp. 95–109.
19. V.D. Chepoi, F.F. Dragan and C. Yan, Additive Sparse Spanners for Graphs with Bounded Length of Largest Induced Cycle, *Theoretical Computer Science*, 347 (2005), 54–75.
20. M.J. Demmer and M. Herlihy, The arrow distributed directory protocol, In *Proceedings of the 12th International Symposium on Distributed Computing (DISC)*, 1998, Springer-Verlag, New York, pp. 119–133.
21. R. Diestel, *Graph Theory,* second edition, Graduate Texts in Mathematics, vol. 173, Springer, 2000.
22. Y. Dourisboure, F.F. Dragan, C. Gavoille, and C. Yan, Spanners for bounded tree-length graphs, *Theor. Comput. Sci.*, 383 (2007), 34–44.
23. Y. Dourisboure, C. Gavoille, Tree-decompositions with bags of small diameter, *Discrete Mathematics,* 307 (2007), 2008-2029.
24. F.F. Dragan, F. Fomin and P. Golovach, Spanners in sparse graphs, *Journal of Computer and System Sciences,* 2010 (available on-line at DOI:10.1016/j.jcss.2010.10.002)
25. M. Elkin, Y. Emek, D.A. Spielman, S.-H. Teng, Lower-Stretch Spanning Trees, *SIAM J. Comput.* 38 (2008), 608–628.
26. M. Elkin, D. Peleg, Approximating k-spanner problems for $k \geq 2$, *Theor. Comput. Sci.* 337 (2005), 249–277.
27. Y. Emek and D. Peleg, Approximating minimum max-stretch spanning trees on unweighted graphs, *SIAM J. Comput.*, 38 (2008), 1761–1781.
28. S. P. Fekete and J. Kremer, Tree spanners in planar graphs, *Discrete Appl. Math.,* 108 (2001), pp. 85–103.
29. C. Gavoille, M. Katz, N.A. Katz, C. Paul, and D. Peleg, Approximate distance labeling schemes, *Proceedings of the 9th Annual European Symposium on Algorithms (ESA 2001)*, Aarhus, Denmark, August 28-31, 2001, Lecture Notes in Computer Science 2161, Springer, pp. 476–488.
30. F. Gavril, The intersection graphs of subtrees in trees are exactly the chordal graphs, *J. Comb. Theory (B)*, 16 (1974), 47-56.
31. J.R. Gilbert, D.J. Rose, A. Edenbrandt, A separator theorem for chordal graphs, *SIAM J. Algebraic Discrete Methods*, 5 (1984), 306–313.
32. A.J. Goldman, Optimal center location in simple networks, *Transportation Science*, 5 (1971), 212–221.
33. M. Herlihy, F. Kuhn, S. Tirthapura, and R. Wattenhofer, Dynamic analysis of the arrow distributed protocol, *Theory Comput. Syst.*, 39 (2006), 875–901.
34. D. Kratsch, H.-O. Le, H. Müller, E. Prisner and D. Wagner, Additive tree spanners, *SIAM J. Discrete Math.*, 17 (2003), 332–340.
35. C. Liebchen and G. Wünsch, The zoo of tree spanner problems, *Discrete Appl. Math.,* 156 (2008), 569–587.
36. D. Lokshtanov, On the complexity of computing tree-length, *Discrete Appl. Math.*, 158 (2010), 820–827.
37. D. Peleg, Low stretch spanning trees, In *Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science (MFCS 2002)*, Lecture Notes in Computer Science 2420, Springer, 2002, pp. 68–80.
38. D. Peleg and E. Reshef, Low complexity variants of the arrow distributed directory, *J. Comput. System Sci.*, 63 (2001), 474–485.
39. D. Peleg and D. Tendler, Low stretch spanning trees for planar graphs, *Tech. Report MCS01-14,* Weizmann Science Press of Israel, Israel, 2001.
40. J. A. Makowsky and U. Rotics, Optimal spanners in partial k-trees, *manuscript.*
41. N. Robertson, P.D. Seymour, Graph minors. II. Algorithmic aspects of tree-width, *Journal of Algorithms*, 7 (1986), 309–322.
42. M. Thorup and U. Zwick, Compact routing schemes, In *Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA 2001)*, July 4-6, 2001, Heraklion, Crete, Greece, ACM, pp. 1–10.
43. J.R. Walter, Representations of Rigid Cycle Graphs, Ph.D. *Wayne State Univ., Detroit* (1972)