APPROXIMATING DISTANCES IN COMPLICATED GRAPHS BY DISTANCES IN SIMPLE GRAPHS WITH APPLICATIONS

A dissertation submitted to
Kent State University in partial
fulfillment of the requirements for the
degree of Doctor of Philosophy

by

Chenyu Yan

August, 2007

Dissertation written by

Chenyu Yan

B.S., LanZhou University, 1995

M.A., Kent State University, 2003

M.S., Kent State University, 2004

Ph.D., Kent State University, 2007

Approved by

Dr. Feodor F. Dragan , Chair, Doctoral Dissertation Committee

Dr. Richard M. Aron , Members, Doctoral Dissertation Committee

Dr. Ruoming Jin ,

Dr. Hassan Peyravi ,

Dr. Brett Ellman ,

Accepted by

Dr. Robert A. Walker , Chair, Department of Computer Science

Dr. John R.D. Stalvey , Dean, College of Arts and Sciences

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Feodor F. Dragan for his support and help. This dissertation would not be possible without his help. I also would like to thank all the members in my dissertation committee for their comments and suggestions.

CHAPTER 1

INTRODUCTION

A *directed graph* G is a pair (V, E), where V is a finite set and E is a binary relation on V. The set V is called the vertex set of G, and its elements are called vertices. The set E is called the edge set of G, and its elements are called edges. Note that self-loops - edges from a vertex to itself - are possible. In an undirected graph G, the edge set E consists of unordered pairs of vertices, rather than ordered pairs. That is, an edge is a set $\{u, v\}$, where $u, v \in V$ and $u \neq v$. By convention, we use the notation (u, v) for an edge, rather than the set notation $\{u, v\}$, and $(u, v), (v, u)$ are considered to be the same edge. In an undirected graph, self-loops are forbidden, and so every edge consists of exactly two distinct vertices. For convenience, we use $V(G), E(G)$ to denote the vertex set and edge set of a graph G. A graph is given in Figure 1.1. Its vertex set and edge set are: $V(G) = \{1, 2, 3, 4, 5\}$, $E(G) = \{\{1, 2\}, \{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{4, 5\}\}$ Two vertices u and v of a graph are said to be adjacent if the set $\{u, v\}$ is an edge, and non-adjacent otherwise. If $e = \{u, v\}$ is an edge then the vertices u and v are its endpoints, or ends, and we shall say that the edge e connects u and v. This edge is also denoted by uv. Two edges are said to be adjacent if they have a common endpoint. A vertex v and an edge e are incident (to each other) if v is an endpoint of e (i.e., $e = \{u, v\}$), and non-incident otherwise. We say a graph has multiple edges if there exists two vertices u and v in $V(G)$ such that more than one edge in $E(G)$ incident to both u and v. That is there are at least two distinct edges, $e = \{u, v\}, e' = \{u, v\}$.

A *path* of length l from a vertex u to a vertex u' in a graph $G = (V, E)$ is a sequence $P = (v_0, v_1, v_2, \ldots, v_l)$ of vertices such that $u = v_0, u' = v_l$, and for $i = 1, 2, \ldots, l$. We

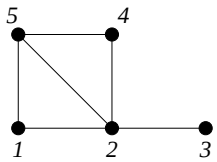


Figure 1.1: A graph with five vertices and six edges

say the path P is induced if $(v_i, v_j) \in E(G)$ if and only if $|i - j| = 1$, where j is an integer with $0 \leq j \leq l$. In Figure 1.1, (1, 2, 4, 5) is a path but not an induced path, but (3, 2, 5) is an induced path. An undirected graph is connected if every pair of vertices are connected by a path. A cycle of length $l + 1$ is a sequence $C = (v_0, v_1, v_2, \ldots, v_l, v_0)$ of vertices such that, for $i = 1, 2, \ldots, l$, $v_{i-1}v_i \in E(G)$ and $v_0v_l \in E(G)$. We say the cycle C is induced if for i and j differing by more than 1 **mod** $l + 1$, then $v_iv_j \notin E(G)$. In Figure 1.2, the cycle (1, 2, 3, 4, 1) is not an induced cycle, but the cycle (2, 3, 4, 5, 6, 7, 2) is. A graph G is a tree if it contains no cycle.

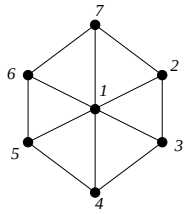


Figure 1.2: A wheel graph

A graph $G = (V, E)$ is *weighted* if each edge of G has an associated *weight*, typically given by a *weight function* $\mathsf{w} : E \longrightarrow R$. Given a weighted graph $G = (V, E)$ and two vertices $u, v \in V$, the distance between u and v is defined to be the total weight of the shortest path between u and v in G. For an unweighted graph, the distance is defined to be the number of edges on the shortest path between u and v in G. By convention, we use $d_G(u, v)$ to denote the distance between u and v in G. Many combinatorial and

algorithmic problems are concerned with the distance d_G on the vertices of a possibly weighted graph $G = (V, E)$. Approximating d_G by a simpler distance (in particular, by tree-distance d_T) is useful in many areas such as communication networks, data analysis, motion planning, image processing, network design, and phylogenetic analysis (see [4, 12, 15, 27, 33, 85, 97, 98, 103, 106]). An arbitrary metric space (in particular a finite metric defined by a general graph) might not have enough structures to exploit algorithmically; on trees, since they have a simpler (acyclic) structure, many hard algorithmic problems have easy solutions. So, the general goal is, for a given graph G, to find a simpler (well-structured, sparse, etc.) graph $H = (V, E')$ with the same vertex-set such that the distance $d_H(u, v)$ in H between two vertices $u, v \in V$ is reasonably close to the corresponding distance $d_G(u, v)$ in the original graph G.

There are several ways to measure the quality of this approximation, two of them leading to the notion of a spanner. For $t \geq 1$, a spanning subgraph H of G is called a *multiplicative t-spanner* of G [33, 98, 97], if $d_H(u, v) \leq t \cdot d_G(u, v)$ for all $u, v \in V$. If $r \geq 0$ and $d_H(u, v) \leq d_G(u, v) + r$ for all $u, v \in V$, then H is called an *additive r-spanner* of G [85]. The parameters t and r are called, respectively, the *multiplicative* and the *additive* stretch factors. Clearly, every additive r-spanner of G is a multiplicative $(r+1)$-spanner of G (but not vice versa). Note that the graphs considered in this dissertation are assumed to be unweighted.

Graph spanners have applications in various areas, especially in distributed systems and communication networks. In [98], close relationships were established between the quality of spanners (in terms of stretch factor and the number of spanner edges $|E'|$), and the time and communication complexities of any synchronizer for the network based on this spanner. Also, sparse spanners are very useful in message routing in communication networks; in order to maintain succinct routing tables, efficient routing schemes can use only the edges of a sparse spanner [99]. Unfortunately, the problem of determining, for a

given graph G and two integers $t \geq 2, m \geq 1$, whether G has a multiplicative t-spanner with m or fewer edges, is NP-complete (see [97]).

The sparsest spanners are tree spanners. Tree spanners occur in biology [9], and as it was shown in [96], they can be used as models for broadcast operations in communication networks. Tree spanners are favored also from the algorithmic point of view - many algorithmic problems are easily solvable on trees. Multiplicative tree t-spanners were studied in [27]. It was shown that, for a given graph G, the problem to decide whether G has a multiplicative tree t-spanner (the multiplicative tree t-spanner problem) is NP-complete for any fixed $t \geq 4$ and is linearly solvable for t= 1, 2. Recently, this NP-completeness result was improved-the multiplicative tree t-spanner problem is NP-complete for any fixed $t \geq 4$ even on some rather restricted graph classes: planar graphs [18], chordal graphs [21] and chordal bipartite graphs [22].

Nevertheless, some particular graph classes, such as cographs, complements of bipartite graphs, split graphs, regular bipartite graphs, interval graphs, permutation graphs, convex bipartite graphs, distance-hereditary graphs, directed path graphs, cocomparability graphs, AT-free graphs, strongly chordal graphs, and dually chordal graphs do admit additive tree r-spanners and/or multiplicative tree t-spanners for sufficiently small r and t (see [19, 26, 84, 93, 100, 101, 109]). We refer also to [4, 18, 21, 26, 27, 61, 85, 96, 97, 104] for more background information on tree and general sparse spanners.

Many graph classes (including hypercubes, planar graphs, chordal graphs, chordal bipartite graphs) do not admit any good tree spanner. For every fixed integer t there are planar chordal graphs and planar chordal bipartite graphs that do not admit tree t-spanners (additive as well as multiplicative) [32, 101]. However, as it was shown in [97], any chordal graph with n vertices admits a multiplicative 5-spanner with at most $2n-2$ edges and a multiplicative 3-spanner with at most $O(n \log n)$ edges (both spanners are constructable in polynomial time). Note also that [98] gives a method for constructing

a multiplicative 3-spanner of the n-vertex hypercube with fewer than $7n$ edges and this construction was improved in [54] to give a multiplicative 3-spanner of the n-vertex hypercube with fewer than $4n$ edges.

In this work, we investigate how to use *simple graphs* to approximate *complicated graphs* in distance. In particular, we will present our results on graph spanners, collective tree spanners and distance approximating trees.

In Chapter 2, we will show how to find sparse spanners with small additive stretch factors in chordal graphs and their generalizations. Recall that a graph G is chordal if its largest induced (chordless) cycles are of length 3. A graph is ξ-chordal if its largest induced cycles are of length ξ. The class of chordal graphs does not admit good tree spanners. As it was mentioned in [100, 101], Le and McKee have independently showed that for every fixed integer t there is a chordal graph without tree t-spanners (additive, as well as multiplicative). Recently, Brandstädt et al. [21] have showed that, for any $t \geq 4$, the problem to decide whether a given chordal graph G admits a multiplicative tree t-spanner is NP-complete even when G has the diameter at most $t+1$ (t is even), respectively, at most $t+2$ (t is odd). Thus, the only hope for chordal graphs is to get sparse (with $O(n)$ edges) small stretch factor spanners. Peleg and Schäffer have already showed in [97] that any chordal graph admits a multiplicative 5-spanner with at most $2n-2$ edges and a multiplicative 3-spanner with at most $O(n \log n)$ edges. Both spanners can be constructed in polynomial time. We will show that these results can be improved. Specifically, it can be proved that every chordal graph admits an additive 4-spanner with at most $2n-2$ edges and an additive 3-spanner with at most $O(n \log n)$ edges. An additive 4-spanner can be constructed in linear time while an additive 3-spanner is constructable in $O(m \log n)$ time, where m is the number of edges of G. Even more, the method designed for chordal graphs is extended to all ξ-chordal graphs. As a result, it was shown that any such graph admits an additive $(\xi+1)$-spanner with

at most $2n-2$ edges which is constructable in $O(\xi \cdot n + m)$ time. Note that the method from [97] essentially uses the characteristic clique trees of chordal graphs and therefore cannot be extended (at least directly) to general ξ-chordal graphs for $\xi \geq 4$.

One of the applications of spanners is routing messages in wireless and sensor networks [5]. Usually, we use unit disk graphs (UDG) to model a wireless and sensor network. Since it is very difficult to route messages in UDG, many researchers designed different spanners (topologies) and routing schemes for UDG. Spanners such as *Relative Neighborhood Graph (RNG), Gabriel Graph (GG)* etc. can significantly simplify the routing task in wireless and sensor networks. However, they also have some weak points. For example, the communication delay can be huge. Some routing schemes such as greedy and compass can not even guarantee the delivery of messages. To address this issue, we put forth a new concept-*collective tree spanners*. Before introducing collective tree spanners, let us take a look at the tree spanner problem. Tree spanner was first defined and studied by Cai and Corneil [27]. The definition of tree spanners is very similar to the definition of graph spanners except the requirement that H must be a spanning tree of G. Collective tree spanners are a generalization of tree spanners. Given a graph G and integers μ, r and t, if G admits a system of spanning trees such that for any two vertices u, v of G, there is a spanning tree T_i in the system with $d_{T_i}(u,v) \leq d_G(u,v) + r$ (or $d_{T_i}(u,v) \leq t \cdot d_G(u,v)$), then we say G admits a system of μ collective additive (multiplicative) tree r-spanners (t-spanners). Similarly, we say that a graph G admits a system of μ collective tree (t,r)-spanners, if, for any two vertices u, v of G, there is a spanning tree T_i in the system such that $d_{T_i}(u,v) \leq t \cdot d_G(u,v) + r$ holds. Unlike the tee spanners, we are trying to use several spanning trees to approximate the original graph distances. One of the applications of collective tree spanners is routing messages in communication networks which is usually modeled as a graph. Peleg, Upfal [99] showed that it is very difficult to design good routing schemes for general graphs.

In contrast, Thorup and Zwick [108] proved that every tree admits a very good routing scheme. Therefore, if one can use small number of spanning trees to approximate the original graph G's distances, then the messages can be routed in one of the spanning trees instead of in the original graph. Another reason for us to study the collective tree spanners problem is because to find one good spanning tree even for some special graph classes, like chordal and planar graphs, is difficult. Moreover, it is known that for any integer t, there is a chordal graph which does not have a multiplicative (as well as an additive) tree t-spanner. In contrast, we were able to prove that every chordal graph admits a system of $O(\log n)$ collective additive tree 2-spanners. In Chapter 3, we will present some results on collective tree spanners of special classes of graphs such as chordal, AT-free, bounded tree-width and UDG. As discussed above, UDG is often used to model wireless and sensor networks. We were able to show that each UDG admits a system of $O(\log n)$ collective tree (3, 4)-spanners. This implies that UDG admits a routing scheme with constant stretch factor. According to what we know, this is the first routing scheme which can achieve constant routing stretch factor, and this can significantly reduce the communication delay between two sensors and significantly prolong the lifetime of wireless and sensor networks. It is also worth to mention that recently, Gupta et. al.[28] designed a very good routing scheme for metric spaces with bounded doubling dimension. This routing scheme works well when the doubling dimension of the metric space is bounded. However, as was shown by Xiang [110], the doubling dimension of UDG is unbounded. Therefore, the routing scheme designed by Gupta et. al. [28] is not applicable to UDG.

In tree and collective tree spanners, the underlying spanning sub-graphs must use the edges in the original graph. For many applications (e.g. in numerical taxonomy or in phylogeny reconstruction), this requirement can be dropped [12, 103, 106]. In this case there is a striking way to measure how sharp d_H approximates d_G, based

on the notion of a pseudoisometry between two metric spaces. This idea is borrowed from the geometry of hyperbolic groups [57, 69]. For graphs and finite metric spaces a related notion of a near-isometry has been already used by Linial et. al. [86]. This motivated the authors of [19] to define another problem - distance approximating tree problem. In this problem, the tree can use edges which do not exist in the original graph, but with restrictions. Formally, the problem is defined as follows: given a graph $G = (V, E)$ and integers δ and Δ, a tree $T = (V, E')$ is a distance (Δ, δ)-approximating tree of $G = (V, E)$ if $d_T(u, v) \leq \Delta \cdot d_G(u, v) + \delta$ and $d_G(u, v) \leq \Delta \cdot d_T(u, v) + \delta$, for all $u, v \in V$, hold. A tree $T = (V, E')$ is a distance $(1, \delta)$-approximating tree of $G = (V, E)$ (or, simply, a distance δ-approximating tree of G) if $|d_G(u, v) - d_T(u, v)| \leq \delta$ for all $u, v \in V$. The distance (Δ, δ)-approximating tree problem asks for a given graph G to decide whether G has a distance (Δ, δ)-approximating tree. As mentioned before, distance approximating tree problem was motivated by its application in numerical taxonomy and phylogeny reconstruction. It is also known that the class of chordal graphs does not admit any good tree t-spanners for any integer t. Furthermore, as shown by Brandstädt et al. [21], it is very hard to find good tree t-spanners even for chordal graphs. On the other hand, in [15], Brandstädt et al. proved that every chordal graph G admits a tree $T(G)$(constructable in linear time) which is both a (3, 0)- and a (1, 2)-approximating tree of G. So, from the metric point of view chordal graphs do look like trees, but the notion of tree spanners failed to capture this. Note that the result is optimal in the sense that there are chordal graphs which do not admit any distance (1, 1)-approximating trees [30].

The result was used in [19, 30, 74] to provide efficient approximate solutions for several problems on chordal graphs. It is known that the (exact) distance matrix $D(G)$ of a chordal graph $G = (V, E)$ cannot be computed in less than "matrix multiplication"

time. Using a distance (1, 2)-approximating tree $T(G)$ of G, after a linear time preprocessing of G (and then of $T(G)$), in only $O(1)$ time, one can compute $d_G(x, y)$ with an error of at most 2 for any $x, y \in V$ (see [19] for further details). As another application, consider the p-center problem: given a graph G (or, more generally, a metric space) and an integer $p > 0$, we are searching for the smallest radius r^* and a subset of vertices X of G with $|X| \leq p$ such that $d_G(v, X) \leq r^*$ for every vertex v of G. The problem is NP-hard even for chordal graphs. Solving the p-center problem on a distance (1, 2)-approximating tree $T(G)$ of G (on trees this problem is polynomial time solvable [76]), we will find an optimal covering radius r' of $T(G)$ and a set of centers Y with $|Y| \leq p$. Then, Y can be taken as an approximate solution for G since $d_G(v, Y) \leq r' + 2 \leq r^* + 4$ for all $v \in V$ (see [30] for further details). Clearly, similar results can be obtained for any graph admitting a good distance approximating tree.

The result was also used by Gupta in [74] for bandwidth approximation in chordal graphs. If a graph G has a distance (Δ, δ)-approximating tree $T(G)$ for some constants Δ and δ, then the bandwidth of a linear arrangement of G will be within some constant of the bandwidth of the same arrangement for $T(G)$. Gupta developed in [74] a simple randomized $O(\log^{2.5} n)$-approximation algorithm for bandwidth minimization on trees and used it to get an approximation algorithm with a similar performance guarantee for chordal graphs (see [74]for further details).

In [83], Krauthgamer et al. used the existence of good distance approximating trees for chordal graphs to obtain an embedding of any chordal graph into l_2 with a small r-dimensional volume distortion.

Later, in [30], Chepoi and Dragan extended the method of [19] from chordal graphs to all ξ-chordal graphs. A graph G is said to be ξ-chordal if no induced cycle of G has more than ξ edges. It was proven that, for every ξ-chordal graph $G = (V, E)$, there exists a tree $T = (V, F)$ (constructable in linear time) such that $|d_G(u, v) - d_T(u, v)| \leq \lfloor \frac{\xi}{2} \rfloor + \alpha$

for all vertices $u, v \in V$, where $\alpha = 1$ if $\xi \neq 4, 5$ and $\alpha = 2$ otherwise. Clearly, this result can be used to provide efficient approximate solutions for several problems on ξ-chordal graphs. Here, we will mention only one implication provided in [82]. Krauthgamer and Lee, in [82], proved first that the Levin's conjecture on intrinsic dimensionality of graphs holds for trees. Then, relying on low-distortion embeddings of ξ-chordal graphs into trees, due to [30], they extended that result to all ξ-chordal graphs: the Levin's conjecture on intrinsic dimensionality of graphs holds for all ξ-chordal graphs with bounded ξ (see [82] for further details).

Motivated by those applications of distance approximating trees, in Chapter 4, we investigate the complexity of finding a good distance (Δ, δ)-approximating tree (for small Δ and δ) for a given graph G.We prove that the distance $(\Delta, 0)$-approximating tree problem is NP-complete for any $\Delta \geq 5$ and the distance (1, 1)-approximating tree problem is polynomial time solvable. We reduce 3SAT to our problem. The reduction is very complicated. In the proof, we need to use several gadgets such as flower and wing gadgets. On the positive side, we will show that to decide whether a general graph admits a distance (1, 1)-approximating tree can be decided in polynomial time and the algorithm will construct such a tree if it exists.

All graphs occurring in this dissertation are connected, finite, undirected, loopless, and without multiple edges. For each integer $l \geq 0$, let $B_l(u)$ denote the *ball* of radius l centered at u:

$$B_l(u) = \{v \in V : d_G(u, v) \leq l\}.$$

Let $N_l(u)$ denote the *sphere* of radius l centered at u:

$$N_l(u) = \{v \in V : d_G(u, v) = l\}.$$

$N_l(u)$ is called also the *lth neighborhood* of u. A *layering* of G with respect to some vertex u is a partition of V into the spheres $N_l(u)$, $l = 0, 1, \ldots$. By $N(u)$ we denote

the *neighborhood* of u, i.e., $N(u) = N_1(u)$. More generally, for a subset $S \subseteq V$ let $N(S) = \bigcup_{u \in S} N(u)$.

CHAPTER 2

GRAPH SPANNERS

2.1 Introduction

In this chapter we are interested in finding sparse spanners with small additive stretch factors in chordal graphs and their generalizations. A graph G is chordal if its largest induced (chordless) cycles are of length 3. A graph is ξ-chordal if its largest induced cycles are of length ξ.

The class of chordal graphs does not admit good tree spanners. As mentioned before, H.-O. Le and T.A. McKee [81, 95] have independently shown that for every fixed integer t there is a chordal graph without tree t-spanners (additive as well as multiplicative). Recently, Brandstädt et al. [21] have shown that, for any $t \geq 4$, the problem to decide whether a given chordal graph G admits a multiplicative tree t-spanner is NP-complete even when G has the diameter at most $t+1$ (t is even), respectively, at most $t+2$ (t is odd). Thus, the only hope for chordal graphs is to get sparse (with $O(n)$ edges) small stretch factor spanners. Peleg and Schäffer have already showed in [97] that any chordal graph admits a multiplicative 5-spanner with at most $2n-2$ edges and a multiplicative 3-spanner with at most $O(n \log n)$ edges. Both spanners can be constructed in polynomial time.

In this chapter, we improve those results. We show that every n-vertex chordal graph admits an additive 4-spanner with at most $2n-2$ edges and an additive 3-spanner with at most $O(n \log n)$ edges. Both spanners can be constructed in polynomial time. Our spanners are not only additive but also easier to construct. An additive 4-spanner can be constructed in linear time while an additive 3-spanner can be constructed in $O(m \log n)$

time where m is the number of edges of G. Furthermore, our method can be extended to all ξ-chordal graphs. Any such graph admits an additive $(\xi+1)$-spanner with at most $2n-2$ edges which can be constructed in $O(n \cdot \xi + m)$ time. Results of this chapter were published in [31, 32].

2.1.1 Basic notions and notations

Let $\sigma = [v_1, v_2, \ldots, v_n]$ be any ordering of the vertex set of a graph G. We will write $a < b$ whenever in a given ordering σ vertex a has a smaller number than vertex b. Moreover, $\{a_1, \cdots, a_l\} < \{b_1, \cdots, b_k\}$ is an abbreviation for $a_i < b_j$ $(i = 1, \cdots, l;$ $j = 1, \cdots, k)$. In this chapter, we will use two kinds of orderings, namely, BFS-orderings and LexBFS-orderings.

In a *breadth-first search (BFS)*, started at vertex u, the vertices of a graph G with n vertices are numbered from n to 1 in decreasing order. The vertex u is numbered by n and put on an initially empty queue of vertices. Then a vertex v at the head of the queue is repeatedly removed, and neighbors of v that are still unnumbered are consequently numbered and placed onto the queue. Clearly, BFS operates by proceeding vertices in layers: the vertices closest to the start vertex are numbered first, and most distant vertices are numbered last. BFS may be seen to generate a rooted tree T with vertex u as the root. We call T the *BFS-tree* of G. A vertex v is the *father* in T of exactly those neighbors in G which are inserted into the queue when v is removed. An ordering σ generated by a BFS will be called a *BFS-ordering* of G. Denote by $f(v)$ the father of a vertex v with respect to σ. The following properties of a BFS-ordering will be used in what follows.

(P1) If $x \in N_i(u), y \in N_j(u)$ and $i < j$, then $x > y$ in σ.

(P2) If $v \in N_q(u)(q > 0)$ then $f(v) \in N_{q-1}(u)$ and $f(v)$ is the vertex from $N(v)$ with

the largest number in σ.

(P3) If $x > y$, then either $f(x) > f(y)$ or $f(x) = f(y)$.

Lexicographic breadth-first search (LexBFS), started at a vertex u, orders the vertices of a graph by assigning numbers from n to 1 in the following way. The vertex u gets the number n. Then each next available number k is assigned to a vertex v (as yet unnumbered) which has lexically largest vector $(s_n, s_{n-1}, \ldots, s_{k+1})$, where $s_i = 1$ if v is adjacent to the vertex numbered i, and $s_i = 0$ otherwise. An ordering of the vertex set of a graph generated by LexBFS we will call a *LexBFS-ordering*. Clearly any LexBFS-ordering is a BFS-ordering (but not conversely). Note also that for a given graph G, both a BFS-ordering and a LexBFS-ordering can be generated in linear time [72]. Besides the properties of BFS-ordering, LexBFS-ordering has one more property.

(P4) If $a < b < c$ and $ac \in E$ and $bc \notin E$ then there exists a vertex d such that $c < d, db \in E$ and $da \notin E$.

In particular, we can associate a tree T rooted at v_n with every LexBFS-ordering $\sigma = [v_1, v_2, \ldots, v_n]$ simply connecting every vertex v ($v \neq v_n$) to its neighbor $f(v)$ with the largest number in σ. We call this tree a *LexBFS-tree* of G rooted at v_n and vertex $f(v)$ *the father* of v in T.

2.2 Additive 4-spanners with $O(n)$ edges

For a chordal graph $G = (V, E)$ and a vertex $u \in V$, consider a BFS of G started at u and let $q = max\{d_G(u, v) : v \in V\}$. For a given k, $0 \leq k \leq q$, let S_1^k, S_2^k,..., $S_{p_k}^k$ be the connected components of a subgraph of G induced by the kth neighborhood of u. In [19], there was defined a graph Γ whose vertices are the connected components S_i^k, $k = 0, 1, \ldots, q$ and $i = 1, \ldots, p_k$. Two vertices S_i^k, S_j^{k-1} are adjacent if and only if there is an edge of G with one end in S_i^k and another end in S_j^{k-1}. Before we describe

our construction of the additive 4-spanner $H = (V, E')$ for a chordal graph G, first we recall two important lemmas.

Lemma 1 *[19] Let G be a chordal graph. For any connected component S of the subgraph of G induced by $N_k(u)$, the set $N(S) \cap N_{k-1}(u)$ induces a complete subgraph.*

Lemma 2 *[19] Γ is a tree.*

Now, to construct H, we choose an arbitrary vertex $u \in V$ and perform a Breadth-First-Search in G started at u. Let $\sigma = [v_1, \ldots, v_n]$ be a BFS-ordering of G. The construction of H is completed according to the following algorithm (for an illustration see Figure 2.1).

PROCEDURE 1. Additive 4-spanners for chordal graphs

Input: A chordal graph $G = (V, E)$ with BFS-ordering σ, and connected components $S_1^k, S_2^k, \ldots, S_{p_k}^k$ for any k, $0 \leq k \leq q$, where $q = max\{d_G(u, v) : v \in V\}$.

Output: A spanner $H = (V, E')$ of G.

Method:

$E' = \emptyset$;
for $k = q$ **downto** 1 **do**
  for $j = 1$ **to** p_k **do**
    $M = \emptyset$;
    for each vertex $v \in S_j^k$ add edge $vf(v)$ to E' and vertex $f(v)$ to M;
    pick vertex $c \in M$ with the minimum number in σ;
    for every vertex $x \in M \setminus \{c\}$ add edge xc to E';
return $H = (V, E')$.

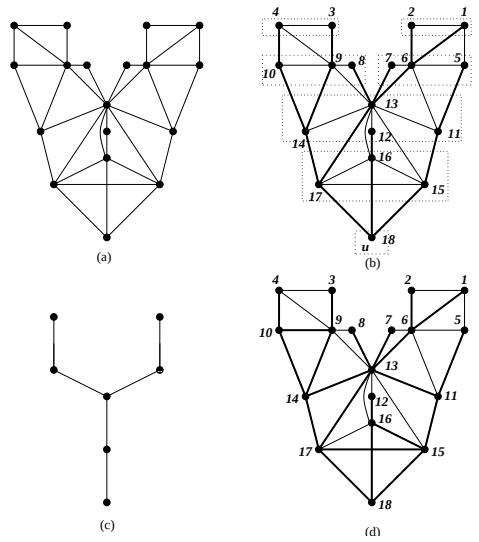


Figure 2.1: (a) A chordal graph G. (b) A BFS-ordering σ, BFS-tree T associated with σ and a layering of G. (c) The tree Γ of G associated with that layering. (d) Additive 4-spanner (actually, additive 3-spanner) H of G constructed by PROCEDURE 1 (5 edges are added to the BFS-tree T).

Lemma 3 *H is an additive 4-spanner for G.*

Proof Consider nodes S_i^l and S_j^m of the tree Γ and their lowest common ancestor S_m^p in Γ. For any two vertices $x \in S_i^l$ and $y \in S_j^m$ of G, we have

$$d_G(x, y) \geq l - p + m - p,$$

since any path of G connecting x and y must pass S_m^p.

From our construction of H (for every vertex v of G the edge $vf(v)$ is present in H), we can easily show that there exist vertices $x', y' \in S_m^p$ such that

$$d_H(x, x') = l - p,$$

$$d_H(y, y') = m - p.$$

Hence we only need to show that

$$d_H(x', y') \leq 4.$$

If $x' = y'$ then we are done. If vertices x' and y' are distinct, then by Lemma 1, $N(S_m^p) \cap N_{p-1}(u)$ is a clique of G. According to Procedure 1, fathers of both vertices x' and y' are in M and they are connected in H by a path of length at most 2 via vertex c of M. Therefore, $d_H(x', y') \leq d_H(x', f(x')) + d_H(f(x'), f(y')) + d_H(f(y'), y') \leq 1+2+1 = 4$. This concludes our proof. □

We can easily show that the bound given in Lemma 3 is tight. For a chordal graph presented in Figure 2.2, we have $d_G(y, b) = 1$. The spanner H of G constructed by our method is shown with bold edges. In H we have $d_H(y, b) = 5$. Therefore, $d_H(y, b) - d_G(y, b) = 4$.

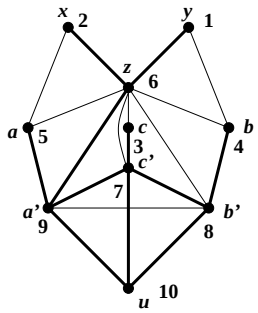


Figure 2.2: A chordal graph with a BFS-ordering which shows that the bound given in Lemma 3 is tight. We have $d_H(y, b) - d_G(y, b) = 4$ and $d_H(y, b)/d_G(y, b) = 5$.

Lemma 4 *If G has n vertices, then H contains at most $2n - 2$ edges.*

Proof The edge set of H consists of two sets E_1 and E_2, where E_1 are those edges connecting two vertices between two different layers (edges of type $vf(v)$) and E_2 are those edges which have been used to build a star for a clique M inside a layer (edges of type $cf(v)$). Obviously, E_1 has exactly $n - 1$ edges; actually, they are the edges of the BFS-tree of G. For each connected component S_i^l of size s, we have at most s vertices in M. Therefore, while proceeding component S_i^l, at most $s - 1$ edges are added to E_2.

The total size of all the connected components is at most n, so E_2 contains at most $n-1$ edges. Hence, the graph H contains at most $2n-2$ edges. □

Lemma 5 *H can be constructed in linear $O(n+m)$ time.*

Proof A BFS-tree of G can be constructed in linear time. To construct H, we only need to find the connected components of the kth neighborhood of u ($1 \leq k \leq q$) and for each component compute the set M and build a star on M. Since the size of M is not larger than the size of that connected component, all these can be done in $O(n+m)$ total time (for all ks). Hence, the construction of H can easily be done in total $O(n+m)$ time. □

Combining Lemmas 3-5 we get the following result.

Theorem 1 *Every n-vertex chordal graph $G = (V, E)$ admits an additive 4-spanner with at most $2n-2$ edges. Moreover, such a sparse spanner of G can be constructed in linear time.*

Figure 2.3 presents a chordal graph with its additive 3-spanner constructed according to PROCEDURE 1.

Notice that any additive 4-spanner is a multiplicative 5-spanner. As we mentioned earlier the existence of multiplicative 5-spanners with at most $2n-2$ edges in chordal graphs was already shown in [97], but their method of constructing such spanners is more complicated than ours and can take more than linear time.

2.2.1 Additive 3-spanners with $O(n \cdot \log n)$ edges

To construct an additive 3-spanner for a chordal graph $G = (V, E)$, first we get a LexBFS-ordering σ of the vertices of G (see Figure 2.4). Then, we construct an additive 4-spanner $H = (V, E_1 \bigcup E_2)$ for G using the algorithm from Section **??**. Finally, we

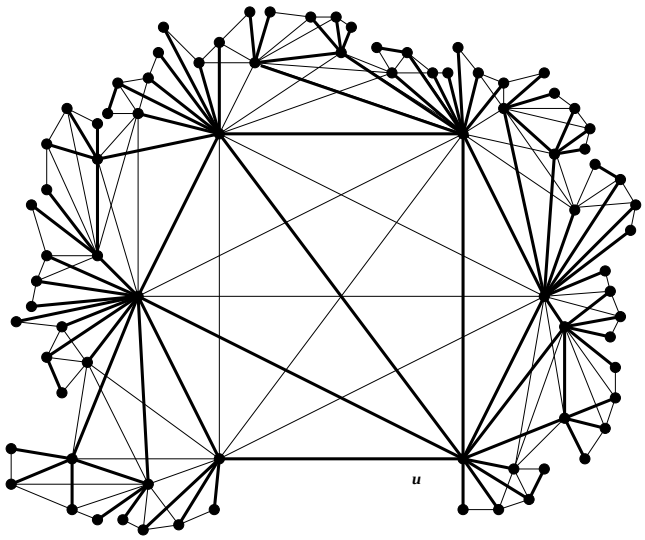


Figure 2.3: A chordal graph and its additive 3-spanner constructed by PROCEDURE 1. The spanner is shown with dark edges, it has 80 vertices and 90 edges.

update H by adding some more edges. In what follows, we will need the following known result.

Theorem 2 *[71] Every n-vertex chordal graph G 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 $n/2$.*

An $O(n+m)$ algorithm for finding such a separating clique C is also given in [71].

As before, for a given k, $0 \leq k \leq q$, let S_1^k, S_2^k,..., $S_{p_k}^k$ be the connected components of a subgraph of G induced by the kth neighborhood of u. For each connected component S_i^k (which is obviously a chordal graph), we run the following algorithm which is similar to the algorithm in [97] (see also [96]), where a method for construction of a multiplicative 3-spanner for a chordal graph is described. The only difference is that we run that algorithm on every connected component from each layer of G instead of

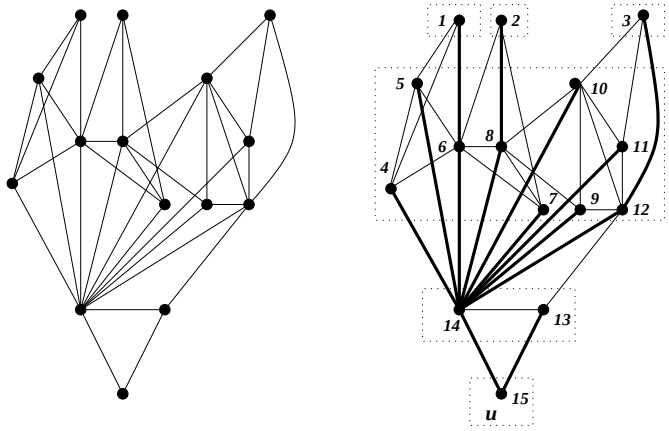


Figure 2.4: (a) A chordal graph G. (b) A LexBFS-ordering σ, LexBFS-tree associated with σ and a layering of G.

on the whole graph G. For the purpose of completeness, we present the algorithm here (for an example see Figure 2.5).

PROCEDURE 2. A balanced clique tree for a connected component S_i^k

Input: A subgraph Q of G induced by a connected component S_i^k.

Output: A balanced clique tree for Q.

Method:

find a maximum separating clique C of the graph Q
  as prescribed in Theorem 2;
suppose C partitions the rest of Q into connected
  components $\{Q_1, \ldots, Q_r\}$;
for each Q_i such that Q_i is not a clique, construct a balanced clique tree
  $T(Q_i)$ recursively;
construct $T(Q)$ by taking C to be the root and

connecting the root of each tree $T(Q_i)$ as a child of C.

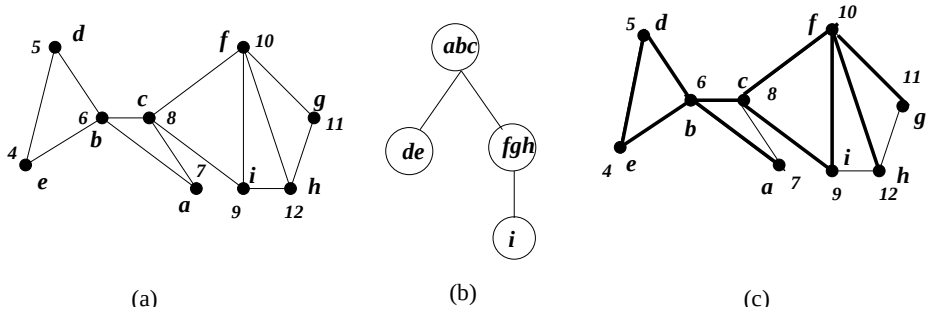


Figure 2.5: (a) A chordal graph induced by set S_1^2 of the graph G presented in Figure 2.4, (b) its balanced clique tree and (c) edges of $E_3(S_1^2) \bigcup E_4(S_1^2)$.

The nodes of the final balanced tree for S_i^k (denote it by $T(S_i^k)$) represent a certain collection of disjoint cliques $\{C_i^k(1), \ldots C_i^k(s_i^k)\}$ that cover entire set S_i^k (see Figure 2.5 for an illustration). For each clique $C_i^k(j)$ $(1 \leq j \leq s_i^k)$ we build a star centered at its vertex with the minimum number in LexBFS-ordering σ. We use $E_3(i,k)$ to denote this set of star edges. Evidently, $|E_3(i,k)| \leq |S_i^k| - 1$.

Consider a clique $C_i^k(j)$ in S_i^k. For each vertex v of $C_i^k(j)$ and each clique $C_i^k(j')$ on the path of balanced clique tree $T(S_i^k)$ connecting node $C_i^k(j)$ with the root, if v has a neighbor in $C_i^k(j')$ (i.e., there exists an edge of G between v and a vertex of $C_j^k(j')$), then select one such neighbor w and put the edge vw of G into set $E_4(i,k)$ (initially $E_4(i,k)$ is empty). We do this for every clique $C_i^k(j)$, $j \in \{1, \ldots, s_i^k\}$. Since the depth of the tree $T(S_i^k)$ is at most $log_2|S_i^k| + 1$ (see [97], [96]), any vertex v from S_i^k may contribute at most $log_2|S_i^k|$ edges to $E_4(i,k)$. Therefore, $|E_4(i,k)| \leq |S_i^k| \cdot log_2|S_i^k|$.

Define now two sets of edges in G, namely,

$$E_3 = \bigcup_{k=1}^{q} \bigcup_{i=1}^{p_k} E_3(i,k)$$

and

$$E_4 = \bigcup_{k=1}^{q} \bigcup_{i=1}^{p_k} E_4(i,k),$$

and consider a spanning subgraph $H^* = (V, E_1 \bigcup E_2 \bigcup E_3 \bigcup E_4)$ of G (see Figure 2.6). Recall that $E_1 \bigcup E_2$ is the set of edges of an additive 4-spanner H constructed for G by PROCEDURE 1 (see Section 2.2).

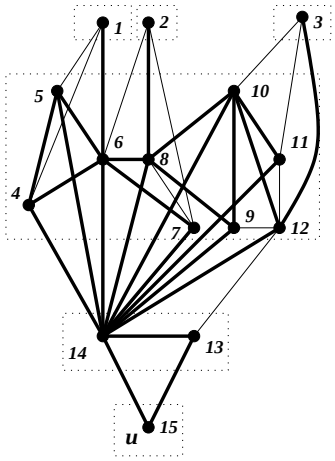


Figure 2.6: An additive 3-spanner H^* of graph G presented in Figure 2.4.

The following lemmas for H^* hold.

Lemma 6 *If G has n vertices, then H^* has at most $O(n \cdot \log n)$ edges.*

Proof We know already that $|E_1| + |E_2| \leq 2n - 2$. Also,

$$|E_3| \quad = \quad \sum_{k=1}^{q} \sum_{i=1}^{p_k} |E_3(i,k)| \leq \sum_{k=1}^{q} \sum_{i=1}^{p_k} (|S_i^k| - 1) \leq n - 1$$

and

$$|E_4| \quad = \quad \sum_{k=1}^{q} \sum_{i=1}^{p_k} |E_4(i,k)| \leq \sum_{k=1}^{q} \sum_{i=1}^{p_k} (|S_i^k| \cdot log_2 |S_i^k|) \leq n \cdot \log_2 n.$$

Hence, the total number of edges in H^* is at most $O(n \cdot \log n)$. □

To prove that H^* is an additive 3-spanner for G, we will need the following auxiliary lemmas.

Lemma 7 *[72] Let G be a chordal graph and σ be a LexBFS-ordering of G. Then, σ is a perfect elimination ordering of G, i.e., for any vertices a, b, c of G such that $a < \{b, c\}$ and $ab, ac \in E(G)$, vertices b and c must be adjacent.*

Lemma 8 *[44] Let G be an arbitrary graph and $T(G)$ be a BFS-tree of G with the root u. Let also v be a vertex of G and w $(w \neq v)$ be an ancestor of v in $T(G)$ from layer $N_i(u)$. Then, for any vertex $x \in N_i(u)$ with $d_G(v, w) = d_G(v, x)$, inequality $x \leq w$ holds.*

Lemma 9 *H^* is an additive 3-spanner for G.*

Proof As in the proof of Lemma 3, consider again nodes S_i^k and S_j^l of the tree Γ and their lowest common ancestor S_m^p in Γ. Then $d_G(x, y) \geq k - p + l - p$ holds for any two vertices $x \in S_i^k$ and $y \in S_j^l$ and there must exist vertices $x', y' \in S_m^p$ such that

$$d_{H^*}(x, x') = k - p,$$

$$d_{H^*}(y, y') = l - p.$$

So, to have $d_{H^*}(x, y) - d_G(x, y) \leq 3$, we only need to show $d_{H^*}(x', y') \leq 3$. We may assume $x' \neq y'$.

First note that, since additive 4-spanner H is a subgraph of H^*, $d_{H^*}(x', y') \leq 4$ holds (see the proof of Lemma 3). Hence, if $d_G(x, y) > k-p+l-p$, then $d_{H^*}(x, y)-d_G(x, y) \leq 3$ and we are done.

We may assume now that $d_G(x, y) = k-p+l-p$ and, therefore, there exists a vertex z in S_m^p such that $d_G(x, z) = k - p$ and $d_G(y, z) = l - p$ (z is a vertex of a shortest path connecting x and y in G). Let S_t^{p+1}, S_r^{p+1} be the two connected components on the paths of Γ between S_m^p, S_i^k and S_m^p, S_j^l, respectively. From $x', z \in N(S_t^{p+1}) \cap N_p(u)$, we conclude that x' and z either coincide or are adjacent in G (see Lemma 1). Similarly, $d_G(y', z) \leq 1$.

Since $x', y', z \in N_p(u)$, $d_G(x, x') = d_G(x, z) = k - p$, $d_G(y, y') = d_G(y, z) = l - p$, and x' is an ancestor of x and y' is an ancestor of y in the LexBFS-tree associated with σ, by Lemma 8, $z \leq \{x', y'\}$.

We claim that x' and y' are adjacent in G. Indeed, if z coincides with x' or y', then $x'y' \in E(G)$ because $d_G(z, x') \leq 1$ and $d_G(z, y') \leq 1$. If z is distinct from both x' and y', then the inequality $z < \{x', y'\}$ together with $zx', zy' \in E(G)$ imply $x'y' \in E(G)$ (by Lemma 7). Thus, $x'y' \in E(G)$

Now, if x' and y' are in one clique $C_m^p(t)$ (for some t), then, since in H^* vertices of $C_m^p(t)$ are connected by a star, $d_{H^*}(x', y') \leq 2$ must hold.

If x' and y' are in cliques $C_m^p(t)$ and $C_m^p(r)$ respectively (for some t, r), then one of these cliques is descendent of the other in the balanced clique tree $T(S_m^p)$. This is because for two cliques $C_m^p(t)$ and $C_m^p(r)$ that have no descendence relationship in the tree $T(S_m^p)$, the clique $C_m^p(h)$ that is their lowest common ancestor in the tree separates the vertices of $C_m^p(t)$ and $C_m^p(r)$, hence no edge is possible between them.

Assuming, without loss of generality, that $C_m^p(t)$ is an ancestor of $C_m^p(r)$ in the tree $T(S_m^p)$, y' is connected in H^* to some vertex x'' of $C_m^p(t)$. Since vertices of $C_m^p(t)$ are connected by a star in H^*, we get $d_{H^*}(x', y') \leq d_{H^*}(x', x'') + d_{H^*}(x'', y') \leq 2 + 1 = 3$.

Thus, in any cases we have $d_{H^*}(x', y') \leq 3$. □

Lemma 10 *If a chordal graph G has n vertices and m edges, then its additive 3-spanner H^* can be constructed in $O(m \cdot \log n)$ time.*

Proof As it was shown in Section 2.2, the additive 4-spanner can be constructed in $O(n + m)$ time. For each connected component S_i^k with $n_{i,k}$ vertices and $m_{i,k}$ edges, its balanced clique tree can be constructed in $O(m_{i,k} \cdot \log n_{i,k})$ time. To build a star on each clique $C_i^k(j)$ and find all edges of $E_4(i, k)$, we need at most $\sum_{v \in S_i^k}(deg_G(v) \cdot \log n_{i,k})$ time.

So, the total time needed to construct H^* is

$$O(n+m)+\sum_{k=1}^{q}\sum_{i=1}^{p_k}(O(m_{i,k}\cdot\log n_{i,k})+\sum_{v\in S_i^k}(deg_G(v)\cdot\log n_{i,k}))$$
$$=O(m\cdot\log n).$$

This concludes the proof. □

The main result of this subsection is the following.

Theorem 3 *Every chordal graph $G=(V,E)$ with n vertices and m edges admits an additive 3-spanner with at most $O(n\cdot\log n)$ edges. Moreover, such a sparse spanner of G can be constructed in $O(m\cdot\log n)$ time.*

In [97], it was shown that any chordal graph admits a multiplicative 3-spanner H' with at most $O(n\cdot\log n)$ edges which is constructable in $O(m\cdot\log n)$ time. But, it is worth to note that the spanner H' gives a better than H^* approximation of distances only for adjacent in G vertices. For pairs $x,y\in V$ at distance at least 2 in G, the multiplicative stretch factor given by H^* is at most 2.5 which is better than the multiplicative stretch factor of at most 3 given by H'.

2.3 Spanners for ξ-chordal Graphs

Let u be an arbitrary vertex of a ξ-chordal graph $G=(V,E)$, σ be a BFS-ordering of G and T be the BFS-tree associated with σ. For each $l\geq 0$ define a graph Q^l with the lth sphere $N_l(u)$ as a vertex set. Two vertices $x,y\in N_l(u)$ $(l\geq 1)$ are adjacent in Q^l if and only if they can be connected by a path outside the ball $B_{l-1}(u)$. Let $Q_1^l,\ldots,Q_{p_l}^l$ be all the connected components of Q^l. Similar to chordal graphs and as shown in [30] we define a graph Γ whose vertex-set is the collection of all connected components of the graphs Q^l, $l=0,1,\ldots$, and two vertices are adjacent in Γ if and only if there is an edge of G between the corresponding components. The following lemma holds.

Lemma 11 *[30]* Γ *is a tree.*

To construct our spanner H for G, we use the following procedure (for an illustration see Figure 2.7).

PROCEDURE 3. Additive $(\xi+1)$-spanners for ξ-chordal graphs

Input: A ξ-chordal graph $G=(V,E)$ with a BFS-ordering σ, and connected components Q_1^l, Q_2^l, ..., $Q_{p_l}^l$ for any l, $0 \leq l \leq q$, where $q = max\{d_G(u,v) : v \in V\}$.

Output: A spanner $H=(V,E')$ of G.

Method:

$E' = \emptyset$;
for $l = q$ **downto** 1 **do**
  for $j = 1$ **to** p_l **do**
    for each vertex $v \in Q_j^l$ add $vf(v)$ to E';
    pick vertex c in Q_j^l with the minimum number in σ;
    for each $v \in Q_j^l \setminus \{c\}$ **do**
      connected = FALSE;
      while connected = FALSE **do**
      /* this while loop works at most $\lfloor \xi/2 \rfloor$ times for each v */
        if $vc \in E(G)$ **then**
          add vc to E';
          connected = TRUE;
        else if $vf(c) \in E(G)$ **then**
            add $vf(c)$ to E';
            connected = TRUE;
          else $v = f(v)$, $c = f(c)$

return $H = (V, E')$.

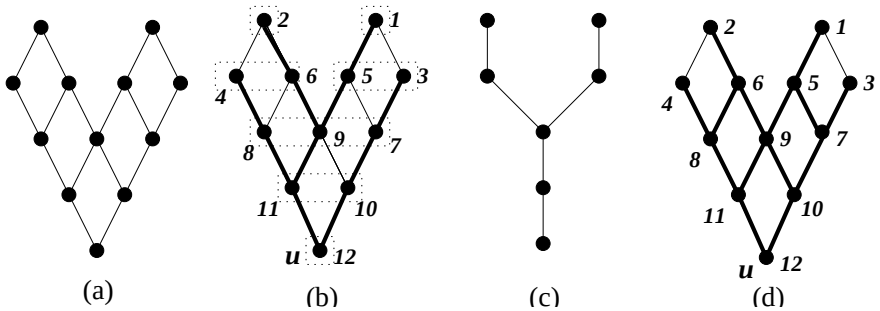


Figure 2.7: (a) A 4-chordal graph G. (b) A BFS-ordering σ, BFS-tree associated with σ and a layering of G. (c) The tree Γ of G associated with that layering. (d) Additive 5-spanner (actually, additive 2-spanner) H of G constructed by PROCEDURE 3.

Clearly, H contains all edges of BFS-tree T because for each $v \in V$ the edge $vf(v)$ is in H. For a vertex v of G, let P_v be the path of BFS-tree T connecting v with the root u. We call it *the maximum neighbor path of* v in G (evidently, P_v is a shortest path of G). Additionally to the edges of T, H contains also some bridging edges connecting vertices from different maximum neighbor paths.

Lemma 12 *Let c be vertex of Q_i^l with the minimum number in σ ($l \in \{1, \ldots, q\}$, $i \in \{1, \ldots, p_l\}$). Then, for any $a \in Q_i^l$, there is a (a, c)-path in H of length at most xi consisting of a subpath $(a, \ldots, x)$ of path P_a, edge xy and a subpath $(y, \ldots, c)$ of path P_c. In particular,*

$$d_H(a, c) \leq \xi.$$

Moreover, $0 \leq d_G(c, y) - d_G(a, x) \leq 1$.

Proof If a is adjacent to c in G, then by construction of H, $d_H(a, c) = 1$, and we are done. So, assume a is not adjacent to c. Then, since a and c are in the same connected component Q_i^l, there must exist a path of G outside the ball $B_{l-1}(u)$ which

connects a and c. Choose an induced subpath P of that path. Consider in G also maximum neighbor paths P_a and P_c, and let x be the vertex of P_a closest to a which has a neighbor in P_c (neighbors are considered in graph G). Denote a neighbor of x in P_c which is closest to c by y. Clearly, path P^* of G formed by (a, x)-subpath of P_a, edge xy and (c, y)-subpath of P_c is induced. Furthermore, only vertices $f(a)$ and $f(c)$ of P^* can have neighbors in P.

Now, if the length of P^* is larger than ξ, then we can find an induced cycle in G of length at least $\xi + 1$ by joining paths P and P^* (even if vertices $f(a)$ and $f(c)$ of P^* have a common neighbor in P we will still get an induced cycle of length at least $\xi + 1$).

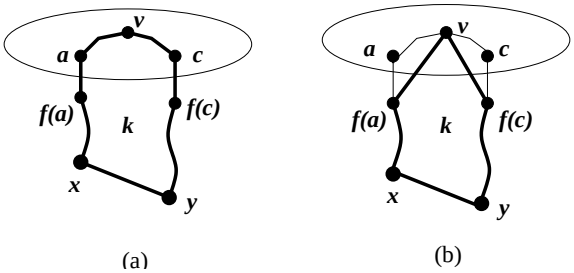


Figure 2.8: The distance between a and c is at most k in H.

So, the length of P^* cannot exceed k (Figure 2.8 illustrates this) and we claim that P^* is a path in H, too. Indeed, both maximum neighbor paths P_a and P_c are in H (since they are from BFS-tree T). If we assume that xy is not in E', then by *while* loop of PROCEDURE 3 and the choice of x and y, we must have $x \in N_j(u)$ and $y \in N_{j+1}(u)$ for some $j < l$. Moreover, $x \neq f(y)$ (otherwise, $xy = f(y)y$ is in E'). Let $P_a = (a = a_1, a_2, \ldots, a_{s-1}, a_s = x, \ldots, u)$ and $P_c = (c = c_1, c_2, \ldots, c_{s-1} = y, c_s = f(y), \ldots, u)$. Since $a > c$ and $a_h \neq c_h$ for any $h \in \{1, \ldots, s\}$, by property (P3), we have $a_h > c_h$ $(h \in \{1, \ldots, s\})$. Now, $x \in N(y)$ and $x > f(y)$ contradict with property (P2). Consequently, edge xy must be in E'. From this we deduce also that if $x \in N_j(u)$ for

some $j < l$, then y is either in $N_j(u)$ or in $N_{j-1}(u)$.

Thus, P^* is a path of H and therefore, $d_H(a, c) \leq \xi$. □

For any n-vertex ξ-chordal graph $G = (V, E)$ the following lemma holds.

Lemma 13 *H is an additive $(\xi + 1)$-spanner of G.*

Proof Consider vertices $x \in Q_i^l$ and $y \in Q_j^m$, and assume that the lowest common ancestor of Q_i^l and Q_j^m in Γ is Q_s^p. Since Γ is a tree, any path connecting x and y passes Q_s^p. So, we have

$$d_G(x, y) \geq l - p + m - p.$$

Since H contains all edges of BFS-tree T, there must exist vertices $a, b \in Q_s^p$ such that $d_H(x, a) = l - p$ and $d_H(y, b) = m - p$. Now, we only need to show that

$$d_H(a, b) \leq \xi + 1.$$

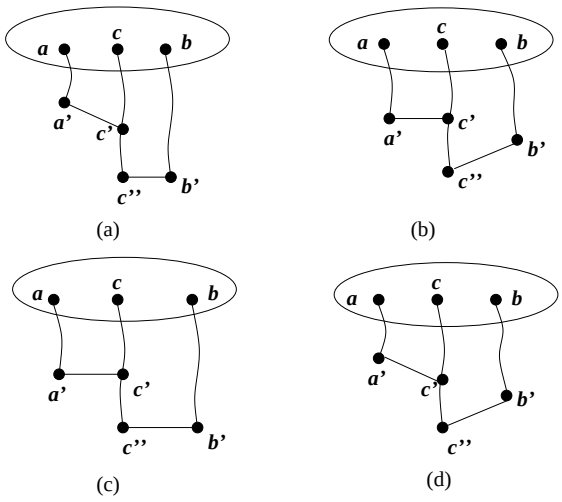


Figure 2.9: Four possibilities (up to symmetry) of connection between a and b: (a) a', c' are in neighbor layers, b', c'' are in the same layer; (b) a', c' are in the same layer, b', c'' are in neighbor layers; (c) a', c' are in the same layer, b', c'' are in the same layer; (d) a', c' are in neighbor layers, b', c'' are in neighbor layers.

Consider the maximum neighbor paths P_a, P_c and P_b in G. According to Lemma 12, there exist (a, c)-path and (b, c)-path in H both of length at most k such that

- (a, c)-path is formed by a subpath $(a, \ldots, a')$ of path P_a, edge $a'c'$ and a subpath $(c', \ldots, c)$ of path P_c, and
- (b, c)-path is formed by a subpath $(b, \ldots, b')$ of path P_b, edge $b'c''$ and a subpath $(c'', \ldots, c)$ of path P_c.

Moreover, $0 \leq d_G(c, c') - d_G(a, a') \leq 1$ and $0 \leq d_G(c, c'') - d_G(b, b') \leq 1$. Hence, up to symmetry, we have only four possible connections in H between vertices a and b that use those (a, c)- and (b, c)-paths (consult with Figure 2.9).

In all cases we have

$$\begin{aligned} d_H(b, a) &\leq d_H(b, b') + 1 + d_H(c'', c') + 1 + d_H(a', a) \\ &\leq d_H(b, b') + 1 + d_H(c'', c') + 1 + d_H(c', c) \\ &= d_H(b, b') + 1 + d_H(c'', c) + 1 \\ &\leq \xi + 1. \end{aligned}$$

Thus, H is an additive $(\xi + 1)$-spanner of G. □

Lemma 14 *If G has n vertices, then H has at most $2n - 2$ edges.*

Proof For each vertex $v \in V$ distinct from u we add to E' one or two edges. Since for u no such edges have been added to E', the total number of edges in H is at most $2n - 2$. □

Lemma 15 *If G is a ξ-chordal graph with n vertices and m edges, then H can be constructed in $O(n \cdot \xi + m)$ time.*

Proof First we show that the connected components of the graphs Q^l $(l = 0, 1, \ldots)$ can be found in total linear time. A BFS-tree of G can be constructed in linear time. Having a BFS-tree, we start from the sphere $N_q(u)$ of largest radius, find its connected components and contract each of them into a vertex. Then find the connected components in the graph induced by $N_{q-1}(u)$ and the set of contracted vertices, contract each of them and descend to the lower level, until we reach the vertex u. So, we can assume that all the connected components Q^l_1, Q^l_2, $\ldots$, $Q^l_{p_l}$ for any l, $0 \leq l \leq q$ are given.

Now, to get overall $O(n \cdot \xi + m)$ time bound, we need only to note that, by Lemma 12, for each vertex v, the *while* loop of PROCEDURE 3 will work at most $\lfloor \xi/2 \rfloor$ time. □

Summarizing, we have the following result for ξ-chordal graphs.

Theorem 4 *Every ξ-chordal graph $G = (V, E)$ with n vertices and m edges admits an additive $(\xi + 1)$-spanner with at most $2n - 2$ edges. Moreover, such a sparse spanner of G can be constructed in $O(n \cdot \xi + m)$ time.*

2.4 Subclasses of 4-chordal graphs

There is an interesting bipartite analog of chordal graphs, so-called *chordal bipartite graphs*. These are bipartite graphs whose largest induced cycles are of length 4. In other words, they are exactly bipartite 4-chordal graphs. By Theorem 4, every such graph with n vertices admits an additive 5-spanner with at most $2n - 2$ edges. It is well-known that this family of graphs has also very close relations to so-called *strongly chordal graphs* (a subclass of chordal graphs). Those relations can be expressed (see [20, 24, 46, 58, 91, 105]) in terms of Γ-free orderings, totally balancedness of the open/closed neighborhood hypergraphs and in many other ways. As it was shown in [19], unlike general chordal graphs, every strongly chordal graph admits an additive tree 3-spanner. Hence, it is very natural to ask whether chordal bipartite graphs admit additive tree

r-spanners for some small integer r. In this section first we note that for every fixed integer l there is a chordal bipartite graph which does not have additive tree $(2l-1)$-spanners and multiplicative tree $2l$-spanners. Then, we show that a slight modification of the PROCEDURE 3 can produce an additive 4-spanner with at most $2n-2$ edges for any chordal bipartite graph. In fact, we will prove a more general result; any 4-chordal graph which does not contain a house as an induced subgraph (see Figure 2.10) admits an additive 4-spanner with at most $2n-2$ edges. 4-Chordal graphs without induced houses are known also as *House-Hole-free graphs* or *HH-free graphs* for short.

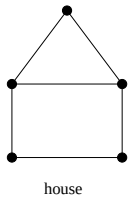


Figure 2.10: Forbidden induced subgraph.

Our construction of bad chordal bipartite graphs is similar to the one presented in [101] for chordal graphs. It was proved in [101] that for every fixed integer l there is a chordal graph which does not have additive tree l-spanners as well as multiplicative tree $(l+1)$-spanners.

Let G_1 be the induced 4-cycle C_4 (the square), and let G_2 be the graph obtained from G_1 by adding for each edge of G_1 a C_4 which shares that edge with G_1. For any integer $l > 2$ the graph G_l is obtained by adding for every edge in $E(G_{l-1}) \setminus E(G_{l-2})$ one new C_4 which shares this edge with G_{l-1} (Figure 2.11 shows graphs G_1 and G_3). These graphs G_l $(l > 0)$ all are planar and chordal bipartite.

Lemma 16 *No additive tree $(2l-1)$-spanner and no multiplicative tree $2l$-spanner is possible for G_l.*

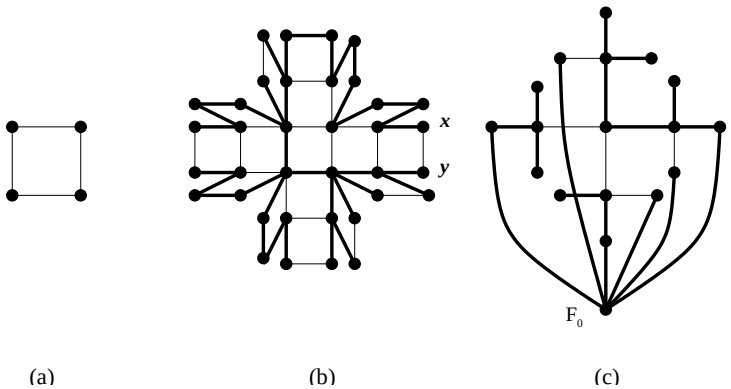


Figure 2.11: (a) Chordal bipartite graph G_1. (b) Chordal bipartite graph G_3 with a spanning tree T. (c) The dual graph of G_3 (for simplicity we did not show all edges incident to F_0) and the dual tree of G_3 with respect to T.

Proof Given a natural plane embedding of the graph G_l, let G_l^* be its geometric dual. Namely, in each face of G_l, including the outer face F_0, we pick a point. These points will form the vertex set of G_l^*. Now, for each edge e of G_l we draw a dual edge in G_l^* which crosses e and connects the vertices of G_l^* that correspond to two faces sharing e in G_l. Note that G_l^* is a multigraph (has parallel edges); its vertex F_0 has three edges to each neighbor.

Let T be a spanning tree of G_l. The dual tree T^* contains all the edges in G_l^* which cross the edges of G_l that do not belong to the spanning tree T. See Figure 2.11 for an illustration.

Let B be the largest connected component of the forest $T^* \setminus \{F_0\}$, and let F_1 be the neighbor of F_0 in B. Clearly, B contains at least $ecc_{T^*}(F_0) \geq ecc_{G_l^*}(F_0) = l$ vertices, where

$$ecc_{T^*}(F_0) = max\{d_{T^*}(F_0, F) : F \in V(T^*)\}$$

and

$$ecc_{G_l^*}(F_0) = max\{d_{G_l^*}(F_0, F) : F \in V(T^*)\}.$$

Note also that in T^*, F_0 must be able to reach the central square.

The edge F_0F_1 crosses an edge xy on the outer cycle in G_l. Since F_0F_1 is an edge in T^*, x and y are not adjacent in T. Moreover, $d_T(x,y)+1$ equals the number of edges in G_l^* that start in B and end outside B. Since all vertices of G_l^* except F_0 have degree 4, this number equals

$$\sum_{F \in V(B)} (4 - deg_B(F)).$$

By the well known sum of degrees formula, and since B is a tree, this equals

$$4|V(B)| - 2|E(B)| = 2|V(B)| + 2 \geq 2l + 2.$$

Therefore, we have $d_T(x,y) \geq 2l+1$, which means T can not be an additive $(2l-1)$-spanner or a multiplicative $2l$-spanner of G_l. □

Let now $G = (V,E)$ be an HH-free graph, σ be a BFS-ordering of G started at a vertex u and T be the BFS-tree of G associated with σ. Assume also that the sets Q_1^l, Q_2^l, ..., $Q_{p_l}^l$ for each l, $l > 0$, (see Section 2.3 for definitions) are already computed. The following algorithm produces an additive 4-spanner for every HH-free graph G.

PROCEDURE 4. Additive 4-spanners for HH-free graphs

Input: An HH-free graph $G = (V,E)$ with a BFS-ordering σ, and connected components Q_1^l, Q_2^l, ..., $Q_{p_l}^l$ for any l, $0 \leq l \leq q$, where $q = max\{d_G(u,v) : v \in V\}$.

Output: A spanner $H = (V,E')$ of G.

Method:

$E' = \emptyset$;

for $l = q$ **downto** 1 **do**

  for $j = 1$ **to** p_l **do**

    for each vertex $v \in Q_j^l$ add $vf(v)$ to E';

pick vertex c in Q_j^l with the minimum number in σ;

for each $v \in Q_j^l \setminus \{c\}$ **do**

connected = FALSE;

while connected = FALSE **do**

/* this while loop works at most twice for each v */

if $vf(c) \in E(G)$ **then** /* here this method differs from the */

add $vf(c)$ to E'; /* one of Section 2.3; we first check */

connected = TRUE; /* adjacency between v and $f(c)$ */

else if $vc \in E(G)$ **then** /* and then between v and c. */

add vc to E';

connected = TRUE;

else $v = f(v)$, $c = f(c)$

return $H = (V, E')$.

Since, clearly, H has at most $2n - 2$ edges and can be constructed in linear time, we need to prove only that H is an additive 4-spanner of G. The following auxiliary lemma for HH-free graph G will be needed in that proof.

Lemma 17 *For any $l > 0$ and any two adjacent vertices a, c from $N_l(u)$ such that $a > c$ in σ, a must be adjacent to $f(c)$.*

Proof Let l be the smallest integer such that $af(c) \notin E(G)$ for $a, c \in N_l(u)$, $ac \in E(G)$, $a > c$. Since $a > c$ and $af(c) \notin E(G)$, by properties of BFS-orderings, we have $f(a) > f(c)$ and $f(a)c \notin E(G)$. To avoid a forbidden induced cycle of length at least 5 in G formed by $a, c, f(a), f(c)$ and some vertices from $B_{l-2}(u)$, vertices $f(a)$ and $f(c)$ must be adjacent. By minimality of l, the farther of $f(c)$ has to be adjacent to $f(a)$. But then, vertices $a, c, f(a), f(c), f(f(c))$ induce a house, which is impossible. □

Lemma 18 *H is an additive 4-spanner for G.*

Proof Similar to the proof of Lemma 13, we only need to show that $d_H(a,b) \leq 4$ holds for any two vertices a, b from Q_s^p. Let c be the vertex of Q_s^p with the minimum number in σ. By the proof of Lemma 12, at least one of the following edges $ac, af(c), f(a)f(c), f(a)f(f(c))$ must exist in G. But by Lemma 17, if $ac \in E(G)$ then $af(c) \in E(G)$ and if $f(a)f(c) \in E(G)$ then $f(a)f(f(c)) \in E(G)$. Therefore, by PROCEDURE 4, either edge $af(c)$ or edge $f(a)f(f(c))$ will go to $E(H)$. Symmetrically, either edge $bf(c)$ or edge $f(b)f(f(c))$ will go to $E(H)$. In any of four possible cases we have $d_H(a,b) \leq 4$. Recall that all edges of BFS-tree T are in H. □

Our final result is the following.

Theorem 5 *Every HH-free graph $G = (V, E)$ with n vertices and m edges admits an additive 4-spanner with at most $2n - 2$ edges. Moreover, such a sparse spanner of G can be constructed in $O(n + m)$ time.*

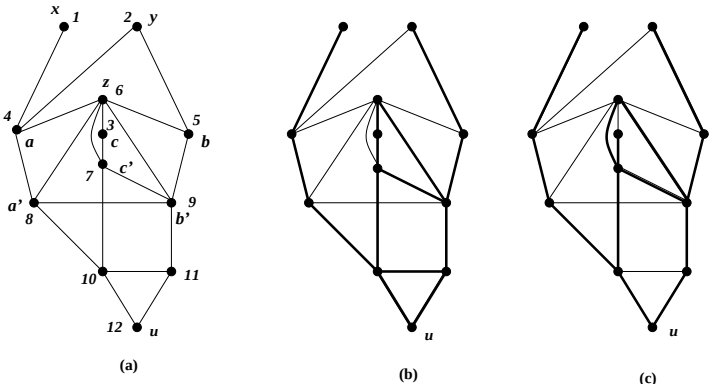


Figure 2.12: (a) A weakly-chordal graph with a LexBFS-ordering (b) An additive 5-spanner generated by PROCEDURE 3. (d) An additive 5-spanner generated by PROCEDURE 4.

It is interesting to note that both procedures (3 and 4) may produce additive 5-spanners for the well known class of weakly chordal graphs (see Figure 2.12). Recall

that G is a *weakly chordal graph* if both G and its complement $\overline{G}$ are 4-chordal. This class is a superclass of HH-free graphs and a subclass of 4-chordal graphs. A question whether a weakly chordal graph admits an additive 4-spanner with $O(n)$ edges remains open.

CHAPTER 3

COLLECTIVE TREE SPANNERS

3.1 Introduction

In Chapter 2, we discussed graph spanners. The sparsest spanners are tree spanners. Tree spanners occur in [9], and as it was shown in [96], they can be used as models for broadcast operations in communication networks. Tree spanners are favored also from the algorithmic point of view - many algorithmic problems are easily solvable on trees. Multiplicative tree t-spanners were studied in [27]. It was shown that, for a given graph G, the problem to decide whether G has a multiplicative tree t-spanner (the multiplicative tree t-spanner problem) is NP-complete for any fixed $t \geq 4$ and is linearly solvable for t= 1, 2. Recently, this NP-completeness result was improved-the multiplicative tree t-spanner problem is NP-complete for any fixed $t \geq 4$ even on some rather restricted graph classes: planar graphs [18], chordal graphs [21] and chordal bipartite graphs [22].

These results tell us that to use one spanning tree to approximate a graph in distance is difficult. Then a natural question is: Can we relax the requirement that we can only use one spanning tree? In this chapter we introduce a new notion - *collective tree spanners*, a notion slightly *weaker* than the one of a tree spanner and slightly *stronger* than the notion of a sparse spanner. We say that a graph $G = (V, E)$ admits a system of μ collective additive tree r-spanners if there is a system $T(G)$ of at most μ spanning trees of G such that for any two vertices x, y of G a spanning tree exists such that $d_T(x, y) \leq d_G(x, y) + r$ (a multiplicative variant of this notion can be defined analogously). Clearly, if G admits a system of μ collective additive tree r-spanners, then

G admits an additive r-spanner with at most $\mu \cdot (n-1)$ edges (take the union of all those trees), and if $\mu = 1$ then G admits an additive tree r-spanner. Furthermore, any result on collective additive tree spanners can be translated into a result on collective multiplicative tree spanners since any graph, admitting a system of μ collective additive tree r-spanners, admits a system of μ collective multiplicative tree $(r+1)$-spanners ($d_T(x,y) \leq d_G(x,y) + r$ implies $d_T(x,y)/d_G(x,y) \leq 1 + r/d_G(x,y) \leq r+1$ for an unweighted graph G). Note also that any graph on n vertices admits a system of at most $n-1$ collective additive tree 0-spanners (take $n-1$ breadth-first-search-trees rooted at different vertices of G).

The introduction of this new notion was inspired by the works [10, 11] of Bartal and subsequent works [29, 60]. For example, motivated by Bartal's work on probabilistic approximation of general metrics with tree metrics, [29] gives a polynomial time algorithm that given a finite n point metric G, constructs $O(n \log n)$ trees and a probability distribution ψ on them such that the expected multiplicative stretch of any edge of G in a tree chosen according to ψ is at most $O(\log n \log \log n)$. These results led to approximation algorithms for a number of optimization problems including the group Steiner tree problem, the metric labeling problem, the buy-at-bulk network design problem and many others (see [10, 11, 29, 60] for more details).

In the following, we define a large class of graphs, called (α, r)-decomposable, and show that any (α, r)-decomposable graph G with n vertices admits a system of at most $\log_{1/\alpha} n$ collective additive tree $2r$-spanners. Then, in sections 3.2.4 and 3.2.5, we show that chordal graphs and chordal bipartite graphs are all $(1/2, 1)$-decomposable graphs, implying that each graph from those families admits a system of at most $\log_2 n$ collective additive tree 2-spanners. These results are complemented by lower bounds, which say that any system of collective additive tree 1-spanners must have $\Omega(\sqrt{n})$ spanning trees for some chordal graphs and $\Omega(n)$ spanning trees for some chordal bipartite graphs.

Furthermore, we show that any ξ-chordal graph is $(1/2,\ \lfloor \xi/2 \rfloor)$-decomposable, implying that each ξ-chordal graph admits a system of at most $\log_2 n$ collective additive tree $(2\lfloor \xi/2 \rfloor)$-spanners.

Thus, as a byproduct, we get that chordal graphs and chordal bipartite graphs admit additive 2-spanners with at most $(n-1)\log_2 n$ edges and ξ-chordal graphs admit additive 2-spanners with at most $(n-1)\log_2 n$ edges. Our result for chordal graphs improves the known results from [97] and [32] on 3-spanners and answers the question posed in [32] whether chordal graphs admit additive 2-spanners with $O(n \log_2 n)$ edges.

In section 3.4, we show that any unit disk graph (UDG) $G = (V, E)$ admits a system of $O(\log n)$ collective tree (3, 4)-spanners, i.e. for any two vertices $u, v \in V$, there is a spanning tree T_i in the system such that $d_{T_i}(u, v) \leq 3d_G(u, v) + 4$ holds. These tree spanners can be constructed in polynomial time. In section 3.5, we present our results on collective tree spanners on AT-free related graphs. In section 3.2.6, we discuss an application of the collective tree spanners to the problem of designing compact and efficient routing schemes in graphs. For any graph on n vertices admitting a system of at most μ collective additive tree r-spanners, there is a routing scheme of deviation r with addresses and routing tables of size $O(\mu \log n / \log \log n)$ bits per vertex (for details see section 3.2.6). This leads, for example, to a routing scheme of deviation $(2\lfloor \xi/2 \rfloor)$ with addresses and routing tables of size $O(\log^3 n / \log \log n)$ bits per vertex on the class of ξ-chordal graphs.

Results of this chapter were partially published in [51, 52, 48, 49, 50, 53].

3.1.1 Basic notions and notations

All graphs occurring in this chapter are connected, finite, undirected, loopless and without multiple edges. The *distance* $d_G(u, v)$ between the vertices u and v is the length of a shortest path connecting u and v. Our graphs can also have (non-negative) weights

on edges, $w(e)$, $e \in E$, unless otherwise is specified. In a weighted graph $G = (V, E)$ the *distance* $d_G(u, v)$ between the vertices u and v is the length of a shortest path connecting u and v. If the graph is unweighted then, for convenience, each edge has weight 1. In an unweighted graph $G = (V, E)$ the *length* of a path from a vertex v to a vertex u is the number of edges in the path.

For a subset $S \subseteq V$, let $rad_G(S)$ and $diam_G(S)$ be the radius and the diameter, respectively, of S in G, i.e., $rad_G(S) = \min_{v \in V}\{\max_{u \in S}\{d_G(u, v)\}\}$, $diam_G(S) = \max_{u,v \in S}\{d_G(u,\ v)\}$. A vertex $v \in V$ such that $d_G(u, v) \leq rad_G(S)$ for any $u \in S$, is called a central vertex for S. The value $rad_G(V)$ is called *the radius* of G. Define the layers of G with respect to a vertex u as follows: $L_i(u) = \{x \in V : x$ can be connected to u by a path with i edges but not by a path with $i-1$ edges$\}$, $i = 0, 1, 2, \ldots$. In a path $P = (v_0, v_1, \ldots, v_l)$ between vertices v_0 and v_l of G, vertices $v_1, \ldots, v_{l-1}$ are called *inner vertices*. Let r be a non-negative real number. A set $D \subseteq V$ is called an *r-dominating set* for a set $S \subseteq V$ of a graph G if $d_G(v, D) \leq r$ holds for any $v \in S$.

A tree-decomposition [102] of a graph G is a tree T whose nodes, called *bags*, are subsets of $V(G)$ such that: 1) $\bigcup_{X \in V(T)} X = V(G)$; 2) for all $\{u, v\} \in E(G)$, there exists $X \in V(T)$ such that $u, v \in X$; and 3) for all $X, Y, Z \in V(T)$, if Y is on the path from X to Z in T then $X \cap Z \subseteq Y$. The *width* of a tree-decomposition is one less than the maximum cardinality of a bag. Among all the tree-decompositions of G, let T be the one with minimum *width*. The width of T is called the *tree-width* of the graph G and is denoted by $tw(G)$. We say that G *has bounded tree-width* if $tw(G)$ is bounded by a constant. It is known that the tree-width of an outerplanar graph and of a series-parallel graph is at most 2 (see, e.g., [16, 78]).

A related notion to tree-width is *clique-width*. Based on the following operations on vertex-labeled graphs, namely (i) creation of a vertex labeled by integer l, (ii) disjoint union (i.e., co-join), (iii) join between all vertices with label i and all vertices with label

j for $i \neq j$, and (iv) relabeling all vertices of label i by label j, the notion of *clique-width* $cw(G)$ of a graph G is defined in [56] as the minimum number of labels which are necessary to generate G by using these operations. Clique-width is a complexity measure on graphs somewhat similar to tree-width, but more powerful since every set of graphs with bounded tree-width has bounded clique-width [37] but not conversely (cliques have clique-width 2 but unbounded tree-width). It is well-known that the clique-width of a cograph is at most 2 and the clique-width of a distance-hereditary graph is at most 3 (see [73]).

As discussed in the previous chapter, the well-known chordal graphs are exactly the 3-chordal graphs. An induced cycle of G of size at least 5 is called a *hole*. The complement of a hole is called an *anti-hole*. A graph G is *weakly chordal* if it has neither holes nor anti-holes as induced subgraphs. Clearly, weakly chordal graphs and their complements are 4-chordal. A *cograph* is a graph having no induced paths on 4 vertices (P_4s).

The *genus* of a graph G is the smallest integer g such that G embeds in a surface of genus g without edge crossings. Planar graphs can be embedded on a sphere, hence $g = 0$ for them. A planar graph is *outerplanar* if all its vertices belong to its outerface.

3.2 (α, r)–Decomposable graphs and their collective tree spanners

Different balanced separators in graphs were used by many authors in designing efficient graph algorithms (see [40, 41, 66, 67, 70, 77, 88, 89]). For example, bounded size balanced separators and bounded diameter balanced separators were recently employed in [66, 67, 77] for designing compact distance labeling schemes for different so-called well-separated families of graphs. We extend those ideas and apply them to our problem.

Let α be a positive real number smaller than 1 and r be a non-negative integer. We say that an n-vertex graph $G = (V, E)$ is (α, r)–*decomposable* if there is a separator

$S \subseteq V$, such that the following three conditions hold:

Balanced Separator condition - the removal of S leaves no connected component with more than αn vertices;

Bounded Separator-Radius condition - $rad_G(S) \leq r$, i.e., there exists a vertex c in G (called *a central vertex for* S) such that $d_G(v, c) \leq r$ for any $v \in S$;

Hereditary Family condition - each connected component of the graph, obtained from G by removing vertices of S, is also an (α, r)–decomposable graph.

Note that, by definition, any graph of radius at most r is (α, r)–decomposable and that the size of S does not matter.

3.2.1 Collective tree spanners of (α, r)–decomposable graphs

Using the first and third conditions of the definition, one can construct for any (α, r)–decomposable graph G a *(rooted) balanced decomposition tree* $\mathcal{BT}(G)$ as follows. If G is of radius at most r, then $\mathcal{BT}(G)$ is a one node tree. Otherwise, find a balanced separator S in G, which exists according to the Balanced Separator condition. Let $G_1, G_2, \ldots, G_p$ be the connected components of the graph $G - S$ obtained from G by removing vertices of S. For each graph G_i $(i = 1, \ldots, p)$, which is (α, r)–decomposable by the Hereditary Family condition, construct a balanced decomposition tree $\mathcal{BT}(G_i)$ recursively, and build $\mathcal{BT}(G)$ by taking S to be the root and connecting the root of each tree $\mathcal{BT}(G_i)$ as a child of S. See Figure 3.3 for an illustration. Clearly, the nodes of $\mathcal{BT}(G)$ represent a partition of the vertex set V of G into *clusters* $S_1, S_2, \ldots, S_q$ of radius at most r each. For a node X of $\mathcal{BT}(G)$, denote by $G(\downarrow X)$ the (connected) subgraph of G induced by vertices $\bigcup\{Y : Y$ is a descendent of X in $\mathcal{BT}(G)\}$ (here we assume that X is a descendent of itself).

It is easy to see that a balanced decomposition tree $\mathcal{BT}(G)$ of a graph G with n

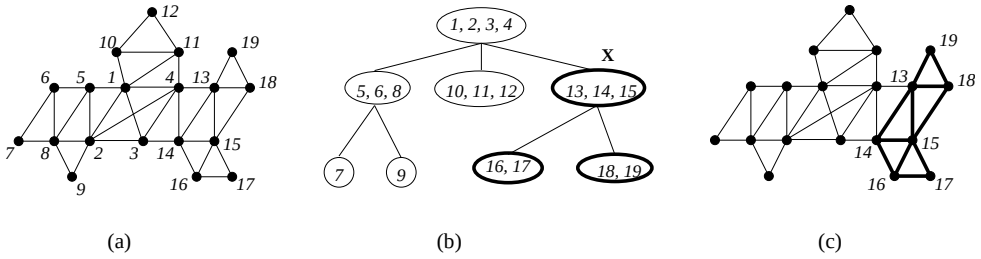


Figure 3.1: (a) A graph G, (b) its balanced decomposition tree $\mathcal{BT}(G)$ and (c) an induced subgraph $G(\downarrow X)$ of G.

vertices and m edges has depth at most $\log_{1/\alpha} n$, which is $O(\log_2 n)$ if α is a constant. Moreover, assuming that a balanced and bounded radius separator can be found in polynomial, say $p(n)$, time (for the special graph classes we consider later, $p(n)$ will be at most $O(n^3)$), the tree $\mathcal{BT}(G)$ can be constructed in $O((p(n)+m)\log_{1/\alpha} n)$ total time. Indeed, in each level of recursion we need to find balanced and bounded radius separators in current disjoint subgraphs and to construct the corresponding subgraphs of the next level. Also, since the graph sizes are reduced by a factor α, the recursion depth is at most $\log_{1/\alpha} n$.

Consider now two arbitrary vertices x and y of an (α, r)–decomposable graph G and let $S(x)$ and $S(y)$ be the nodes of $\mathcal{BT}(G)$ containing x and y, respectively. Let also $NCA_{\mathcal{BT}(G)}(S(x), S(y))$ be the nearest common ancestor of nodes $S(x)$ and $S(y)$ in $\mathcal{BT}(G)$ and $(X_0, X_1, \ldots, X_t)$ be the path of $\mathcal{BT}(G)$ connecting the root X_0 of $\mathcal{BT}(G)$ with $NCA_{\mathcal{BT}(G)}\ (S(x), S(y)) = X_t$ (in other words, $X_0, X_1, \ldots, X_t$ are the common ancestors of $S(x)$ and $S(y)$). The following lemmata are crucial to all our subsequent results.

Lemma 19 *Any path $P^G_{x,y}$, connecting vertices x and y in G, contains a vertex from $X_0 \cup X_1 \cup \ldots \cup X_t$.*

Let $SP^G_{x,y}$ be a shortest path of G connecting vertices x and y, and let X_i be the

node of the path $(X_0, X_1, \ldots, X_t)$ with the smallest index such that $SP^G_{x,y} \bigcap X_i \neq \emptyset$ in G. Then, the following lemma holds.

Lemma 20 *We have $d_G(x,y) = d_{G'}(x,y)$, where $G' := G(\downarrow X_i)$.*

Proof It is enough to show that the path $SP^G_{x,y}$ consists of only vertices of G'. Let assume, by way of contradiction, that there is a vertex z of $SP^G_{x,y}$ that does not belong to G'. Let $SP^G_{x,z}$ be a subpath of $SP^G_{x,y}$ between x and z. Clearly, the node $S(z)$ of $\mathcal{BT}(G)$, containing vertex z, is not a descendent of X_i. Therefore, the nearest common ancestor of $S(x)$ and $S(z)$ in $\mathcal{BT}(G)$ is a node X_j from $\{X_0, X_1, \ldots, X_t\}$ with $j < i$. But then, by Lemma 19, the path $SP^G_{x,z}$ (and hence the path $SP^G_{x,y}$) must have a vertex in $X_0 \cup X_1 \cup \ldots \cup X_j$, contradicting with the choice of X_i, $i > j$.

For the graph $G' = G(\downarrow X_i)$, consider its arbitrary *Breadth-First-Search–tree (BFS–tree)* T' rooted at a central vertex c for X_i, i.e., a vertex c such that $d_{G'}(v,c) \leq r$ for any $v \in X_i$. Such a vertex exists in G' since G' is an (α, r)–decomposable graph and X_i is its balanced and bounded radius separator. The tree T' has the following distance property with respect to those vertices x and y.

Lemma 21 *We have $d_{T'}(x,y) \leq d_G(x,y) + 2r$.*

Proof We know, by Lemma 20, that a shortest path $SP^G_{x,y}$, intersecting X_i and not intersecting any X_l $(l < i)$, lies entirely in G'. Let x' be the vertex of $SP^G_{x,y} \cap X_i$ closest to x and y' be the vertex of $SP^G_{x,y} \cap X_i$ closest to y. Since T' is a BFS–tree of G' rooted at vertex c, we have

$$d_{T'}(x,c) = d_{G'}(x,c) \leq d_{G'}(x,x') + d_{G'}(x',c) \leq d_{G'}(x,x') + r = d_G(x,x') + r,$$

$$d_{T'}(y,c) = d_{G'}(y,c) \leq d_{G'}(y,y') + d_{G'}(y',c) \leq d_{G'}(y,y') + r = d_G(y,y') + r.$$

That is, $d_{T'}(x,y) \leq d_{T'}(x,c) + d_{T'}(y,c) \leq d_G(x,x') + d_G(y,y') + 2r$. Combining this with the fact that $d_G(x,y) \geq d_G(x,x') + d_G(y,y')$, we obtain $d_{T'}(x,y) \leq d_G(x,y) + 2r$.

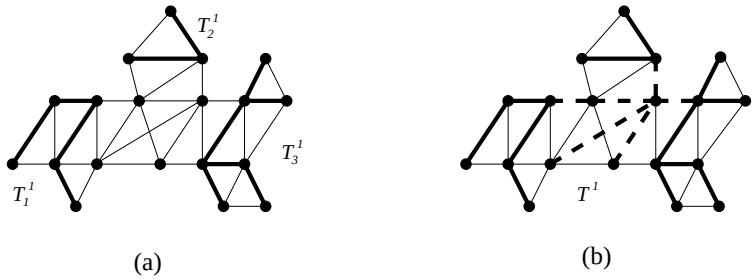


Figure 3.2: (a) Local subtrees T_1^1, T_2^1, T_3^1 of graph G from Figure 3.3 and (b) a corresponding spanning tree T^1 of G (dark solid edges are edges of local subtrees T_1^1, T_2^1, T_3^1, dashed edges are added to create one spanning tree T^1 on top of T_1^1, T_2^1, T_3^1).

Let now $B_1^i, \ldots, B_{p_i}^i$ be the nodes on depth i of the tree $\mathcal{BT}(G)$. For each subgraph $G_j^i := G(\downarrow B_j^i)$ of G ($i = 0, 1, \ldots, depth(\mathcal{BT}(G))$, $j = 1, 2, \ldots, p_i$), denote by T_j^i a BFS–tree of graph G_j^i rooted at a central vertex c_j^i for B_j^i (see Figure 3.2 for an illustration). The trees T_j^i ($i = 0, 1, \ldots, depth(\mathcal{BT}(G))$, $j = 1, 2, \ldots, p_i$) are called *local subtrees* of G, and, given the balanced decomposition tree $\mathcal{BT}(G)$, they can be constructed in $O((t(n)+m)\log_{1/\alpha} n)$ total time, where $t(n)$ is the time needed to find a central vertex c_j^i for B_j^i (a trivial upper bound for $t(n)$ is $O(n^3)$). From Lemma 21 the following general result can be deduced.

Theorem 6 *Let G be an (α, r)–decomposable graph, $\mathcal{BT}(G)$ be its balanced decomposition tree and $\mathcal{LT}(G) = \{T_j^i : i = 0, 1, \ldots, depth(\mathcal{BT}(G)),\ j = 1, 2, \ldots, p_i\}$ be its local subtrees. Then, for any two vertices x and y of G, there exists a local subtree $T_{j'}^{i'}$ in $\mathcal{LT}(G)$ such that*

$$d_{T_{j'}^{i'}}(x, y) \leq d_G(x, y) + 2r.$$

This theorem implies two important results for the class of (α, r)–decomposable graphs. Let G be an (α, r)–decomposable graph with n vertices and m edges, $\mathcal{BT}(G)$ be its balanced decomposition tree and $\mathcal{LT}(G)$ be the family of its local subtrees (defined above). Consider a graph H obtained by taking the union of all local subtrees of G (by

putting all of them together), i.e.,

$$H := \bigcup\{T_j^i : T_j^i \in \mathcal{LT}(G)\} = (V, \cup\{E(T_j^i) : T_j^i \in \mathcal{LT}(G)\}).$$

Clearly, H is a spanning subgraph of G, constructable in $O((p(n)+t(n)+m)\log_{1/\alpha} n)$ total time, and, for any two vertices x and y of G, $d_H(x,y) \leq d_G(x,y) + 2r$ holds. Also, since for every level i ($i = 0, 1, \ldots, depth(\mathcal{BT}(G))$) of balanced decomposition tree $\mathcal{BT}(G)$, the corresponding local subtrees $T_1^i, \ldots, T_{p_i}^i$ are pairwise vertex-disjoint, their union has at most $n-1$ edges. Therefore, H cannot have more than $(n-1)\log_{1/\alpha} n$ edges in total. Thus, we have proven the following result.

Theorem 7 *Any (α, r)–decomposable graph G with n vertices admits an additive $2r$–spanner with at most $(n-1)\log_{1/\alpha} n$ edges.*

Instead of taking the union of all local subtrees of G, one can fix i ($i \in \{0, 1, \ldots, depth(\mathcal{BT}(G))\}$) and consider separately the union of only local subtrees $T_1^i, \ldots, T_{p_i}^i$, corresponding to the level i of the decomposition tree $\mathcal{BT}(G)$, and then extend in linear $O(m)$ time that forest to a spanning tree T^i of G (using, for example, a variant of the Kruskal's Spanning Tree algorithm for the unweighted graphs). We call this tree T^i the *spanning tree of G corresponding to the level i of the balanced decomposition $\mathcal{BT}(G)$*. In this way we can obtain at most $\log_{1/\alpha} n$ spanning trees for G, one for each level i of $\mathcal{BT}(G)$. Denote the collection of those spanning trees by $\mathcal{T}(G)$. By Theorem 6, it is rather straightforward to show that for any two vertices x and y of G, there exists a spanning tree $T^{i'}$ in $\mathcal{T}(G)$ such that $d_{T^{i'}}(x,y) \leq d_G(x,y) + 2r$. Thus, we have

Theorem 8 *Any (α, r)–decomposable graph G with n vertices admits a system $\mathcal{T}(G)$ of at most $\log_{1/\alpha} n$ collective additive tree $2r$–spanners.*

Note that such a system $\mathcal{T}(G)$ for an (α, r)–decomposable graph G with n vertices and m edges can be constructed in $O((p(n)+t(n)+m)\log_{1/\alpha} n)$ time, where $p(n)$ is the

time needed to find a balanced and bounded radius separator S and $t(n)$ is the time needed to find a central vertex for S.

3.2.2 Extracting an appropriate tree from $\mathcal{T}(G)$

Now we will show that, one can assign $O(\log_{1/\alpha} n \times \log n)$ bit labels to vertices of G such that, for any pair of vertices x and y, a tree $T^{i'}$ in $\mathcal{T}(G)$ with $d_{T^{i'}}(x,y) \leq d_G(x,y) + 2r$ can be identified in only $O(\log_{1/\alpha} n)$ time by merely inspecting the labels of x and y, without using any other information about the graph. This will be useful in an application of collective tree spanners, discussed in Section 3.2.6.

Associate with each vertex x of G a $2 \times (depth(\mathcal{BT}(G)) + 1)$ array A_x such that, for each level i of $\mathcal{BT}(G)$, $A_x[1,i] = j$ and $A_x[2,i] = d_{T^i_j}(x, c^i_j)$ if there exists a local subtree T^i_j in $\mathcal{LT}(G)$ containing vertex x, and $A_x[1,i] = nil$ and $A_x[2,i] = \infty$, otherwise (i.e., the depth in $\mathcal{BT}(G)$ of node $S(x)$ containing x is smaller than i). Evidently, each label A_x ($x \in V$) can be encoded using $O(\log_{1/\alpha} n \times \log n)$ bits and a computation of all labels A_x, $x \in V$, can be performed together with the construction of system $\mathcal{T}(G)$.

Given labels A_x, A_y of vertices x and y, the following procedure will return in $O(\log_{1/\alpha} n)$ time an index $i' \in \{0, 1, \ldots, depth(\mathcal{BT}(G))\}$ such that for tree $T^{i'} \in \mathcal{T}(G)$, $d_{T^{i'}}(x,y) \leq d_G(x,y) + 2r$ holds.

set $i' := 0$;
set $minsum := A_x[2,0] + A_y[2,0]$;
set $i := 1$;
while $(A_x[1,i] = A_y[1,i] \neq nil)$ and $(i \leq \log_{1/\alpha} n)$ do
  if $A_x[2,i] + A_y[2,i] < minsum$
    then set $i' := i$ and $minsum := A_x[2,i] + A_y[2,i]$;
  $i := i + 1$;
enddo

return i'.

This procedure simply finds, among all local subtrees containing both x and y, a subtree $T^{i'}_{j'}$ for which the sum $d_{T^{i'}_{j'}}(x, c^{i'}_{j'}) + d_{T^{i'}_{j'}}(y, c^{i'}_{j'})$ is minimum, and then returns its upper index i'.

To show that indeed $d_{T^{i'}}(x, y) \leq d_G(x, y) + 2r$, we will need to recall the proof of Lemma 21 (note that $d_{T^{i'}}(x, y) = d_{T^{i'}_{j'}}(x, y)$, by construction of $T^{i'}$). Let again $S(x)$ and $S(y)$ be the nodes of $\mathcal{BT}(G)$ containing vertices x and y, respectively, and let $(B^0, B^1_{j_1}, \ldots, B^t_{j_t})$ be the path of $\mathcal{BT}(G)$ connecting the root B^0 of $\mathcal{BT}(G)$ with $NCA_{\mathcal{BT}(G)}(S(x), S(y)) = B^t_{j_t}$. In Lemma 21 we proved that there exists an index $i \in \{0, 1, \ldots, t\}$ such that any BFS-tree T' of the graph $G(\downarrow B^i_{j_i})$ rooted at a center c for $B^i_{j_i}$ (including local subtree $T^i_{j_i}$ rooted at $c^i_{j_i}$) satisfies $d_{T'}(x, y) \leq d_{T'}(x, c) + d_{T'}(y, c) \leq d_G(x, y) + 2r$ (see inequalities (1) and (2) in that proof). Since, among local subtrees $T^0, T^1_{j_1}, \ldots, T^t_{j_t}$, the subtree $T^{i'}_{j'}$ has minimum sum $d_{T^{i'}_{j'}}(x, c^{i'}_{j'}) + d_{T^{i'}_{j'}}(y, c^{i'}_{j'})$, we conclude

$$d_{T^{i'}}(x, y) = d_{T^{i'}_{j'}}(x, y) \leq d_{T^{i'}_{j'}}(x, c^{i'}_{j'}) + d_{T^{i'}_{j'}}(y, c^{i'}_{j'}) \leq$$

$$d_{T^i_{j_i}}(x, c^i_{j_i}) + d_{T^i_{j_i}}(y, c^i_{j_i}) \leq d_G(x, y) + 2r.$$

3.2.3 Acyclic hypergraphs, chordal graphs and (α, r)-decomposition

Let $H = (V, \mathcal{E})$ be a *hypergraph* with the vertex set V and the *hyperedge* set $\mathcal{E}$, i.e., $\mathcal{E}$ is a set of non-empty subsets of V. 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 (a set of pairwise adjacent vertices) 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. For these and other hypergraph notions see [14].

The following theorem represents two well-known characterizations of acyclic hypergraphs. Let $\mathcal{C}(G)$ be the set of all maximal (by inclusion) cliques of a graph $G = (V, E)$. The hypergraph $(V, \mathcal{C}(G))$ is called the clique-hypergraph of G. Recall that a graph G is *chordal* if it does not contain any induced cycles of length greater than 3.

Theorem 9 *[6, 13, 14, 25, 59, 107] Let $H = (V, \mathcal{E})$ be a hypergraph. Then the following conditions are equivalent:*

(i) H is an acyclic hypergraph;

(ii) H is conformal and $2SEC(H)$ of H is a chordal graph;

(iii) H is the clique hypergraph $(V, \mathcal{C}(G))$ of some chordal graph $G = (V, E)$.

Later we will need also the following known result. A vertex v of a graph G is called *simplicial* if its neighborhood $N(v)$ form a clique in G.

Theorem 10 *[25, 39] Let $G = (V, E)$ be a graph. Then the following conditions are equivalent:*

(i) G is a chordal graph;

(ii) the clique hypergraph $(V, \mathcal{C}(G))$ of G is acyclic (in other words, G is the intersection graph of a family of subtrees of a tree);

(iii) G has a perfect elimination ordering. i.e., an ordering $v_1, v_2, \ldots, v_n$ of vertices of G such that for any i, $i \in \{1, 2, \ldots, n\}$, vertex v_i is simplicial in graph $G(v_i, \ldots, v_n)$, the subgraph of G induced by vertices $v_i, \ldots, v_n$.

Let now $G = (V, E)$ be an arbitrary graph and r be a positive integer. We say that G *admits a radius r acyclic covering* if there is a family $\mathcal{S}(G) = \{S_1, \ldots, S_k\}$ of subsets of V such that

(1) $\bigcup_{i=1}^{k} S_i = V$;

(2) for any edge xy of G there is a subset S_i ($i \in \{1, \ldots, k\}$) with $x, y \in S_i$;

(3) $H = (V, \mathcal{S}(G))$ is an acyclic hypergraph;

(4) $rad_G(S_i) \leq r$ for each $i = 1, \ldots, k$.

A class of graphs $\mathcal{F}$ is called *hereditary* if every induced subgraph of a graph G belongs to $\mathcal{F}$ whenever G is in $\mathcal{F}$. A class of graphs $\mathcal{F}$ is called (α, r)*–decomposable* if every graph G from $\mathcal{F}$ is (α, r)–decomposable.

Theorem 11 *Let $\mathcal{F}$ be a hereditary class of graphs such that any $G \in \mathcal{F}$ admits a radius r acyclic covering. Then $\mathcal{F}$ is a $(1/2, r)$–decomposable class of graphs.*

Proof Consider a graph $G \in \mathcal{F}$ and let $\mathcal{S}(G) = \{S_1, \ldots, S_k\}$ be its radius r acyclic covering. Since $H = (V, \mathcal{S}(G))$ is an acyclic hypergraph, $2SEC(H)$ is chordal and H is conformal. It is well known [71], that every n-vertex chordal graph Γ contains a maximal clique C such that if the vertices in C are deleted from Γ, every connected component in the graph induced by any remaining vertices is of size at most $n/2$. Moreover, according to [71], for any chordal graph on n vertices and m edges, such a separating clique C can be found in $O(n + m)$ time. Applying this result to an n-vertex chordal graph $2SEC(H)$, we will get in at most $O(n^2)$ time a maximal clique S of $2SEC(H)$ such that any connected component of the graph $2SEC(H) - S$ (obtained from $2SEC(H)$ by deleting vertices of S) has at most $n/2$ vertices. Since $2SEC(H)$ is obtained from G by adding some new edges, removing vertices of S from the original graph G will leave

no connected component (in $G - S$) with more than $n/2$ vertices. Furthermore, since $\mathcal{F}$ is a hereditary class of graphs, all connected components of $G - S$ induce graphs from $\mathcal{F}$ (and they can be assumed by induction to be $(1/2, r)$–decomposable graphs). It remains to note that, from conformality of H, there must exist a set S_i in $\mathcal{S}(G)$ which contains S, i.e., $rad_G(S) \leq rad_G(S_i) \leq r$ must hold.

Since for a chordal graph $G = (V, E)$ the clique hypergraph $(V, \mathcal{C}(G))$ is acyclic and chordal graphs form a hereditary class of graphs, from Theorem 11 and Theorems 7 and 8, we immediately conclude

Corollary 1 *Any chordal graph G with n vertices and m edges admits an additive 2–spanner with at most $(n-1)\log_2 n$ edges, and such a sparse spanner can be constructed in $O(m \log_2 n)$ time.*

Corollary 2 *Any chordal graph G with n vertices and m edges admits a system $\mathcal{T}(G)$ of at most $\log_2 n$ collective additive tree 2–spanners, and such a system of spanning trees can be constructed in $O(m \log_2 n)$ time.*

Note that, since any additive r-spanner is a multiplicative $(r+1)$-spanner, Corollary 1 improves a known result of Peleg and Schäffer on sparse spanners of chordal graphs. In [97], they proved that any chordal graph with n vertices admits a multiplicative 3–spanner with at most $O(n \log_2 n)$ edges and a multiplicative 5-spanner with at most $2n - 2$ edges. Both spanners can be constructed in polynomial time. Note also that their result on multiplicative 5-spanners was earlier improved in [32], where the authors showed that any chordal graph with n vertices admits an additive 4-spanner with at most $2n - 2$ edges, constructable in linear time. Motivated by this and Corollary 2, it is natural to ask whether a system of constant number of collective additive tree 4–spanners exists for a chordal graph (or, generally, for which r, a system of constant number of collective additive tree r–spanners exists for any chordal graph). Recall that

the problem whether a chordal graph admits a (one) multiplicative tree t-spanner is NP-complete for any $t > 3$ [21].

Peleg and Schäffer showed also in [97] that there are n-vertex chordal graphs for which any multiplicative 2-spanner will need to have at least $\Omega(n^{3/2})$ edges. This result leads to the following observation on collective additive tree 1-spanners of chordal graphs.

Observation 1 *There are n-vertex chordal graphs for which any system of collective additive tree 1–spanners will need to have at least $\Omega(\sqrt{n})$ spanning trees.*

Proof Indeed, the existence of a system of $o(\sqrt{n})$ collective additive tree 1–spanners for a chordal graph will lead to the existence of an additive 1-spanner (and hence, of a multiplicative 2-spanner) with $o(n^{3/2})$ edges.

3.2.4 Collective tree spanners in ξ-chordal graphs

A graph G is *ξ-chordal* if it does not contain any induced cycles of length greater than ξ. ξ-chordal graphs naturally generalize the class of chordal graphs. Chordal graphs are precisely the 3-chordal graphs.

Theorem 12 *The class of ξ-chordal graphs is $(1/2, \lfloor \xi/2 \rfloor)$–decomposable.*

Proof By Theorem 11 and since ξ-chordal graphs form a hereditary class of graphs, we need only to show that any ξ-chordal graph G admits a radius $\lfloor \xi/2 \rfloor$ acyclic covering. The existence of a radius $\lfloor \xi/2 \rfloor$ acyclic covering for G easily follows from a famous result of [66], which states that any ξ-chordal graph $G = (V, E)$ admits a special kind of *Robertson and Seymour tree-decomposition* [102]. That is, a tree $\mathcal{DT}(G)$, whose nodes are subsets of V, exists such that

$(1')$ $\bigcup\{S : S \text{ is a node of } \mathcal{DT}(G)\} = V$;

($2'$) for any edge xy of G there is a node S of $\mathcal{DT}(G)$ with $x, y \in S$;

($3'$) for any tree nodes X, Y, Z of $\mathcal{DT}(G)$, if Y is on the path from X to Z in $\mathcal{DT}(G)$ then $X \cap Z \subseteq Y$;

($4'$) $diam_G(S) \leq \lfloor \xi/2 \rfloor$ for each node S of $\mathcal{DT}(G)$.

The reader might notice a close similarity between these four properties and the four properties from the definition of a radius r acyclic covering. In fact, they are almost equivalent. Note that $diam_G(S) \leq \lfloor \xi/2 \rfloor$ implies $rad_G(S) \leq \lfloor \xi/2 \rfloor$. Let $\mathcal{S}(G) = \{S : S$ is a node of $\mathcal{DT}(G)\}$ and consider a hypergraph $H = (V, \mathcal{S}(G))$. We claim that for a family $\mathcal{S}(G)$ of subsets of V, properties (1),(2) and (3) are equivalent to properties ($1'$), ($2'$) and ($3'$). Indeed, since, by property ($3'$), $v \in X \cap Z$ implies v belongs to any Y on the path of $\mathcal{DT}(G)$ from X to Z, for any vertex $v \in V$ the elements of $\mathcal{S}(G)$ containing vertex v induce a subtree in $\mathcal{DT}(G)$. Hence, by definition, $H = (V, \mathcal{S}(G))$ is an acyclic hypergraph. Conversely, let for a graph G, a family $\mathcal{S}(G)$ of subsets of V satisfies properties (1),(2) and (3). Then, the acyclicity of the hypergraph $H = (V, \mathcal{S}(G))$ implies the existence of a tree T with node set $\mathcal{S}(G)$ such that for any vertex $v \in V$, the elements of $\mathcal{S}(G)$ containing v induce a subtree in T. Therefore, if two nodes X and Z of the tree T contain a vertex v then any node Y of T between X and Z must contain v, too.

A balanced separator of radius at most $\lfloor \xi/2 \rfloor$ of a ξ-chordal graph G on n vertices and m edges can be found in $O(n^3)$ time as follows. Use an $O(nm)$ time algorithm from [43] to construct a Robertson-Seymour tree-decomposition $\mathcal{DT}(G)$ of G (it will have at most n nodes [43]). Then define the family $\mathcal{S}(G) = \{S : S$ is a node of $\mathcal{DT}(G)\}$ and consider the 2-section graph $2SEC(H)$ of an acyclic hypergraph $H = (V, \mathcal{S}(G))$. $2SEC(H)$ can be constructed in at most $O(n^3)$ time. Using an algorithm from [71], find a balanced separator C of a chordal graph $2SEC(H)$ in $O(n^2)$ time. We know that C

is a maximal clique of $2SEC(H)$ and there must exist a set $S \in \mathcal{S}(G)$ which coincides with C (by conformality of H). As we showed earlier (see the proof of Theorem 11), $C = S$ is a balanced separator of radius at most $\lfloor \xi/2 \rfloor$ of G.

Thus, from Theorems 7 and 8, we conclude

Corollary 3 *Any ξ-chordal graph G with n vertices admits an additive $(2\lfloor \xi/2 \rfloor)$–spanner with at most $(n-1)\log_2 n$ edges, and such a sparse spanner can be constructed in $O(n^3 \log_2 n)$ time.*

Corollary 4 *Any ξ-chordal graph G with n vertices admits a system $\mathcal{T}(G)$ of at most $\log_2 n$ collective additive tree $(2\lfloor \xi/2 \rfloor)$–spanners, and such a system of spanning trees can be constructed in $O(n^3 \log_2 n)$ time.*

Note that there are ξ-chordal graphs which do not admit any radius r acyclic covering with $r < \lfloor \xi/2 \rfloor$. Consider, for example, the complement $\overline{C_6}$ of an induced cycle $C_6 = (a-b-c-d-e-f-a)$, which is a 4-chordal graph. A family $\mathcal{S}(\overline{C_6})$ consisting of one set $\{a, b, c, d, e, f\}$ gives a trivial radius $2 = \lfloor 4/2 \rfloor$ acyclic covering of $\overline{C_6}$, and a simple consideration shows that no radius 1 acyclic covering can exist for $\overline{C_6}$ (it is impossible, by simply adding new edges to $\overline{C_6}$, to get a chordal graph in which each maximal clique induces a radius one subgraph of $\overline{C_6}$). In next subsection we will show that yet an interesting subclass of 4-chordal graphs, namely the class of chordal bipartite graphs, does admit radius 1 acyclic coverings.

3.2.5 Collective tree spanners in chordal bipartite graphs

A bipartite graph $G = (X \cup Y, E)$ is *chordal bipartite* if it does not contain any induced cycles of length greater than 4 [72].

For a chordal bipartite graph G, consider a hypergraph $H = (X \cup Y, \{N[y] : y \in Y\})$. In what follows we show that H is an acyclic hypergraph.

Lemma 22 *The 2-section graph $2SEC(H)$ of H is chordal.*

Proof First notice that any $y \in Y$ is simplicial in $2SEC(H)$ by construction of H and definition of $2SEC(H)$. Assume now, by way of contradiction, that there is an induced cycle C_p of length p, $p \geq 4$, in $2SEC(H)$. Necessarily, all vertices of C_p are from part X of G, since C_p is induced and all vertices from Y are simplicial in $2SEC(H)$. Let $C_p = (x_1, x_2, \ldots, x_p, x_1)$. For any edge $x_i x_{i+1}$ of C_p (including the edge $x_p x_1$), since it is not an edge of G, there must exist a vertex y_i in Y such that both x_i and x_{i+1} are adjacent to y_i in G. Also, since C_p is induced in $2SEC(H)$, y_i is not adjacent to any other vertex of C_p. Therefore, a cycle $(x_1, y_1, x_2, y_2, \ldots, x_p, y_p, x_1)$ of G must be induced. But, since its length is $2p \geq 8$, a contradiction with G being a chordal bipartite graph arises.

Lemma 23 *The hypergraph $H = (X \cup Y, \{N[y] : y \in Y\})$ is conformal.*

Proof Let C be a clique of $2SEC(H)$ consisting of p vertices. First, note that, by definitions of H and $2SEC(H)$, the clique C can contain at most one vertex from Y. If C contains a vertex from Y (say $y \in C \cap Y$) then for all $v \in C \setminus \{y\}$, vy is an edge of G, and therefore $C \subseteq N[y]$ must hold. Let now $C \cap Y = \emptyset$. By induction on p we will show that there exists a vertex $y \in Y$ such that $C \subset N[y]$. Since G is connected, any vertex $x \in C \subseteq X$ has a neighbor in Y. Also, by definition of $2SEC(H)$, for any edge uv of $2SEC(H)$ with $u, v \in X$ there must exist a vertex y in Y adjacent to both u and v. Assume now, by induction, that each $p-1$ vertices of C have a common neighbor y in Y. Consider three different vertices a, b and c in C and three corresponding vertices a', b' and c' in Y such that $C \setminus \{a\} \subset N[a']$, $C \setminus \{b\} \subset N[b']$ and $C \setminus \{c\} \subset N[c']$. Since graph G cannot have any induced cycles of length 6, the cycle $(a - b' - c - a' - b - c' - a)$ of G cannot be induced. Without loss of generality, assume that a is adjacent to a' in G. But then, all p vertices of C are contained in $N[a']$.

Since chordal bipartite graphs form a hereditary class of graphs and for any chordal bipartite graph $G = (X \cup Y, E)$, a family $\{N[y] : y \in Y\}$ of subsets of $X \cup Y$ satisfies all four conditions of radius 1 acyclic covering, by Theorem 11 we have

Theorem 13 *The class of chordal bipartite graphs is* $(1/2, 1)$*-decomposable.*

Hence, by Theorems 7 and 8, we immediately conclude

Corollary 5 *Any chordal bipartite graph G with n vertices and m edges admits an additive 2–spanner with at most $(n-1)\log_2 n$ edges, and such a sparse spanner can be constructed in $O(nm \log_2 n)$ time.*

Corollary 6 *Any chordal bipartite graph G with n vertices and m edges admits a system $\mathcal{T}(G)$ of at most $\log_2 n$ collective additive tree 2–spanners, and such a system of spanning trees can be constructed in $O(nm \log_2 n)$ time.*

Recall that the problem whether a chordal bipartite graph admits a (one) multiplicative tree t-spanner is NP-complete for any $t > 3$ [22]. Also, any chordal bipartite graph G with n vertices admits an additive 4-spanner with at most $2n - 2$ edges which is constructable in linear time [32]. Again, it is interesting to know whether a system of constant number of collective additive tree 4–spanners exists for a chordal bipartite graph. We have the following observation on collective additive tree 1–spanners for chordal bipartite graphs.

Observation 2 *There are chordal bipartite graphs on $2n$ vertices for which any system of collective additive tree 1–spanners will need to have at least $\Omega(n)$ spanning trees.*

Proof Consider the complete bipartite graph $G = K_{n,n}$ on $2n$ vertices (which is clearly a chordal bipartite graph), and let $\mathcal{T}(G)$ be a system of μ collective additive tree 1-spanners of G. Then, for any two adjacent vertices x and y of G there must exist a

spanning tree T in $\mathcal{T}(G)$ such that $d_T(x,y) \leq 2$. If $d_T(x,y) = 2$ then a common neighbor z of x and y in G would form a triangle with vertices x and y, which is impossible for $G = K_{n,n}$. Hence, $d_T(x,y) = 1$ must hold. Thus, any edge xy of G is an edge of some tree $T \in \mathcal{T}(G)$. Since there are n^2 graph edges to cover by spanning trees from $\mathcal{T}(G)$, we conclude $\mu \geq n^2/(2n-1) > n/2$.

3.2.6 Collective tree spanners and routing labeling schemes

Routing is one of the basic tasks that a distributed network of processors must be able to perform. A *routing scheme* is a mechanism that can deliver packets of information from any vertex of the network to any other vertex. More specifically, a routing scheme is a distributed algorithm. Each processor in the network has a routing daemon running on it. This daemon receives packets of information and has to decide whether these packets have already reached their destination, and if not, how to forward them towards their destination. Each packet of information has a *header* attached to it. This header contains the address of the destination of the packet, and in some cases, some additional information that can be used to guide the routing of this message towards its destination. Each routing daemon has a local *routing table* at its disposal. It has to decide, based on this table and on the packet header only, whether to pass the packet to its host, or whether to forward the packet to one of its neighbors in the network.

The efficiency of a routing scheme is measured in terms of its *multiplicative stretch*, called *delay*, (or *additive stretch*, called *deviation*), namely, the maximum ratio (or surplus) between the length of a route, produced by the scheme for some pair of vertices, and their distance.

A straightforward approach for achieving the goal of guaranteeing optimal routes is to store a complete routing table in each vertex v in the network, specifying for each

destination u the first edge (or an identifier of that edge, indicating the output port) along some shortest path from v to u. However, this approach may be too expensive for large systems since it requires a total of $O(n^2 \log d)$ memory bits in an n-processor network with maximum degree d [64]. Thus, an important problem in large scale communication networks is the design of routing schemes that produce efficient routes and have relatively low memory requirements (see [7, 38, 55, 75, 96, 99, 108]).

This problem can be approached via localized techniques based on *labeling schemes* [96]. Informally speaking, the routing problem can be presented as requiring us to assign a label to every vertex of a graph. This label can contain the address of the vertex as well as the local routing table. The labels are assigned in such a way that at every source vertex v and given the address of any destination vertex u, one can decide the output port of an outgoing edge of v that leads to u. The decision must be taken locally in v, based solely on the label of v and the address of u.

Following [96], one can give the following formal definition. A family $\Re$ of graphs is said to *have an* $l(n)$ *routing labeling scheme* if there is a *function* L labeling the vertices of each n-vertex graph in $\Re$ with distinct labels of up to $l(n)$ bits, and there exists an efficient algorithm, called the *routing decision*, that given the label of a source vertex v and the label of the destination vertex (the header of the packet), decides in time polynomial in the length of the given labels and using only those two labels, whether this packet has already reached its destination, and if not, to which neighbor of v to forward the packet. Thus, the goal is, for a family of graphs, to find routing labeling schemes with small stretch factor, relatively short labels and fast routing decision.

To obtain routing schemes for general graphs that use $o(n)$-bit label for each vertex, one has to abandon the requirement that packets are always routed on shortest paths, and settle instead for the requirement that packets are routed on paths with relatively small stretch [7, 8, 38, 55, 99, 108]. A delay 3 scheme that uses labels of size $\tilde{O}(n^{2/3})$

was obtained in [38], and a delay 5 scheme that uses labels of size $\tilde{O}(n^{1/2})$ was obtained in [55].[1] Recently, authors of [108] further improved these results. They presented a routing scheme that uses only $\tilde{O}(n^{1/2})$ bits of memory at each vertex of an n-vertex graph and has delay 3. Note that, each routing decision takes constant time in their scheme, and the space is optimal, up to logarithmic factors, in the sense that every routing scheme with delay < 3 must use, on some graphs, routing tables of total size $\Omega(n^2)$, and hence $\Omega(n)$ at some vertex (see [62, 65, 68]).

There are many results on optimal (with delay 1) routing schemes for particular graph classes, including complete graphs, grids (alias meshes), hypercubes, complete bipartite graphs, unit interval and interval graphs, trees and 2-trees, rings, tori, unit circular-arc graphs, outerplanar graphs, and squaregraphs. All those graph families admit optimal routing schemes with $O(d \log n)$ labels and $O(\log d)$ routing decision. These results follow from the existence of special so called *interval routing* schemes for those graphs. We will not discuss details of this scheme here; for precise definitions and an overview of this area, we refer the reader to [64].

Observe that in interval routing schemes the local memory requirement increases with the degree of the vertex. Routing labeling schemes aim at overcoming the problem of large degree vertices. In [63], a shortest path routing labeling scheme for trees of arbitrary degree and diameter is described that assigns each vertex of an n-vertex tree a $O(\log^2 n/\log\log n)$-bit label. Given the label of a source vertex and the label of a destination it is possible to compute, in constant time, the neighbor of the source that heads in the direction of the destination. A similar result was independently obtained also in [108]. This result for trees was recently used in [42, 43] to design interesting low deviation routing schemes for chordal graphs and general ξ-chordal graphs. [42] describes a routing labeling scheme of deviation 2 with labels of size $O(\log^3 n/\log\log n)$

[1] Here, $\tilde{O}(f)$ means $O(f \text{ polylog } n)$.

bits per vertex and $O(1)$ routing decision for chordal graphs. [43] describes a routing labeling scheme of deviation $2\lfloor \xi/2 \rfloor$ with labels of size $O(\log^3 n)$ bits per vertex and $O(\log \log n)$ routing decision for the class of ξ-chordal graphs.

Our collective additive tree spanners give much simpler and easier to understand means of constructing compact and efficient routing labeling schemes for all (α, r)-decomposable graphs. We simply reduce the original problem to the problem on trees.

Let G be an (α, r)-decomposable graph and let $\mathcal{T}(G) = \{T^1, T^2, \ldots, T^\mu\}$ ($\mu \leq O(\log_2 n)$) be a system of μ collective additive tree $2r$-spanners of G. We can preprocess each tree T^i using the $O(n \log_2 n)$ algorithm from [63] and assign to each vertex v of G a tree-label $L^i(v)$ of size $O(\log^2 n / \log \log n)$ bits associated with the tree T^i. Then we can form a label $L(v)$ of v of size $O(\log^3 n / \log \log n)$ bits by concatenating the μ tree-labels. We store in $L(v)$ also the string A_v of length $O(\log^2 n)$ bits described in Subsection 3.2.2. Thus, $L(v) := A_v \circ L^1(v) \circ \ldots \circ L^\mu(v)$.

Now assume that a vertex v wants to send a message to a vertex u. Given the labels $L(v)$ and $L(u)$, v first uses their substrings A_v and A_u to find in $\log_2 n$ time an index i such that for tree $T^i \in \mathcal{T}(G)$, $d_{T^i}(v, u) \leq d_G(v, u) + 2r$ holds. Then, v extracts from $L(u)$ the substring $L^i(u)$ and forms a header of the message $H(u) := i \circ L^i(u)$. Now, the initiated message with the header $H(u) = i \circ L^i(u)$ is routed to the destination using the tree T^i: when the message arrives at an intermediate vertex x, vertex x using own substring $L^i(x)$ and the string $L^i(u)$ from the header makes a constant time routing decision.

Thus, the following result is true.

Theorem 14 *Each (α, r)-decomposable graph with n vertices and m edges admits a routing labeling scheme of deviation $2r$ with addresses and routing tables of size $O(\log^3 n / \log \log n)$ bits per vertex. Once computed by the sender in $\log_2 n$ time, headers never*

Graph class	Scheme construction time	Addresses and routing tables (bits per vertex)	Message initiation time	Routing decision time	Deviation
Chordal	$O(m \log_2 n + n \log_2^2 n)$	$O(\log^3 n / \log\log n)$	$\log_2 n$	$O(1)$	2
Chordal bipartite	$O(nm \log_2 n)$	$O(\log^3 n / \log\log n)$	$\log_2 n$	$O(1)$	2
Cocomparability	$O(m \log_2 n + n \log_2^2 n)$	$O(\log^3 n / \log\log n)$	$\log_2 n$	$O(1)$	2
ξ-Chordal	$O(n^3 \log_2 n)$	$O(\log^3 n / \log\log n)$	$\log_2 n$	$O(1)$	$2\lfloor \xi/2 \rfloor$
Circular-arc	$O(n \log_2 n + m)$	$O(\log^2 n)$	$O(1)$	$O(1)$	2

Table 3.1: Routing labeling schemes obtained for special graph classes via collective additive tree spanners

change. Moreover, the scheme is computable in $O((p(n) + t(n) + m + n \log_2 n) \log_2 n)$ *time, and the routing decision is made in constant time per vertex, where* $p(n)$ *is the time needed to find a balanced and bounded radius separator* S *and* $t(n)$ *is the time needed to find a central vertex for* S*.*

Projecting this theorem to the particular graph classes considered in this chapter, we obtain the following results summarized in Table 3.1. For circular-arc graphs, the labels are of size $O(\log^2 n)$ bits per vertex since this size labels are needed to decide in constant time which tree T or T_u is good for routing for given source x and destination y. We will choose tree $T' \in \{T, T_u\}$ such that $d_{T'}(x, y) = \min\{d_T(x, y), d_{T_u}(x, y)\}$. According to [96], in $O(n \log_2 n)$ total time the vertices of an n-vertex tree T can be labeled with labels of up to $O(\log^2 n)$ bits such that given two labels of two vertices x, y of T, it is possible to compute in constant time the distance $d_T(x, y)$, by merely inspecting the labels of x and y.

3.2.7 Extension to the weighted graphs

Although in our previous discussions graph G assumed (for simplicity) to be unweighted, the obtained results, in slightly modified form, are true even for weighted graphs.

Let $G = (V, E, \mathsf{w})$ be a *weighted graph* with the weight function $\mathsf{w} : E \to R^+$. In a weighted graph G, the *length of a path* is the sum of the weights of edges participating in the path. The *distance* $d_G(x, y)$ between vertices x and y is the length of a shortest length path connecting vertices x and y.

It is easy to see that, if in Sections 3.2 we consider shortest path trees instead of BFS-trees, interpret r as an upper bound on the weighted radius of a balanced separator $S \subseteq V$, and denote the maximum edge weight by w, then the following corollaries from the previous results are true.

i. Any weighted (α, r)–decomposable graph with n vertices, where r is an upper bound on the weighted radius of a balanced separator, admits a system of at most $\log_{1/\alpha} n$ collective additive tree $2r$–spanners.

ii. Any weighted ξ-chordal graph with n vertices admits a system of at most $\log_2 n$ collective additive tree $(2\lfloor c/2 \rfloor \mathsf{w})$–spanners.

iii. Any weighted chordal, chordal bipartite, or cocomparability graph with n vertices admits a system of at most $\log_2 n$ collective additive tree $2\mathsf{w}$–spanners.

3.3 (α, γ, r)-Decomposable graphs and their collective tree spanners

In Section 3.2, we introducted (α, r)-decomposable graphs. In this section, we introduce another class of graphs – (α, γ, r)-decomposable graphs. As before, let α be a positive real number smaller than 1, γ be a positive integer and r be a non-negative real number. We say that an n-vertex graph $G = (V, E)$ is (α, γ, r)–*decomposable* if there is

a separator $S \subseteq V$, such that the following three conditions hold:

Balanced Separator condition - the removal of S from G leaves no connected component with more than αn vertices;

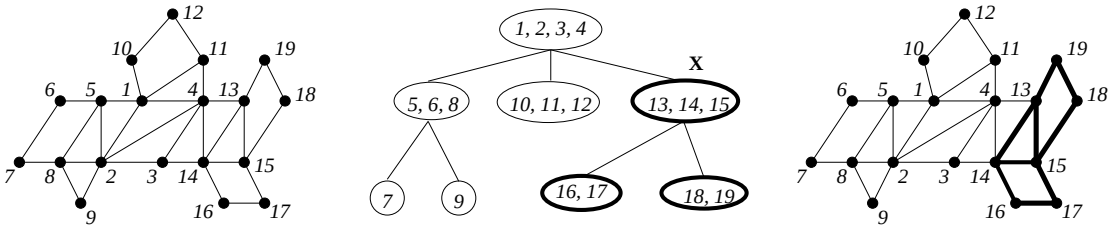
Bounded r-Dominating Set condition - there exists a subset $D \subseteq V$ such that $|D| \leq \gamma$ and for any vertex $u \in S$, $d_G(u, D) \leq r$ (we say that D *r-dominates* S);

Hereditary Family condition - each connected component of the graph, obtained from G by removing vertices of S, is also an (α, γ, r)–decomposable graph.

Note that, by definition, any graph $G = (V, E)$ having an r-dominating set (for V) of size at most γ is (α, γ, r)–decomposable, for any positive $\alpha < 1$.

Using these three conditions, one can construct for any (α, γ, r)-decomposable graph G a *(rooted) balanced decomposition tree* $\mathcal{BT}(G)$ as follows. If G has an r-dominating set of size at most γ, then $\mathcal{BT}(G)$ is a one node tree. Otherwise, find a balanced separator S with bounded r-dominating set in G, which exists according to the first and second conditions. Let $G_1, G_2, \ldots, G_p$ be the connected components of the graph $G \setminus S$ obtained from G by removing vertices of S. For each graph G_i $(i = 1, \ldots, p)$, which is (α, γ, r)-decomposable by the Hereditary Family condition, construct a balanced decomposition tree $\mathcal{BT}(G_i)$ recursively, and build $\mathcal{BT}(G)$ by taking S to be the root and connecting the root of each tree $\mathcal{BT}(G_i)$ as a child of S. Clearly, the nodes of $\mathcal{BT}(G)$ represent a partition of the vertex set V of G into *clusters* $S_1, S_2, \ldots, S_q$, each of them having in G an r-dominating set of size at most γ. For a node X of $\mathcal{BT}(G)$, denote by $G(\downarrow X)$ the (connected) subgraph of G induced by vertices $\cup\{Y : Y \text{ is a descendent of } X \text{ in } \mathcal{BT}(G)\}$ (here we assume that X is a descendent of itself). See Figure 3.3 for an illustration.

It is easy to see that a balanced decomposition tree $\mathcal{BT}(G)$ of a graph G with n vertices and m edges has depth at most $\log_{1/\alpha} n$, which is $O(\log_2 n)$ if α is a constant.



(a) (b) (c)

Figure 3.3: (a) A graph G, (b) its balanced decomposition tree $\mathcal{BT}(G)$ and (c) an induced subgraph $G(\downarrow X)$ of G.

Moreover, assuming that a special balanced separator (mentioned above) can be found in polynomial, say $p(n)$, time, the tree $\mathcal{BT}(G)$ can be constructed in $O((p(n)+m)\log_{1/\alpha} n)$ total time.

Consider now two arbitrary vertices x and y of an (α, γ, r)-decomposable graph G and let $S(x)$ and $S(y)$ be the nodes of $\mathcal{BT}(G)$ containing x and y, respectively. Let also $NCA_{\mathcal{BT}(G)}(S(x), S(y))$ be the nearest common ancestor of nodes $S(x)$ and $S(y)$ in $\mathcal{BT}(G)$ and $(X_0, X_1, \ldots, X_t)$ be the path of $\mathcal{BT}(G)$ connecting the root X_0 of $\mathcal{BT}(G)$ with $NCA_{\mathcal{BT}(G)}\ (S(x), S(y)) = X_t$ (in other words, $X_0, X_1, \ldots, X_t$ are the common ancestors of $S(x)$ and $S(y)$). The following lemmata are crucial to our subsequent results.

Lemma 24 *Any path $P^G_{x,y}$, connecting vertices x and y in G, contains a vertex from $X_0 \cup X_1 \cup \cdots \cup X_t$.*

Let $SP^G_{x,y}$ be a shortest path of G connecting vertices x and y, and let X_i be the node of the path $(X_0, X_1, \ldots, X_t)$ with the smallest index such that $SP^G_{x,y} \cap X_i \neq \emptyset$ in G. Then, the following lemma holds.

Lemma 25 *We have $d_G(x, y) = d_{G'}(x, y)$, where $G' := G(\downarrow X_i)$.*

Let D_i be an r-dominating set of X_i in $G' = G(\downarrow X_i)$ of size at most γ. For the graph G', consider a set of $|D_i|$ *Shortest-Path-trees (SP-trees)* $\mathcal{T}(D_i)$, each rooted at a (different) vertex from D_i. Then, there is a tree $T' \in \mathcal{T}(D_i)$ which has the following distance property with respect to those vertices x and y.

Lemma 26 *For those vertices $x, y \in G(\downarrow X_i)$, there exits a tree $T' \in \mathcal{T}(D_i)$ such that $d_{T'}(x,y) \leq d_G(x,y) + 2r$.*

Proof We know, by Lemma 25, that a shortest path $SP^G_{x,y}$, intersecting X_i and not intersecting any X_l ($l < i$), lies entirely in $G' = G(\downarrow X_i)$. Let x' be a vertex of $SP^G_{x,y} \cap X_i$, and denote by l_1 the distance in $SP^G_{x,y}$ between x and x' and by l_2 the distance in $SP^G_{x,y}$ between x' and y. Since $SP^G_{x,y}$ is a shortest path of G, we have

$$d_G(x,y) = d_{G'}(x,y) = l_1 + l_2. \tag{3.1}$$

Since D_i is an r-dominating set of X_i in G', there exists a vertex $c \in D_i$ such that $d_{G'}(c,x') \leq r$. Consider any Shortest-Path-tree T' of G' rooted at c. We have $d_{T'}(c,x) = d_{G'}(c,x) \leq d_{G'}(c,x') + d_G(x',x) \leq r + l_1$. Similarly, $d_{T'}(c,y) \leq r + l_2$. By triangle inequality, we have

$$d_{T'}(x,y) \leq d_{T'}(c,x) + d_{T'}(c,y) \leq (r + l_1) + (r + l_2). \tag{3.2}$$

Combining (3.1) and (3.2), we obtain $d_{T'}(x,y) \leq d_G(x,y) + 2r$. □

Let now $B^i_1, \ldots, B^i_{p_i}$ be the nodes on depth i of the tree $\mathcal{BT}(G)$ and let $D^i_1, \ldots, D^i_{p_i}$ be the corresponding r-dominating sets. For each subgraph $G^i_j := G(\downarrow B^i_j)$ of G ($i = 0, 1, \ldots, depth(\mathcal{BT}\ (G)),\ j = 1, 2, \ldots, p_i$), denote by $\mathcal{T}^i_j$ the set of SP-trees of graph G^i_j rooted at the vertices of D^i_j. Thus, for each G^i_j, we construct at most γ Shortest-Path-trees. We call them *local subtrees* of G. Lemma 26 implies

Theorem 15 *Let G be an (α, γ, r)-decomposable graph, $\mathcal{BT}(G)$ be its balanced decomposition tree and $\mathcal{LT}(G) = \{T \in \mathcal{T}_j^i : i = 0, 1, \ldots, depth(\mathcal{BT}(G)), j = 1, 2, \ldots, p_i\}$ be its set of local subtrees. Then, for any two vertices x and y of G, there exists a local subtree $T' \in \mathcal{T}_{j'}^{i'} \subseteq \mathcal{LT}(G)$ such that*

$$d_{T'}(x, y) \leq d_G(x, y) + 2r.$$

This theorem leads to two import results for the class of (α, γ, r)-decomposable graphs. Let G be an (α, γ, r)-decomposable graph with n vertices and m edges, $\mathcal{BT}(G)$ be its balanced decomposition tree and $\mathcal{LT}(G)$ be the family of its local subtrees (defined above). Consider a graph H obtained by taking the union of all local subtrees of G (by putting all of them together), i.e.,

$$H := \bigcup\{T : T \in \mathcal{T}_j^i \subseteq \mathcal{LT}(G)\} = (V, \cup\{E(T) : T \in \mathcal{T}_j^i \subseteq \mathcal{LT}(G)\}).$$

Clearly, H is a spanning subgraph of G and for any two vertices x and y of G, $d_H(x, y) \leq d_G(x, y) + 2r$ holds. Also, since for any level i ($i = 0, 1, \ldots,$ $depth(\mathcal{BT}(G))$) of balanced decomposition tree $\mathcal{BT}(G)$, the corresponding graphs $G_1^i, \ldots, G_{p_i}^i$ are pairwise vertex-disjoint and $|\mathcal{T}_j^i| \leq \gamma$ ($j = 1, 2, \ldots, p_i$), the union $\bigcup\{T : T \in \mathcal{T}_j^i, j = 1, 2, \ldots, p_i\}$ has at most $\gamma(n-1)$ edges. Therefore, H has at most $\gamma(n-1)\log_{1/\alpha} n$ edges in total. Thus, we have proven the following result.

Theorem 16 *Any (α, γ, r)-decomposable graph G with n vertices admits an additive $2r$-spanner with at most $\gamma(n-1)\log_{1/\alpha} n$ edges.*

Let $\mathcal{T}_j^i := \{T_j^i(1), T_j^i(2), \ldots, T_j^i(\gamma-1), T_j^i(\gamma)\}$ be the set of SP-trees of graph G_j^i rooted at the vertices of D_j^i. Here, if $p := |D_j^i| < \gamma$ then we can set $T_j^i(k) := T_j^i(p)$ for any k, $p+1 \leq k \leq \gamma$. By arbitrarily extending each forest $\{T_1^i(q), T_2^i(q), \ldots, T_{p_i}^i(q)\}$ ($q \in \{1, \ldots, \gamma\}$) to a spanning tree $T^i(q)$ of the graph G we can construct at most γ

spanning trees of G for each level i ($i = 0, 1, \ldots, depth(\mathcal{BT}(G))$) of the decomposition tree $\mathcal{BT}(G)$. Totally, this will result into at most $\gamma \times depth(\mathcal{BT}(G))$ spanning trees $\mathcal{T}(G) := \{T^i(q) : i = 0, 1, \ldots, depth(\mathcal{BT}(G)), q = 1, \ldots, \gamma\}$ of the original graph G. Thus, from Theorem 15, we have the following.

Theorem 17 *Any (α, γ, r)-decomposable graph G with n vertices and m edges admits a system $\mathcal{T}(G)$ of at most $\gamma \log_{1/\alpha} n$ collective additive tree $2r$-spanners. Moreover, such a system $\mathcal{T}(G)$ can be constructed in $O((p(n)+\gamma(m+n\log n))\log_{1/\alpha} n)$ time, where $p(n)$ is the time needed to find a balanced separator S and its r-dominating set D ($|D| \leq \gamma$) in an (α, γ, r)-decomposable graph.*

From Theorem 17, results of [63, 108] and [52] we conclude.

Corollary 7 *Every (α, γ, r)-decomposable graph G with n vertices admits a routing labeling scheme of deviation $2r$ with addresses and routing tables of size $O(\gamma \log_{1/\alpha} n \log^2 n / \log\log n)$ bits per vertex. Once computed by the sender in $\gamma \log_{1/\alpha} n$ time, headers never change, and the routing decision is made in constant time per vertex.*

3.3.1 Graphs having balanced separators of bounded size

In this subsection we consider graphs that have balanced separators of bounded size. To see that planar graphs are $(2/3, \sqrt{6n}, 0)$-decomposable, we recall the following theorem from [87].

Theorem 18 *[87] Let G be any n-vertex planar graph. Then the vertices of G can be partitioned into three sets A, B, C, such that no edge joins a vertex in A with a vertex in B, neither A nor B has more than 2/3n vertices, and C contains no more than $2\sqrt{2n}$ vertices. Furthermore A, B, C can be found in $O(n)$ time.*

Later, Djidjev [40] improved the constant $2\sqrt{2}$ to $\sqrt{6}$. Obviously, every connected component of $G \setminus C$ is still a planar graph. This theorem was extended in [2, 41, 70] to

bounded genus graphs: a graph G with genus at most g admits a separator C of size $O(\sqrt{gn})$ such that any connected component of $G \setminus C$ contains at most $2n/3$ vertices. Moreover, such a balanced separator C can be found in $O(n+g)$ time [2]. Evidently, each connected component of $G \setminus C$ has genus bounded by g, too. Hence, the following results follow.

Theorem 19 *Every n–vertex planar graph is $(2/3, \sqrt{6n}, 0)$-decomposable. Every n–vertex graph with genus at most g is $(2/3, O(\sqrt{gn}), 0)$-decomposable.*

There is another extension of Theorem 19, namely, to the graphs with an excluded minor [3]. A graph H is a *minor* of a graph G if H can be obtained from a subgraph of G by contracting edges. By an H-minor one means a minor of G isomorphic to H. Thus the Pontryagin-Kuratowski-Wagner Theorem asserts that planar graphs are those without K_5 and $K_{3,3}$ minors. The following result was proven in [3].

Theorem 20 *[3] Let G be an n-vertex graph and H be an h-vertex graph. If G has no H-minor, then the vertices of G can be partitioned into three sets A, B, C, such that no edge joins a vertex in A with a vertex in B, neither A nor B has more than 2/3n vertices, and C contains no more than $\sqrt{h^3 n}$ vertices. Furthermore A, B, C can be found in $O(\sqrt{hn}(n+m))$ time, where m is the number of edges in G.*

Since induced subgraphs of an H-minor free graph are H-minor free, we conclude.

Theorem 21 *Let G be an n-vertex graph and H be an h-vertex graph. If G has no H-minor, then G is $(2/3, \sqrt{h^3 n}, 0)$-decomposable.*

Now we turn to graphs with bounded tree-width. The following theorem is true.

Theorem 22 *Every graph with tree-width at most k is $(1/2, k+1, 0)$-decomposable.*

Proof It is well known that if $tw(G) = k$ for a graph $G = (V, E)$, then G can be transformed, by adding new edges, to a chordal graph $G^+ = (V, E^+)$ such that the maximum clique of G^+ is of size $k+1$ (see, e.g., [16, 78]). Moreover, if k is a constant, then the chordal graph G^+ can be constructed in at most $O(|V| + |E^+|)$ time [16, **?**]. In [71] it was shown that every n-vertex chordal graph Γ contains a maximal clique C such that if the vertices in C are deleted from Γ, every connected component in the graph induced by any remaining vertices is of size at most $n/2$. Moreover, according to [71], for any chordal graph on n vertices and m edges, such a separating clique C can be found in $O(n+m)$ time. Applying this result to an n-vertex chordal graph G^+, we obtain a set $S \subseteq V$ of at most $k+1$ vertices such that each connected component of $G^+ \setminus S$ will have at most $n/2$ vertices. Since G is a spanning subgraph of G^+, any connected component of $G \setminus S$ will have at most $n/2$ vertices, too.

Thus, any graph G with $tw(G) = k$ has a balanced separator consisting of at most $k+1$ vertices. Since induced subgraphs of a graph with tree-width at most k have also tree-width at most k (see, e.g., [16, 78]), the result follows. □

Table 3.2 summarizes the results on collective additive tree spanners of graphs having balanced separators of bounded size. The results are obtained by combining Theorem 17 with Theorems 19, 21 and 22. Note that, for planar graphs, the number of trees in the collection is at most $O(\sqrt{n})$ (not $\sqrt{6n}\log_{3/2} n$ as would follow from Theorem 17). This number can be obtained by solving the recurrent formula $\mu(n) = \sqrt{6n} + \mu(2/3n)$. Similar argument works for other two families of graphs.

Those systems of collective tree spanners described in Table 3.2 can be constructed in $O((n + \sqrt{n}(m + n\log n))\log n) = O(n^{3/2}\log^2 n)$ time for planar graphs, in $O((n + g + \sqrt{gn}(m + n\log n))\ \log n) = O(n^{3/2}g^{1/2}\ \log^2 n)$ time for graphs with genus g, in $O((\sqrt{hn}m + \sqrt{h^3 n}(m + n\log n))\log n) = O(h^{3/2}n^{1/2}(m\log n + n\log^2 n))$ time for graphs

Graph class	Number of trees in the collection, μ	Additive str. factor, r	Construction time
Planar graphs	$O(\sqrt{n})$	0	$O(n^{3/2}\log^2 n)$
Graphs with genus g	$O(\sqrt{gn})$	0	$O(n^{3/2}g^{1/2}\log^2 n)$ $\log^2 n)$
Graphs without an h-vertex minor	$O(\sqrt{h^3 n})$	0	$O(h^{3/2}n^{1/2}(m\log n$ $+n\log^2 n))$
Graphs with tree-width $k-1$	$k\log_2 n$	0	$O((n^2+km$ $+kn\log n)\log n)$

Table 3.2: Collective additive tree spanners of n-vertex m-edge graphs having balanced separators of bounded size.

without an h-vertex minor, and in $O((n^2+km+kn\log n)\log n)$ time for graphs with tree-width at most $k-1$ (for any constant $k \geq 2$).

Note that, any shortest path routing labeling scheme in n-vertex planar graphs requires at least $\Omega(\sqrt{n})$-bit labels [1]. Hence, by Corollary 7, there must exist n-vertex planar graphs, for which any system of collective additive tree 0-spanners needs to have at least $\Omega(\sqrt{n}\log\log n/\log^2 n)$ trees. We conclude this subsection with another lower bound, which follows from a result in [34]. Recall that all outerplanar graphs have tree-width at most 2.

Proposition 1 *No system of constant number of collective additive tree r-spanners can exist for outerplanar graphs, for any constant $r \geq 0$.*

3.3.2 Graphs with bounded clique-width

In this subsection we will prove that each graph with clique-width at most k is $(2/3, k, \mathsf{w})$-decomposable. Recall that w denotes the maximum edge weight in a graph G, i.e., $\mathsf{w} := max\{w(e) : e \in E(G)\}$.

Theorem 23 *Every graph with clique-width at most k is $(2/3, k, \mathsf{w})$-decomposable.*

Proof It was shown in [17] that the vertex set V of any graph $G = (V, E)$ with n vertices and clique-width $cw(G)$ at most k can be partitioned (in polynomial time) into two subsets A and $B := V \setminus A$ such that both A and B have no more than $2/3n$ vertices and A can be represented as the disjoint union of at most k subsets $A_1, \ldots, A_k$ (i.e., $A = A_1 \dot\cup \ldots \dot\cup A_k$), where each A_i ($i \in \{1, \ldots, k\}$) has the property that any vertex from B is either adjacent to all $v \in A_i$ or to no vertex in A_i.

Using this, we form a balanced separator S of G as follows. Initially set $S := \emptyset$, and in each subset A_i, arbitrarily choose a vertex v_i. Then, if $N(v_i) \cap B \neq \emptyset$, put v_i and $N(v_i) \cap B$ into S. Since for any edge $ab \in E$ with $a \in A$ and $b \in B$, vertex b must belong to S, we conclude that S is a separator of G, separating $A \setminus S$ from $B \setminus S$. Moreover, each connected component of $G \setminus S$ lies entirely either in A or in B and therefore has at most $2/3n$ vertices. By construction of S, any vertex $u \in B \cap S$ is adjacent to a vertex from $A' := A \cap S$. As $|A'| \leq k$ and w is an upper bound on any edge weight, we deduce that A' w-dominates S in G.

Thus, S is a balanced separator of G and is w-dominated by a set A' of cardinality at most k. To conclude the proof, it remains to recall that induced subgraphs of a graph with clique-width at most k have clique-width at most k, too (see, e.g., [37]), and therefore, by induction, the connected components of $G \setminus S$ induce $(2/3, k, \mathsf{w})$-decomposable graphs. □

Combining Theorem 23 with the results of Section 3.3, we obtain the following corollary.

Corollary 8 *Any graph with n vertices and clique-width at most k admits a system of at most $k \log_{3/2} n$ collective additive tree $2\mathsf{w}$-spanners, and such a system of spanning trees can be found in polynomial time.*

To complement the above result, we give the following lower bound.

Proposition 2 *There are (infinitely many) unweighted n-vertex graphs with clique-width at most 2 for which any system of collective additive tree 1-spanners will need to have at least* $\Omega(n)$ *spanning trees.*

Proof Consider the complete bipartite graph $G = K_{n,n}$ on $2n$ vertices. Since G does not have any induced P_4, it is a cograph. It is known that any cograph has clique-width at most 2 (see. e.g., [73]). We show that G will require at least $\Omega(n)$ spanning trees in any system of collective additive tree 1-spanners. Let $\mathcal{T}(G)$ be a system of μ collective additive tree 1-spanners of G. Then, for any two adjacent vertices x and y of G there must exist a spanning tree T such that $d_T(x, y) \leq 2$. If $d_T(x, y) = 2$ then a common neighbor z of x and y in G would form a triangle with vertices x and y, which is impossible for $G = K_{n,n}$. Hence, $d_T(x, y) = 1$ must hold. Thus, every edge xy of G is an edge of some tree $T \in \mathcal{T}(G)$. Since there are n^2 graph edges to cover by spanning trees from $\mathcal{T}(G)$, we conclude $\mu \geq n^2/(2n-1) > n/2$. □

3.3.3 Graphs with bounded chordality

In this subsection, we consider the class of ξ-chordal graphs and its subclasses. We show that every ξ-chordal graph is $(1/2, 1, \lfloor \xi/2 \rfloor \mathsf{w})$-, $(1/2, 5, \lfloor (\xi+2)/3 \rfloor \mathsf{w})$- and $(1/2, 4, (\lfloor \xi/3 \rfloor + 1)\mathsf{w})$-decomposable, every 4-chordal graph is $(1/2, 6, \mathsf{w})$-decomposable and every weakly chordal graph is $(1/2, 4, \mathsf{w})$-decomposable.

In what follows we will need a special ordering of the vertex set of a graph $G = (V, E)$, which refines well known BFS-ordering produced by a *breadth-first search*. *Lexicographic breadth-first search (LexBFS)*, was defined in first chapter. An ordering of the vertex set of a graph G generated by LexBFS we will call a *LexBFS-ordering* of G, and use σ to denote it. The number assigned to a vertex is called *LexBFS-ordering number*. For any vertex v, $\sigma(v)$ is used to denote its LexBFS-ordering number. For convenience, in the sequel, $\sigma(x) > \sigma(y)$ is simplified as $x > y$. The *father* of a vertex v is the vertex in

$N[v]$ which has the largest LexBFS-ordering number. $f(v)$ is used to denote the father of v. LexBFS may be seen to generate a rooted tree T with a vertex u as the root. The properties mentioned in section 2.1.1 will be used in what follows.

Arbitrary ξ-chordal graphs:

Here, we consider the class of ξ-chordal graphs, $\xi \geq 3$. We start with an easy consequence of a result from [52].

Theorem 24 *Every n-vertex ξ-chordal graph is $(1/2, 1, \lfloor \xi/2 \rfloor \mathsf{w})$-decomposable.*

Proof In 11, we showed that any ξ-chordal graph has a subset $S \subseteq V$ of vertices computable in $O(n^3)$ time such that any connected component of $G \setminus S$ has at most $n/2$ vertices and any two vertices x and y of S can be connected in G by a path with at most $\lfloor \xi/2 \rfloor$ edges. Since in our weighted case any edge has weight at most w, we conclude that in G any vertex x of S ($\lfloor \xi/2 \rfloor \mathsf{w}$)-dominates S. Hence, as induced subgraphs of ξ-chordal graphs are ξ-chordal, the result follows. □

Corollary 9 *Every n-vertex ξ-chordal graph admits a system of at most $\log_2 n$ collective additive tree $(2\lfloor \xi/2 \rfloor \mathsf{w})$-spanners, and such a system of spanning trees can be found in $O(n^3 \log n)$ time.*

In what follows we will show that every ξ-chordal graph with $k \geq 4$ is also $(1/2, 5, \lfloor (\xi + 2)/3 \rfloor \mathsf{w})$- and $(1/2, 4, (\lfloor k/3 \rfloor + 1)\mathsf{w})$-decomposable. To prove these, we first show that any graph has a special balanced separator S. Let $N(C) := \bigcup_{v \in C} N(v) \setminus C$ and $N[C] = N(C) \cup C$, for any set $C \subseteq V$.

Lemma 27 *Any graph G has a separator S such that each connected component of $G \setminus S$ contains at most $n/2$ vertices.*

Proof Let $\sigma = (v_1, v_2, \ldots, v_n)$ be a LexBFS-ordering of G and $B_i := \{v_i, v_{i+1}, \ldots, v_n\}$. Clearly, for any $i = 1, 2, \ldots, n-1$, B_i is connected. Let $C^*(i)$ be a largest (by number of vertices) connected component of $G \setminus B_i$. In what follows, i will be chosen to be the largest index such that $|V(C^*(i))| \leq n/2$. Evidently, $i \geq \lceil n/2 \rceil$ and, by maximality of i, a largest connected component $C^*(i+1)$ of graph $G \setminus B_{i+1}$ must have more than $n/2$ vertices. It is easy to see that if $C_1, C_2, \ldots, C_k, C^*(i+1)$ are the connected components of $G \setminus B_{i+1}$, then the connected components of $G \setminus B_i$ will be $C_1, C_2, \ldots, C_k, C_{k+1}, \ldots, C_{k+p}$, where $C_{k+1}, \ldots, C_{k+p}$ are the connected components of the subgraph of G induced by vertices of $C^*(i+1) \setminus \{v_i\}$.

Since $|V(C^*(i+1))| > n/2$, $|B_{i+1}| + \sum_{i=1}^{k} |V(C_i)| < n/2$ holds. Let $C' = V(C^*(i+1)), A = B_{i+1} \cap N(C')$ and $S = A \cup \{v_i\}$. Clearly, all connected components of $G \setminus S$ have at most $n/2$ vertices as they coincide with components $C_{k+1}, \ldots, C_{k+p}$ and the connected components of the subgraph of G induced by vertices of $C_1, C_2, \ldots, C_k$ and $B_{i+1} \setminus A$. □

From the proof of Lemma 27, one can easily design a procedure to find such a balanced separator S in at most $O(|V||E|)$ time. Our goal in this subsection is to show that in ξ-chordal graphs separator S has a small r_ξ-dominating set.

Let $G = (V, E)$ be a ξ-chordal graph with $\xi \geq 4$ and $\sigma = (v_1, v_2, \ldots, v_n)$ be a LexBFS-ordering of G (the *LexBFS-ordering number* of $v_i = \sigma(i)$ is $i = \sigma^{-1}(v_i)$). For any vertex $x \in V$, define $V_{>x} = \{u \in V : u > x\}$ and $G_{>x}$ to be a subgraph of G induced by $V_{>x}$. Let also $S = A \cup \{v_i\}$ be a separator of G computed as described in the proof of Lemma 27. That is, $C^*(i+1)$ is the largest connected components of $G \setminus B_{i+1}$ and $A = N(C^*(i+1)) \cap B_{i+1}$ (see Fig. 3.4 for an illustration). By the properties of LexBFS-orderings, the following observation clearly holds.

Proposition 3 *No vertex of $C^*(i+1)$ has a neighbor in $V_{>f(v_i)}$.*

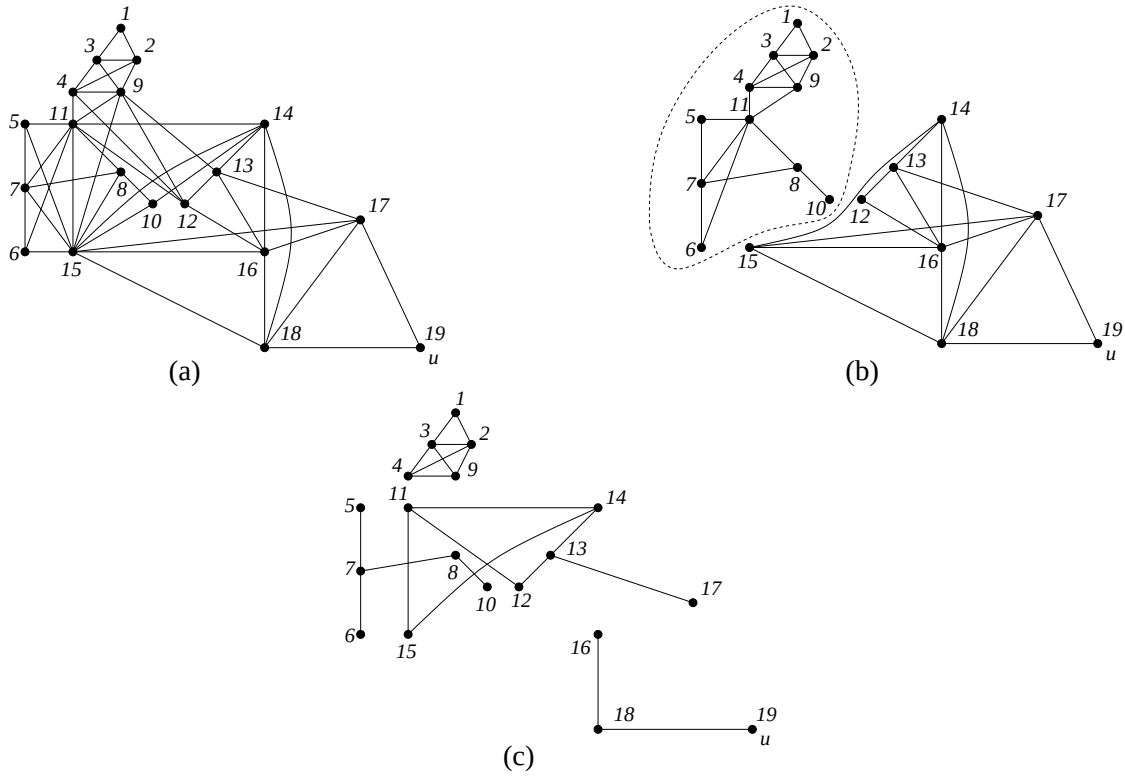


Figure 3.4: (a) A 4-chordal graph G with a LexBFS-ordering. (b) A largest connected component $C^*(12)$ of $G \setminus B_{12}$ (circled). A balanced separator $S = \{11, 12, 13, 14, 15\}$ and the connected components of $G \setminus S$.

We say that a vertex x has the level number $l(x)$ if $x \in L_{l(x)}(v_n)$. Since for any $y \in A$, $v_i < y \leq f(v_i)$ holds, all the vertices of S are either in $L_{l(v_i)}(v_n)$ or in $L_{l(v_i)-1}(v_n)$. Let $S_1 := S \cap L_{l(v_i)}(v_n)$ and $S_2 := S \cap L_{l(v_i)-1}(v_n)$.

Lemma 28 *There is a set D of at most five vertices in G such that D ($\lfloor(\xi+2)/3\rfloor$w)-dominates S. Moreover, if G is a ξ-chordal graph with $4 \leq \xi \leq 6$, then D consists of only four vertices.*

Proof Define vertices $f_0, \cdots, f_{a(v_i)}$ as follows: $f_0 := v_i, f_1 := f(v_i), \cdots, f_{a(v_i)} := f(f_{a(v_i)-1})$, where $a(v_i) := \min\{\lfloor(\xi+2)/3\rfloor, l(v_i)\}$. We claim that the set $\{f_1, f_{a(v_i)-1}, f_{a(v_i)}\}$ is a ($\lfloor(\xi+2)/3\rfloor$w)-dominating set for S_1.

If $a(v_i) = l(v_i)$ then $f_{a(v_i)} = v_n$ and trivially S_1 is $(\lfloor(\xi+2)/3\rfloor\mathsf{w})$-dominated by $f_{a(v_i)} = v_n$ since $S_1 \subseteq L_{a(v_i)}(v_n)$. Therefore, assume that $a(v_i) \neq l(v_i)$, and let x be an arbitrary vertex of $S_1 \setminus \{v_i\}$. Consider vertices $f'_0 := x, f'_1 := f(f'_0), \cdots, f'_{a(v_i)} := f(f'_{a(v_i)-1})$. If there is an index i $(0 \leq i < a(v_i))$ such that f_i coincides with f'_i or $f'_i f_i \in E(G)$ or $f'_i f_{i+1} \in E(G)$, then the distance between $f_{a(v_i)-1}$ (or $f_{a(v_i)}$) and x is at most $\lfloor(\xi+2)/3\rfloor\mathsf{w}$ and the claim clearly holds. Hence, we may assume that there is no such index i. Since $f_i < f'_i$ (by property (P3) of LexBFS-orderings), one concludes that $f_i f'_{i+1} \notin E(G)$, too, for any $i = 0, \cdots, a(v_i) - 1$.

Let $P_G(f_{a(v_i)}, f'_{a(v_i)})$ be an induced path between $f_{a(v_i)}$ and $f'_{a(v_i)}$ such that, if $(f_{a(v_i)},\ f'_{a(v_i)}) \notin E(G)$, then all its inner vertices are from levels L_j, $j \leq l(f_{a(v_i)}) - 1$. Let $P_G(f_1,\ f'_0)$ be an induced path of G obtained by concatenating the two paths $P_G(f_1, f_{a(v_i)}) := (f_1, \cdots, f_{a(v_i)})$, $P_G(f'_0, f'_{a(v_i)}) := (f'_0, \cdots, f'_{a(v_i)})$ with $P_G(f_{a(v_i)}, f'_{a(v_i)})$. Obviously, $P_G(f_1, f'_0)$ has at least $\lfloor(\xi+2)/3\rfloor - 1 + \lfloor(\xi+2)/3\rfloor + 1 = 2\lfloor(\xi+2)/3\rfloor$ edges. Let also $P'_G(f_1, f'_0)$ be an induced path between f_1 and f'_0 all inner vertices of which are from $C^*(i+1)$. By construction of S, we know that $x > v_i$. This and property (P3) of LexBFS-orderings imply that all inner vertices of $P_G(f_1, f'_0)$ are from $V_{>f(v_i)}$. By Proposition 3, no vertex from $V(P'_G(f_1, f'_0)) \setminus \{f_1, f'_0\}$ is adjacent to a vertex from $V(P_G(f_1, f'_0)) \setminus \{f_1, f'_0\}$. Now, by concatenating the two induced paths $P'_G(f_1, f'_0)$ and $P_G(f_1, f'_0)$, we obtain a chordless cycle in G. Since G is a ξ-chordal graph, the path $P'_G(f_1, f'_0)$ cannot have more than $\lfloor(\xi+2)/3\rfloor$ edges (otherwise, G will have an induced cycle with at least $\xi+1$ edges). Hence $d(f_1, x) \leq \lfloor(\xi+2)/3\rfloor\mathsf{w}$ and our claim that the set $\{f_1, f_{a(v_i)-1}, f_{a(v_i)}\}$ is a $(\lfloor(\xi+2)/3\rfloor\mathsf{w})$-dominating set for S_1 is proven.

Clearly, if G is a ξ-chordal graph with $4 \leq \xi \leq 6$, then $a(v_i)$ is at most $2 = \lfloor(\xi+2)/3\rfloor$. Therefore, in this case, S_1 is w-dominated by $f_{a(v_i)} = v_n$, if $a(v_i) = l(v_i) = 1$, or $(2\mathsf{w})$-dominated by $f_{a(v_i)-1} = f_1$ and $f_{a(v_i)} = f_2$, if $a(v_i) = \lfloor(\xi+2)/3\rfloor = 2$.

Let now v' be the vertex of S_2 with the smallest LexBFS-ordering number. Define $\mathrm{a}(v') := \min\{\lfloor(\xi+2)/3\rfloor, l(v_i)-1\}$. Let $f''_0 := v', f''_1 := f(f''_0), \cdots, f''_{a(v')} := f(f''_{a(v')-1})$. Let x be an arbitrary vertex in $S_2 \setminus \{v'\}$. Note that, by the definition of S_2, both v' and x have neighbors in $C^*(i+1)$. Since $C^*(i+1)$ is connected, there is an induced path $P'_G(v', x)$ all inner vertices of which are from $C^*(i+1)$. Using similar arguments as before, one can show that the set $\{v', f''_{a(v')-1}\}$ is a $(\lfloor(\xi+2)/3\rfloor\mathsf{w})$-dominating set for S_2.

Set $D := \{v', f''_{a(v')-1}\} \cup \{v_i, f_{a(v_i)-1}, f_{a(v_i)}\}$. Clearly, D is a $(\lfloor(\xi+2)/3\rfloor\mathsf{w})$-dominating set for S. This concludes the proof of the lemma. □

In a similar way we can prove

Lemma 29 *There is a set D' of at most four vertices in G such that D' is a $((\lfloor\xi/3\rfloor+1)\mathsf{w})$-dominating set for S.*

Proof Set $a(v_i) := \min\{\lfloor c/3\rfloor, l(v_i)\}$. Let $f_0 := v_i, f_1 := f(f_0), \cdots, f_{a(v_i)} := f(f_{a(v_i)-1})$. We claim that the set $\{f_1, f_{a(v_i)}\}$ is a $((\lfloor\xi/3\rfloor+1)\mathsf{w})$-dominating set of S_1.

If $a(v_i) = l(v_i)$, then $f_{a(v_i)} = v_n$ and claim clearly holds. So, assume $a(v_i) \neq l(v_i)$. Let x be an arbitrary vertex in $S_1 \setminus \{v_i\}$. Set $f'_0 := x, f'_1 := f(f'_0), \cdots, f'_{a(v_i)} := f(f'_{a(v_i)-1})$. If there is an index i, $0 \leq i \leq a(v_i)$, such that $f_i = f'_i$ or $f_i f'_i \in E(G)$ or $f_{i+1} f'_i \in E(G)$ (for $i < a(v_i)$), then we are done. Hence, we may assume that no such i exists. We have also $f_i f'_{i+1} \notin E(G)$, for any $i = 0, \cdots, a(v_i)-1$ since $f_i < f'_i$ holds by property (P3) of LexBFS-orderings. Let $P_G(f_{a(v_i)}, f'_{a(v_i)})$ be an induced path (of length at least 2) between $f_{a(v_i)}$ and $f'_{a(v_i)}$ all inner vertices of which are from levels L_j, $j \leq l(f_{a(v_i)}) - 1$. By concatenating the paths $P_G(f_1, f_{a(v_i)}) := (f_1, f_2, \cdots, f_{a(v_i)})$, $P_G(f'_0, f'_{a(v_i)}) := (f'_0, f'_1, \cdots, f'_{a(v_i)})$ with path $P_G(f_{a(v_i)}, f'_{a(v_i)})$, one gets an induced path with at least $(\lfloor\xi/3\rfloor - 1) + \lfloor\xi/3\rfloor + 2 = 2\lfloor\xi/3\rfloor + 1$ edges. Since $f_1 f'_0 \notin E(G)$, there must exist an induced path $P'_G(f_1, f'_0)$ between f_1 and f'_0 all inner vertices of which are from

$C^*(i+1)$. By Proposition 3 and the fact that $x > v_i$, we have also that no inner vertex of $P'_G(f_1, f'_0)$ is adjacent to inner vertices of $P_G(f_1, f'_0)$. Therefore, these two paths form an induced cycle. Since G is a ξ-chordal graph, $P'_G(f_1, f'_0)$ must have at most $\lfloor \xi/3 \rfloor + 1$ edges. This proves the claim.

Let now v' be the vertex with the smallest LexBFS-ordering number in S_2. Define $a(v') := \min\{\lfloor \xi/3 \rfloor, l(v_i) - 1\}$. Let $f''_0 := v', f''_1 := f(f''_0), \cdots, f''_{a(v')} := f(f''_{a(v')-1})$. Using similar arguments as before, one can show that the set $\{f''_0, f''_{a(v')}\}$ is a $((\lfloor \xi/3 \rfloor + 1)\mathsf{w})$-dominating set for S_2.

Set $D' := \{f_1, f_{a(v_i)}, f''_0, f''_{a(v')}\}$. Clearly, D' is a $((\lfloor \xi/3 \rfloor + 1)\mathsf{w})$-dominating set for S. This completes the proof. □

Clearly, for a given S, both sets D and D' can be found in linear time. Thus, we have proven the following results.

Theorem 25 *Let G be a ξ-chordal graph. Then, G is $(1/2, 4, \lfloor(\xi+2)/3\rfloor\mathsf{w})$-decomposable, if $4 \leq \xi \leq 6$, and is $(1/2, 5, \lfloor(\xi+2)/3\rfloor\mathsf{w})$- and $(1/2, 4, (\lfloor \xi/3 \rfloor + 1)\mathsf{w})$-decomposable, if $\xi > 6$.*

Corollary 10 *Let G be an n-vertex and m-edge ξ-chordal graph. If $\xi > 6$, then G admits a system of at most $5\log_2 n$ collective additive tree $(2\lfloor(\xi+2)/3\rfloor\mathsf{w})$-spanners and a system of at most $4\log_2 n$ collective additive tree $(2(\lfloor \xi/3 \rfloor + 1)\mathsf{w})$-spanners. If $4 \leq \xi \leq 6$, then G admits a system of at most $4\log_2 n$ collective additive tree $(2\lfloor(\xi+2)/3\rfloor\mathsf{w})$-spanners. Moreover, such systems of spanning trees can be found in $O(nm\log n)$ time.*

From Theorem 24 and Theorem 25 we conclude that any 3-chordal graph is $(1/2, 1, \mathsf{w})$-decomposable, any 4-chordal graph or 5-chordal graph is $(1/2, 1, 2\mathsf{w})$-decomposable, any 6-chordal graph is $(1/2, 1, 3\mathsf{w})$- and $(1/2, 4, 2\mathsf{w})$-decomposable, any 7-chordal graph is $(1/2, 1,\ 3\mathsf{w})$-decomposable, and any 8-chordal graph is $(1/2, 1, 4\mathsf{w})$- and $(1/2, 4, 3\mathsf{w})$-decomposable. In the next subsubsection we will show that the result for 4-chordal

graphs can be refined. In Table 3.3 we present our decomposition results for all ξ-chordal graphs.

4-Chordal graphs:

Here, we show that every 4-chordal graph is $(1/2, 6, \mathsf{w})$-decomposable and every weakly chordal graph is $(1/2, 4, \mathsf{w})$-decomposable.

Let $G = (V, E)$ be a 4-chordal graph and $\sigma = (v_1, v_2, \ldots, v_n)$ be a LexBFS-ordering of G. Let also $S = A \cup \{v_i\}$ be a separator of G computed as described in the proof of Lemma 27. That is, $C^*(i+1)$ is the largest connected components of $G \setminus B_{i+1}$ and $A = N(C^*(i+1)) \cap B_{i+1}$.

Denote by $\overline{C}_6$ the complement of an induced cycle C_6 on 6 vertices. First we will show that any 4-chordal graph not containing $\overline{C}_6$ as an induced subgraph is $(1/2, 4, \mathsf{w})$-decomposable. Clearly, these graphs contain all weakly chordal graphs.

Lemma 30 *If G is a 4-chordal graph not containing $\overline{C}_6$ as an induced subgraph, then there exists a set D of at most four vertices in G such that $S \subseteq N[D]$.*

Proof Let $A^- := \{w \in A : wv_i, wf(v_i) \notin E(G)\}$. We will show that there are at most two vertices a, b in G such that $A^- \subseteq N[\{a, b\}]$. Consider a vertex $w \in A^-$. Obviously, $w > v_i$. By properties (P2) and (P3) of LexBFS-orderings, $f(v_i) < f(w)$ must hold. By Proposition 3, one concludes that $w < f(v_i)$ holds. Let $x \in C^*(i+1)$ be a vertex from $N(w)$ which can be connected to v_i in $C^*(i+1)$ with minimum number of edges.

Claim 1. $f(v_i)x \in E(G)$.
Proof The claim can be proved by contradiction. Assume $f(v_i)x \notin E(G)$. Let $P = (v_i = u_0, u_1, \ldots, u_l = x)$ be a path between v_i and x in $C^*(i+1)$ with minimum number of edges. Clearly, $N(w) \cap P = \{x\}$. Let $u_{l'}$ be the vertex of P with largest index which is adjacent to $f(v_i)$. Then path $P_1 = (f(v_i), u_{l'}, u_{l'+1}, \ldots, u_l, w)$ is an induced path connecting $f(v_i)$ and w, and it consists of at least 3 edges. Since $f(v_i) < f(w)$, there

must be an induced path P_2 between $f(v_i)$ and w all inner vertices of which are from $V_{>f(v_i)}$. Moreover, no vertex from P_2 can be adjacent to any vertex from $P_1 \setminus \{f(v_i), w\}$. Since P_2 consists of at least 2 edges, by combining P_1 and P_2, one gets an induced cycle in G with at least 5 edges. As G is a 4-chordal graph, that is impossible. □(Claim)

Consider a layering $\{v_n\}, L_1(v_n), L_2(v_n), L_3(v_n), \ldots$ of graph G, where $L_i(v_n) = \{x \in V : x$ can be connected to v_n by a path with i edges but not by a path with $i-1$ edges$\}$. Since all the vertices in B_{i+1} have larger LexBFS-ordering numbers than v_i, by property (P1) of LexBFS-orderings, each vertex in A^- is either in level $L_{l(v_i)}(u_n)$ or in level $L_{l(v_i)-1}(u_n)$ (recall that $w < f(v_i)$ for any $w \in A^-$). Define $A_u = \{u : u \in A^- \cap L_{l(v_i)}(v_n)\}$ and $A_d = \{u : u \in A^- \cap L_{l(v_i)-1}(v_n)\}$. Set also $N_\downarrow(x) := N(x) \cap (L_{l(x)-1}(v_n) \cap V_{>f(v_i)})$ for any $x \in V$. Since for every vertex $w \in A^-$, $f(w) > f(v_i)$ holds, $N_\downarrow(w)$ is not empty for any $w \in A^-$. We have $l(v_i) > 1$, since otherwise, $f(v_i) = v_n$ and therefore w must be adjacent to or coincide with $f(v_i)$.

Claim 2. For any vertex $w \in A^-$, $N_\downarrow(w) \subseteq N_\downarrow(f(v_i))$ holds.

Proof Assume that the statement is not true. Then, one can find a vertex $w' \in N_\downarrow(w)$ such that $w' > f(v_i)$ and $w'f(v_i) \notin E(G)$. By Claim 1, there is a vertex x in $C^*(i+1)$ which is adjacent to both $f(v_i)$ and w. We know also that x is not adjacent to any vertex of $V_{>f(v_i)}$. We distinguish between two cases. First assume $w \in A_u$. There must exist an induced path $P_{f(v_i),w'}$ between $f(v_i)$ and w' such that its inner vertices are all from layers $L_s(v_n)$, $s \leq l(f(v_i)) - 1$. This path has at least 2 edges. Moreover, no inner vertex of $P_{f(v_i),w'}$ is adjacent to w or x. Therefore, paths $(f(v_i), x, w, w')$ and $P_{f(v_i),w'}$ will form a chordless cycle with at least 5 edges in G, which is impossible.

Assume now that $w \in A_d$. Since $w < f(v_i) < w'$ and $w'f(v_i) \notin E$ but $w'w \in E$, by property (P4) of LexBFS-orderings, there is a vertex $t > w'$ such that $tf(v_i) \in E(G)$ and $tw \notin E(G)$. Let $P_{t,w'}$ be an induced path connecting t and w' all inner vertices of which are from $\bigcup_{i \leq l(w')-1} L_i(v_n)$. $P_{t,w'}$ has at least one edge. Hence, the path $P_{t,w'}$

together with $(t, f(v_i), x, w, w')$ will form an induced cycle with at least 5 edges in G, which is impossible. □(Claim)

Claim 3. For any two vertices $w, z \in A_u$ or $w, z \in A_d$, sets $N_\downarrow(w)$ and $N_\downarrow(z)$ are comparable.

Proof The claim can be proved by contradiction. Assume $w, z \in A_u$ and $N_\downarrow(w)$ and $N_\downarrow(z)$ are not comparable. Then, there exist two vertices $w' \in N_\downarrow(w)$ and $z' \in N_\downarrow(z)$ such that $w'z, z'w \notin E(G)$. By Claim 2, we know $f(v_i)w', f(v_i)z' \in E(G)$. Let $x, y \in C^*(i+1)$ be two vertices such that $xw, xf(v_i), yz, yf(v_i) \in E$, the existence of which follows from Claim 1. As w', z' are from $V_{>f(v_i)}$ and x, y are from $C^*(i+1)$, there cannon be an edge between sets $\{x, y\}$ and $\{z', w'\}$.

First, we show that both wz and $w'z'$ must be in $E(G)$. Assume $w'z' \notin E(G)$. Let $P_{w,z}$ be an induced path between w and z such that all its inner vertices are from $G^*(i+1)$. $P_{w',z'}$ is used to denote an induced path between w' and z' such that its inner vertices are from $\bigcup_{i \leq l(w')-1} L_i(v_n)$. Clearly, the inner vertices of $P_{w',z'}$ are not adjacent to any vertex from $P_{w,z}$. Since $P_{w,z}$ has at least one edge and $P_{w',z'}$ has at least 2 edges, $P_{w,z}, ww', zz'$ and $P_{w',z'}$ will form a hole in G, which is impossible. This proves that $w'z'$ must be in $E(G)$. Similarly, if $wz \notin E(G)$, then $P_{w,z}$ has at least 2 edges. Moreover, any inner vertex of $P_{w,z}$ is adjacent neither to w' nor to z'. Hence, $P_{w,z}, ww', w'z', z'z$ form an induced cycle with at least 5 edges in G, which is impossible. Thus, both wz and $w'z'$ are in $E(G)$.

Second, we claim that neither wy nor zx is in $E(G)$. If $wy \in E(G)$, then since $wz, w'z' \in E(G)$, vertices w, y, z, w', z' and $f(v_i)$ would give an induced $\overline{C}_6$ which is also forbidden in G. In a similar way, one can show that $zx \in E(G)$ is impossible.

It is easy to see now that vertices $w, z, y, f(v_i), w'$ form an induced cycle with 5 edges in G. A contradiction obtained proves that $N_\downarrow(w)$ and $N_\downarrow(z)$ are comparable for any $w, z \in A_u$. When $w, z \in A_d$, one can show that $N_\downarrow(w)$ and $N_\downarrow(z)$ are comparable in a

similar way. □(Claim)

Claim 3 ensures that there can be found two vertices a and b in G such that $a \in \bigcap_{w \in A_u} N_{\downarrow}(w)$ and $b \in \bigcap_{w \in A_d} N_{\downarrow}(w)$. Hence, $A^- = A_d \bigcup A_u$ is completely contained in $N[\{a, b\}]$, implying $S \subseteq N[\{v_i, f(v_i), a, b\}]$. □

Hence, we have the following results.

Theorem 26 *Let G be a 4-chordal graph not containing $\overline{C}_6$ as an induced subgraph. Then G is $(1/2, 4, \mathsf{w})$-decomposable.*

Corollary 11 *Any n-vertex m-edge 4-chordal graph G not containing $\overline{C}_6$ as an induced subgraph (in particular, any weakly chordal graph) admits a system of at most $4 \log_2 n$ collective additive tree $(2\mathsf{w})$-spanners. Moreover, such a system of spanning trees can be constructed in $O(nm \log n)$ time.*

Note that the class of weakly chordal graphs properly contains such known classes of graphs as interval graphs, chordal graphs, chordal bipartite graphs, permutation graphs, trapezoid graphs, House-Hole-Domino–free graphs, distance-hereditary graphs and many others. Hence, the results of this subsection generalize some known results from [34, 52]. We recall also that, as it was shown in [34], no system of constant number of collective additive tree r-spanners can exist for unweighted weakly chordal graphs for any constant $r \geq 0$.

The above results can easily be extended to all 4-chordal graphs (note that in the proof of Lemma 30 the absence of $\overline{C}_6$ in G was important only for Claim 3). We can show that every 4-chordal graph is $(1/2, 6, \mathsf{w})$-decomposable.

Lemma 31 *If G is a 4-chordal graph, then there exists a set D of at most six vertices in G such that $S \subseteq N[D]$.*

Proof Let A_u, A_d be the same vertex sets as defined in the proof of Lemma 30. Let x be a vertex of A_u with minimum $|N_\downarrow(x)|$ among all vertices of A_u. Similarly, let y be a vertex of A_d with minimum $|N_\downarrow(y)|$ among all vertices of A_d. We claim that for any vertex $z \in A_u$, if $xz \notin E(G)$, then $zf(x) \in E(G)$ must hold.

Assume $zf(x) \notin E(G)$ for some $z \in A_u$. By the choice of x, there must exist a vertex $z' \in N_\downarrow(z)$ such that $xz' \notin E(G)$. Since x, z are in A_u, there must exist an induced path $P_G(x, z)$ all inner vertices of which are from $C^*(i+1)$. This path has at least 2 edges. On the other hand, there is a path $P_G(f(x), z')$ in G such that, if $f(x)z' \notin E(G)$, then all its inner vertices are from levels $L_s(v_n), s < l(v_i) - 1$. Path $P_G(f(x), z')$ has at least 1 edge. Furthermore, by Proposition 3, no vertex on $P_G(f(x), z')$ can be adjacent to inner vertices of $P_G(x, z)$. Therefore, $P_G(f(x), z'), P_G(x, z)$ and two edges $xf(x), zz'$ form an induced cycle with at least 5 edges, which is impossible. This contradiction proves our claim.

Analogously, one can show that for any vertex $z \in A_d$, if $yz \notin E(G)$, then $zf(y) \in E(G)$ must hold. Now, since $S \subseteq N[v_i] \cup N[f(v_i)] \cup A_u \cup A_d$ and $A_u \subseteq N[x] \cup N[f(x)]$, $A_d \subseteq N[y] \cup N[f(y)]$, we conclude that $S \subseteq N[D]$, where $D := \{v_i, f(v_i), x, f(x), y, f(y)\}$. □

Thus, the following results true.

Theorem 27 *Every 4-chordal graph is* $(1/2, 6, \mathsf{w})$*-decomposable.*

Corollary 12 *Any n-vertex m-edge 4-chordal graph G admits a system of at most $6 \log_2 n$ collective additive tree $(2\mathsf{w})$-spanners. Moreover, such a system of spanning trees can be constructed in $O(nm \log n)$ time.*

Corollary 13 *Any n-vertex m-edge 4-chordal graph G admits an additive $(2\mathsf{w})$-spanner with at most $O(n \log n)$ edges. Moreover, such a sparse spanner can be constructed in $O(nm \log n)$ time.*

Chordality of the graph	Decomposition results
3	$(1/2,\ 1,\ \mathsf{w})$
4	$(1/2,\ 1,\ 2\mathsf{w}),\quad (1/2,\ 6,\ \mathsf{w})$
5	$(1/2,\ 1,\ 2\mathsf{w})$
6	$(1/2,\ 1,\ 3\mathsf{w}),\quad (1/2,\ 4,\ 2\mathsf{w})$
7	$(1/2,\ 1,\ 3\mathsf{w})$
8	$(1/2,\ 1,\ 4\mathsf{w}),\quad (1/2,\ 4,\ 3\mathsf{w})$
9	$(1/2,\ 1,\ 4\mathsf{w}),\quad (1/2,\ 5,\ 3\mathsf{w})$
$\xi \geq 10$	$(1/2,\ 1,\ \lfloor \xi/2 \rfloor \mathsf{w}),\quad (1/2,\ 4,\ (\lfloor \xi/3 \rfloor + 1)\mathsf{w})$
$\xi = 3k,\ k \geq 4$	$(1/2,\ 1,\ \lfloor 3k/2 \rfloor \mathsf{w}),\quad (1/2,\ 4,\ (k+1)\mathsf{w}),\quad (1/2,\ 5,\ k\mathsf{w})$

Table 3.3: Summary of the decomposition results obtained for ξ-chordal graphs.

The last result improves and generalizes the results from previous section on sparse spanners of unweighted chordal graphs.

In Table 3.3, we summaries all our decomposition results obtained for ξ-chordal graphs.

3.4 Collective tree spanners for UDG

A graph $G = (V, E)$ is unit disk graph (UDG), if the vertices of G can be embed on the plan such that two vertices $u, v \in V$ are adjacent in G if and only if the Euclidean distance between u and v are at most 1. In this section, we describe how to construct a system of $O(\log n)$ collective tree (3, 4)-spanners for UDG. In the sequel, all graphs are unweighted. Let G be a graph with vertices of G embedded on the plane. For any two vertices a, b of G, if a is adjacent to b in G, we use (a, b) to denote the edge of G. ab is used to denote the line segment with a and b as end points. Each edge (a, b) of G is treated the same way as a line segment, unless noted otherwise. If two edges $(a, b), (c, d)$ of G intersect, it means the two line segment ab and cd intersect. Hence, in the following, when we say: two edges intersect, an edge intersect with a line segment or two line segments intersect, they have the same meaning. Moreover, when we say (a, b) intersects with (c, d), we exclude the cases when c or d is on edge (a, b), or when a

or b is on edge (c, d). For any two vertices a, b of G, $|ab|$ is used to denote the Euclidean distance between a and b on the plane. The *hop distance* between a and b in G means the number of edges on the shortest path between a and b in G. For any vertex a of G, point a means the point at the location corresponding to vertex a when G is embedded on the plane.

Let G be a unit disk graph and r be an arbitrary vertex. Construct a LexBFS tree T_r rooted at r. For any vertex $x \in V(G)$, σ_x is used to denote the LexBFS ordering number of x. First, we show how to get another tree T'_r such that no two edges of T'_r intersect. Let (a, b), (c, d) be two edges of T_r such that they intersect. Let l_a, l_b, l_c, l_d be the hop distances from a, b, c, d to r, respectively. Without loss of generality, assume $l_a < l_b$, $l_c < l_d$, $l_a \leq l_c$ and $\sigma_a > \sigma_c$. The following lemma holds.

Lemma 32 *At least one of the edges in* $\{(a, c), (b, d)\}$ *and at least one of the edges in* $\{(a, d), (c, b)\}$ *must be in* $E(G)$.

Proof Let f be the intersection point between (a, b) and (c, d). Since (a, b) and (c, d) are edges of G, one concludes $|ab| \leq 1$ and $|cd| \leq 1$. According to triangle inequality, the following inequalities

$$|af| + |cf| \geq |ac|$$

$$|df| + |bf| \geq |bd|$$

must hold. Combine the two inequalities, one get

$$2 \geq |ab| + |cd| = |af| + |fb| + |cf| + |fd| \geq |ac| + |bd|$$

This inequality implies that one of the inequalities in $\{|ac| \leq 1, |bd| \leq 1\}$ must be true. This is equivalent to say at least one of the edges in $\{(a, c), (b, d)\}$ is in $E(G)$.

Similarly, one can show that at least one of the edges in $\{(a, d), (c, b)\}$ is in $E(G)$. This concludes our proof.

□

The following property is needed for further discussions.

Property 1 *Assume $\triangle abc$ is a triangle such that all sides of $\triangle abc$ have length at most 1. Then the distance between any point in $\triangle abc$ (including the edges) and $d \in \{a, b, c\}$ is at most 1.*

Note, in the shortest path tree T_r of G, there may be two edges $(a, b), (c, d) \in E(T_r)$ such that they cross each other on the plane. In next subsection, we show how to change T_r to another tree T'_r such that no two edges in T'_r cross each other.

3.4.1 Eliminate intersections between two edges of T_r

The following lemma can be proved.

Lemma 33 $l_a \leq l_c \leq l_a + 1$ *must hold.*

Proof Proof is by contradiction. Assume $l_c \geq l_a + 2$. By lemma 32, one knows that at least one of the edges in $\{(a, c), (b, d)\}$ must be in $E(G)$. Assume $(a, c) \in E(G)$, then since $l_c \geq l_a + 2$, there is a shorter path from c to r via a. This contradicts with T_r is a LexBFS tree. Now, assume $(a, c) \notin E(G)$ and $(b, d) \in E(G)$. Since $l_b = l_a + 1$ and $l_d = l_c + 1$, $l_d \geq l_b + 2$ must hold. Then, there is a shorter path from d to r via b, contradicts with T_r is a LexBFS tree. This proved the lemma.

□

Lemma 32 says that at least one of the edges in $\{(a, d), (c, b)\}$ is in $E(G)$. Since $\sigma_a > \sigma_c$, $(a, d) \notin E(G)$ (otherwise, (a, d) instead of (c, d), will be in $E(T_r)$). Therefore, $(c, b) \in E(G)$ must be true.

Lemma 33 implies that only two cases need to be considered. One case is $l_a = l_c + 1$ and another case is $l_c = l_a$. First, we show how to eliminate the case when $l_c = l_a + 1$. Since T_r is a LexBFS tree, (a, d) can not be in $E(G)$. Add a vertex c' at the point where

(a, b) and (c, d) intersect. c' is called *mirror vertex of* c. Eliminate the edge (c, d) and add (c', d) into $E(T_r)$ and let $\sigma_{c'} = \sigma_c$ and $l_{c'} = l_c$. This process is illustrated in Figure 3.5. If there are several such edges intersect with (c, d), then choose the one such that the intersect point is closest to d and perform the above operation. Since T_r is a tree, the tree will be separated into two connected components T_c and T_d after deleting edge (c, d). Clearly, vertex d must be in T_d and a must be in T_c. The edge (a, b) is in T_c. Therefore, after the above process, T_r is still a tree.

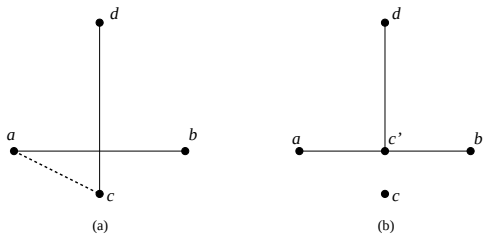


Figure 3.5: (a) Before the change (b) After the change.

Since all the intersections with $l_c = l_a + 1$ are eliminated in T_r, the only case left is when $l_a = l_c$. Let c' be a mirror vertex of c. For any two edges (c', d) and (c', f) in $E(T_r)$, it is impossible for them to intersect at point other than c'. For any vertex c of G, only its mirror vertex c' has the same LexBFS-ordering number as c. Hence, if two edges $(a, b), (c, d) \in E(T_r)$ intersect, then σ_a can not be the same as σ_c. Without loss of generality, we assume σ_a is greater than σ_c. Lemma 32 tells us that one of the edges in $\{(a, d), (c, b)\}$ is in $E(G)$. By the property of LexBFS-ordering, one knows that $(a, d) \notin E(G)$, hence (c, b) must be in $E(G)$.

Before describing how to eliminate all the intersections in T_r with $l_c = l_a$, we give some notations and definitions. The vertices a, b, c, d of G and mirror vertices are called *active vertices*. All other vertices added by the algorithm are called *dummy vertices*. All the dummy vertices are on edges of T_r. For any edge (a, b), the algorithm may add some

dummy vertices on it. An example was illustrated in Figure 3.6 before and after dummy vertices were added on edge (a, b). For any vertex a, let $P_{a,r}$ be the path between a and root r in tree T'. Then the active vertex f which is closest to a on $V(P_{a,r}) \setminus \{a\}$ and satisfy $l_f = l_a - 1$ is called *active father of* a *in* T'. $af(a)$ is used to denote the active father of a. All the active vertices $x \in V(P_{a,r}) \setminus \{a\}$ and with $l_x < l_a$ are called *active ancestor of* a. Let x be a vertex on the edge (a, b) of T_r. If $af(x) = a$ and x is farthest from a among all such vertices, then x is called the *lower end* of edge (a, b) and a' is used to denote it. Clearly, at the beginning, $a' = b$. In the following, whenever we say an edge (a, b) of tree T', we mean a is the father of b in T'.

Figure 3.6: (a) The original edge (a, b) in T_r. (b) After adding dummy vertices. a' is *dummy end*.

The following algorithm is used to eliminate the intersection with $l_c = l_a$.

PROCEDURE 1. Eliminate the intersection in T_r

Input: A LexBFS tree T_r with some mirror vertices

Output: A tree T'_r such that no two edges of T'_r cross each other.

Method:

$E' = E(T_r)$;

$T' = T_r$;

Let $\sigma = [v_1, \cdots, v_{n'}]$ be the LexBFS ordering of the vertices of G and mirror vertices;

/* Break ties arbitrarily */

while there are intersections **do**

for i from 1 to n' **do**

for each lower end v_i' of v_i **do**

if there is an edge (c, d) in $E(T')$ which intersects with the line segment $v_i v_i'$ and $\sigma_{af(c)} < \sigma_{af(v_i)}$ and the intersection point is closest to v_i' among all such kind of intersections **then**

let (a, b) be the edge on line segment $v_i v_i'$ such that (a, b) intersects with (c, d);

add a dummy vertex c' at the intersection point between (a, b) and (c, d);

/* By the choice of (c, d), one knows that no edge (a', b') intersects with line segment $c' v_i'$ such that $\sigma_{af(a')} < \sigma_{af(v_i)}$ */

$E' = E' \setminus \{(c, d), (a, b)\} \cup \{(c', d), (c', b), (c, c')\}$;

/* a becomes the lower end of v_i and c' is a dummy vertex */

set $l_{c'} = l_d$;

$T' = (V, E')$;

while there is a mirror vertex $x \in V(T')$ which is a leaf **do**

eliminate x and the edge incident on it from T';

$T_r' = T'$;

return T_r'

An example illustrating how Procedure 1 is executed is shown in Figure 3.7.

In the following, we will show the graph T' constructed during Procedure 1 is a tree. Formally, we prove the following lemma.

Lemma 34 *Assume T' is the graph constructed during Procedure 1. Let (a, b) and (c, d) be two edges of T' such that they intersect with each other. Without loss of generality,*

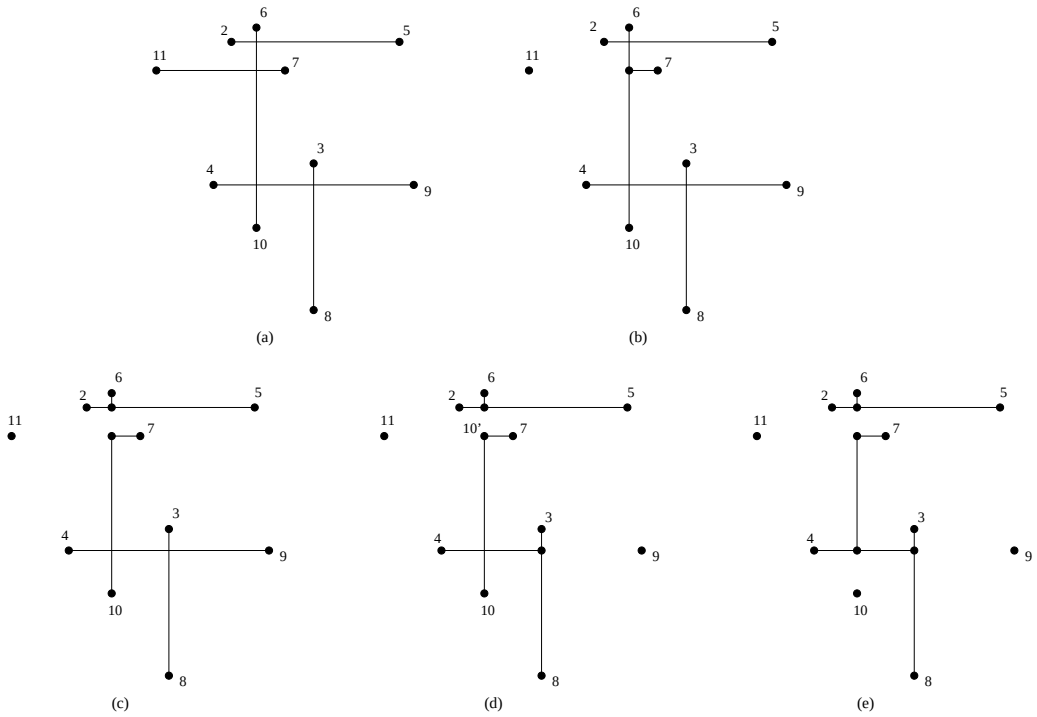


Figure 3.7: (a) The original intersection. Numbers are LexBFS ordering number (b) When vertex 11 is v_i and edge (11, 7) is considered. (c) When vertex 10 is v_i and edge (10, 6) is considered. (d) When vertex 9 is v_i and edge (9, 4) is considered. (e) When vertex 10 is v_i and edge (10, 10') is considered.

assume $l_a \geq l_b$ and $l_c \geq l_d$. Then the following statements hold.

(1) *T' is a tree.*

(2) *for each vertex x of T', l_x is the number of active ancestors of x in T'. .*

(3) $af(b) \neq af(d)$.

(4) $l_b = l_d$.

Proof Proof is by induction. At the beginning, T' is T_r, the lemma clearly holds.

Consider two edges $(a, b), (c, d)$ of T' intersect with each other. Let c' be the intersection point between (a, b) and (c, d). Procedure 1 will do the following operation: eliminate (a, b) and (c, d), and add $(c', b), (c', d), (c, c')$ into $E(T')$. Let T'' be the new

graph after the operation. Assume that before the operation, the lemma holds. Now, let us prove that after the operation, the lemma is still true.

By induction, we know that $l_b = l_d$. Since T' is a tree, $T' \setminus \{(a, b)\}$ will separate the tree into two connected components T_b and T_a. By induction, one knows that l_d is the number of active ancestors of d in T'. The level number of active ancestors will strictly decrease. By induction, one knows that $\sigma_{af(d)} \neq \sigma_{af(b)}$. Combine this with $l_b = l_d$, one deduces that edge (a, b) is not on the path from d to r in T'. Since after deleting edge (a, b), a is still connected to the root r, hence, (c, d) is in T_a. Therefore, creating a vertex on edge (c, d) and make it connected to b will not create any cycle. This proved that T'' is still a tree.

According to Procedure 1, we will set $l_{c'}$ to l_d after the operation. By induction, l_d is the number of active ancestors of d. Since c' is a dummy vertex, by the definition of *active ancestors*, one concludes that $l_{c'} = l_d$. This proved that $l_{c'}$ is the number of active ancestors of c'. Since $l_b = l_d$ before the operation and c' is a dummy vertices, hence, after the operation, l_b is still the number of active ancestors in T''.

All edges added by the operation are either part of (a, b) or part of (c, d). Hence, the above operation will not create any new intersections in T''. Since c' is a dummy vertex, hence, $af(c') = af(d)$ and $l_{c'} = l_d$ must be true. By induction, any edge (a', b') that intersects with (c, c') must have $af(b') \neq af(d)$ and $l_{b'} = l_d$. This implies $af(b') \neq af(c')$ and $l_{b'} = l_{c'}$. Therefore, for edge $(c', d), (c, c')$, the lemma still holds.

Now, consider edge (c', b). Let (a', b') be an edge which intersects with (c', b) in T'. After the operation, $af(b) = af(c') = af(d)$ holds. We claim that $af(b') \neq af(b)$ in T''. Assume $af(b') = af(b)$. This implies $\sigma_{af(b')} = \sigma_{af(d)} \leq \sigma_{af(b)}$. According to Procedure 1, the intersection between (a, b) and (a', b') will first be detected and eliminated. This contradiction proves the claim. By induction, $l_{b'}$ must be equal to l_b. Hence, the lemma still holds for edge (c', b) in T''. Combine this with above, one concludes that the lemma

is true for tree T''. This completes the proof of the lemma.

□

Let T' be a tree generated during the *while* loop in Procedure 1. By the definition of active father and the algorithm, the following observation clearly holds.

Observation 3 *$\sigma_{af(x)}$ will strictly decrease during Procedure 1.*

Assume there are two intersecting edges $(a', b'), (c', d')$ of T'. Let $(a, b), (c, d)$ be two edges in $E(T_r)$ such that a', b' lie on (a, b), and c', d' lie on (c, d). The following lemma holds.

Lemma 35 *One of the following cases must be true*

(1) $af(b') = a$.

(2) $af(d') = c$.

Moreover, if $\sigma_{af(b')} > \sigma_{af(d')}$, then $af(b')$ must be a.

Proof The proof is by contradiction. Assume none of the above cases is true. Without loss of generality, assume $\sigma_a \geq \sigma_c$. By Observation 3, $\sigma_a \geq \sigma_{af(d')}$ holds throughout Procedure 1. By assumption, $af(b')$ is not a. According to Procedure 1, this can happen only when there is an edge (c'', d'') which intersects with line segment ab' and $\sigma_{af(d'')} < \sigma_a$ during previous *while* loop. Since (a', b') is an edge of T', this implies that the intersection point between (c'', d'') and line segment (a, b') must lies on line segment aa'. Let s be the intersection point between (c'', d'') and aa'. Consider when Procedure 1 first detects such an intersection. As mentioned in the comments of Procedure 1, there should not be any other edges (c', d') such that (c', d') intersect with sb' and $\sigma_{af(d')} < \sigma_a$. This contradicts with that at the beginning of the *while* loop of Procedure 1, one can still find an edge (c', d') that intersects with (a', d'). This proved that at least one of the cases must be true.

Consider the second part of the lemma. Assume $af(b') \neq a$. By Observation 3, $\sigma_a > \sigma_{af(b')} > \sigma_{af(d')}$ must hold. By similar arguments to the above, one deduces that this can happen only if in previous iterations of the *while* loop, there is an edge (c'', d'') which intersects with line segment aa' and $\sigma_{af(d'')} < \sigma_a$. Let s be that intersection point. According to Procedure 1, there should be no edge (c', b') such that it intersects with sb' and $\sigma_{af(b')} < \sigma_a$. This contradiction proves the second part of the lemma.

□

Lemma 36 *The inequality $\sigma_{af(b')} \neq \sigma_{af(d')}$ must hold.*

Proof Proof is by contradiction. Assume $\sigma_{af(b')} = \sigma_{af(d')}$. By Lemma 35, we know that either $af(b') = a$ or $af(d') = c$. Without loss of generality, assume $af(b') = a$. If $af(d') = c$, then the lemma clearly holds. So assume $af(d') \neq c$. Since $\sigma_{af(d')}$ will strictly decrease, one concludes $\sigma_c > \sigma_{af(d')} = \sigma_a$. From the comments of Procedure 1, one concludes that there should be no edge (a'', b'') intersect line segment $c'd'$ and $\sigma_{af(b'')} < \sigma_c$. But (a', b') is such an edge. This contradiction proves the lemma.

□

Assume c' is a mirror vertex of c and the intersection point between two edges (a, b) and (c, d) of G. b is defined as the *extended active vertex* of c' and $\mathrm{ea}(c')$ is used to denote it. For convenience, for an active vertex c, if it is not a mirror vertex, then $\mathrm{ea}(c)$ is defined to be c. Vertex c is called the *real ancestor* of c' and $\mathrm{ra}(c')$ is used to denote it. As before, if c is a vertex of G, then $\mathrm{ra}(c)$ is defined to be c itself. Assume T'_r is a tree constructed by Procedure 1 and x is an active vertex of T'. $P_{x,af(x)}$ is used to denote the path between x and its active father $af(x)$ in T'_r. The following lemma holds.

Lemma 37 *Let y be an arbitrary vertex on the path $P_{x,af(x)}$, then the inequality*

$$|xy| \leq 1$$

$$|af(x)y| \leq 1$$

must hold.

Proof Let $(c_0, d_0), (c_1, d_1), \cdots, (c_k, d_k)$ be the edges of T_r such that, for any (c_i, d_i) (where $0 \leq i \leq k$ is an integer), there is at least an edge from $P_{x,af(x)}$ which lies on it. Let $s_i \in V(P_{x,af(x)})$ be the intersection point between (c_i, d_i) and (c_{i+1}, d_{i+1}) where $i = 0, 1, \cdots, k-1$. Clearly, $af(x) = s_k = c_k$. An example is shown in Figure 3.8.

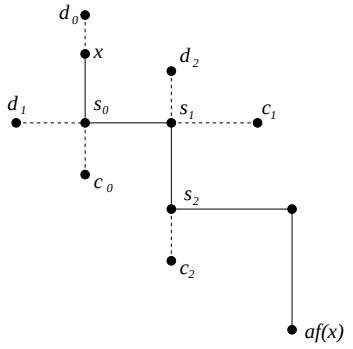


Figure 3.8: $af(x) = s_k = c_k$

By Procedure 1, $\sigma_{c_0} > \sigma_{c_1}$ must be true. Therefore, (c_0, d_1) can not be in $E(G)$. By Lemma 32, one deduces that (c_1, d_0) is in $E(G)$ which implies $|c_1 d_0| \leq 1$. Since x is on line segment $s_0 d_0$, $|c_1 x| \leq 1$ must hold. Consider the triangle $\triangle c_1 s_0 x$. Each side of this triangle has length at most 1, hence by Property 1, and for any vertex y which is on line segment $c_1 s_0$, the inequality $|xy| \leq 1$ must hold. If $k = 1$, then we are done. Otherwise, there must be an edge $(c_2, d_2) \in E(T_r)$ such that it intersects with (c_1, d_1) at s_1 and $\sigma_{c_2} < \sigma_{c_1}$. Note, according to Procedure 1, it's easy to deduce that s_1 must lie on line segment $s_0 c_1$, otherwise s_1 can not be on $P_{x,af(x)}$.

According to the way of constructing T'_r, $c_2 d_2$ can not intersect with line segment $s_0 x$. Otherwise, s_0 can not be on $P_{x,af(x)}$. Since $c_2 d_2$ intersects with $c_1 s_0$ and $c_2 d_2$ does not intersect with $s_0 x$, there may be two cases. First case, $c_2 d_2$ intersects with $c_1 x$. Similar to above, one knows that $(c_1, d_2) \notin E(G)$, by Lemma 32, one deduces that

$(c_2, x) \in E(G)$. Therefore, $|c_2x| \leq 1$. Second case is c_2d_2 does not intersect with c_1x. We also know that c_2d_2 does not intersect with s_0x and does intersect with c_1s_0. This implies that exactly one of the vertices from $\{c_2, d_2\}$ must be in the triangle $\triangle c_1s_0x$. This vertex must be c_2. Since if d_2 is in this triangle, by Property 1, (c_1, d_2) is in $E(G)$ which is forbidden. If c_2 is inside the triangle, then clearly $|c_2x| \leq 1$ must hold.

Hence, in any case, $|c_2x| \leq 1$. Consider triangle $\triangle c_2s_1x$, each side of the triangle has length at most 1. By Property 1, for any vertex y on line segment c_2s_1, $|xy| \leq 1$ must hold. If $k = 2$, then we are done. Otherwise, we can continue this process until reach c_k. This proved the lemma.

The second inequality can be proved in a similar way, hence it is omitted here.

□

From the proof of Lemma 37, it follows that $|ea(x)y| \leq 1$. Hence, the following corollary are given without proof.

Corollary 14 *Let y be an arbitrary vertex on the path $P_{x,af(x)}$, then the inequalities hold*

$$|ea(x)y| \leq 1$$

$$|ra(af(x))y| \leq 1$$

Before we show how to convert a path $P_{s,r}$ in T'_r into a shortest path $P_G(s, r)$ in G, we prove the following lemma.

Let x be an arbitrary point on the plane and $\mathcal{P} = (v_0, v_1, \cdots, v_k)$ be a simple polygon. The following lemma holds.

Lemma 38 *If $|xv_i| \leq 1$ and $|v_iv_{i \bmod (k+1)}| \leq 1$, for $i = 0, \cdots, k$, then for any point y in $\mathcal{P}$ (including the edges of $\mathcal{P}$), $|xy| \leq 1$ must hold.*

Proof Extend the ray from x which passes y. Since y is in $\mathcal{P}$, this ray will hit an edge of $\mathcal{P}$. Let v_iv_{i+1} be the edge. Consider the triangle $\triangle xv_iv_{i+1}$. Clearly, each edge of

the triangle has length at most 1. By Property 1, one concludes $|xy| \leq 1$ holds. This proved the lemma.

□

For an active vertex s, we will use the following algorithm to convert a path $P_{s,r}$ in T'_r to a shortest path $P_G(s,r)$ in G.

PROCEDURE 2. Replace mirror vertex with vertex in G

Input: A path $P_{s,r}$.

Output: A shortest path $P_G(s,r)$ in G.

Method:

$x = y = s$;

$P_G(s,r) = \emptyset$;

$y = ea(s)$, add y into $P_G(s,r)$;

while $x \neq r$ **do**

  if $af(x)$ is a vertex of G **then**

    add $af(x)$ into $P_G(s,r)$ and let $af(x)$ be the father of y;

    set $y = af(x)$;

  else $af(x)$ is a mirror vertex of c **then**

    if $(ea(af(x)), y) \in E(G)$ **then**

      add $ea(af(x))$ into $P_G(s,r)$ and let $ea(af(x))$ be the father of y;

      set $y = ea(af(x))$;

    else

      add c into $P_G(s,r)$;

      let c be the father of y;

      set $y = c$;

set $x = af(x)$;

return $P_G(s, r)$;

Procedure 2 will only replace the mirror vertex on $P_{s,r}$. For any mirror vertex c' on $P_{s,r}$, $r_s(c')$ denotes the vertex in $P_G(s, r)$ used to replace c'. For a vertex $a \in P_G(s, r)$, $m_s(a)$ denotes the vertex in $P_{s,r}$ replaced by a. We will show that $P_G(s, r)$ is a shortest path in G. In order to show this, we need to show that for any vertex $y \in P_G(s, r)$, the following statements are true. First, it is adjacent to its father. Second, its level number l_y is equal to its father's level number plus 1. According to Lemma 34, for each active vertex x of T'_r, $l_{af(x)} = l_x + 1$. According to Procedure 2, each active vertex x will be replaced either by $ea(x)$ or $ra(x)$. By the definition of $ea(x)$ and $ra(x)$, $l_x = l_{ea(x)}$ and $l_x = l_{ra(x)}$ hold. Therefore, the second statement is true, and we only need to prove the first statement, that is, y and its father are adjacent in G.

Lemma 39 *Let p be a vertex in $P_G(s, r)$ and q be its father in $P_G(s, r)$. For any vertex y on $P_{m_s(p),m_s(q)}$, the following inequalities must hold*

$$|py| \leq 1$$

$$|pq| \leq 1$$

Proof We prove the lemma by induction. We start from s. In Procedure 2, $ea(s)$ is used to replace s. According to Lemma 37 and Procedure 2, the statement clearly holds. Assume the statement holds until vertex p on $P_G(s, r)$. If $p = r$, then we are done. Hence, assume $p \neq r$. Let x be a vertex on $P_{s,r}$ such that $af(x) = m_s(p)$. If $r_s(p)$ is $ea(m_s(p))$, then by Lemma 37 and Procedure 2, we are done. Hence assume $r_s(p) \neq ea(m_s(p))$. Then, according to Procedure 2, $m_s(p)$ must be a mirror vertex of p and $(r_s(x), ea(m_s(p))) \notin E(G)$. By the definition of dummy and mirror vertex and

Lemma 34, one concludes $l_{r_s(x)} > l_{r_s(q)} + 1$. This implies that $(r_s(x), r_s(q))$ is not an edge of G.

Assume $m_s(p)$ is the intersection point between (a_1, b_1) and (c_1, d_1). Assume $l_{b_1} = l_{c_1} = l_{a_1} + 1 = l_{d_1} - 1$ (clearly, by above assumption, $p = c_1$ and $b_1 = ea(m_s(p))$). Since $P_G(s, r)$ is a shortest path, $(r_s(x), a_1) \notin E(G)$. By our assumption, $(r_s(x), b_1) \notin E(G)$. We claim $(r_s(x), p)$ and (a_1, b_1) intersect. Assume they do not intersect. Since (p, d_1) intersect with (a_1, b_1), a_1 or b_1 must be in the polygon made by part of edge $(r_s(x), p)$, part of path $P_{x,m_s(p)}$ and line segment $m_s(p)p$ (see Figure 3.9 for an illustration). By

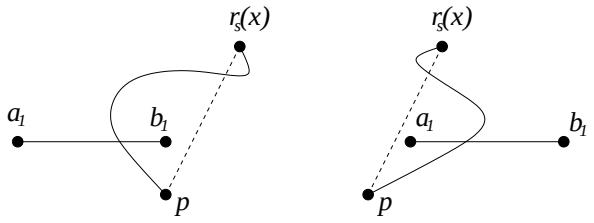


Figure 3.9: (a)b_1 is in the polygon formed by part of edge $(r_s(x), p)$ and part of path $P_{x,m_s(p)}$ (b) b_1 is in the polygon formed by part of edge $(r_s(x), p)$ and part of path $P_{x,m_s(p)}$

Lemma 38, $(r_s(x), b_1)$ or $(r_s(x), a_1)$ must be in $E(G)$. These contradict either with our assumption or with that $P_G(s, r)$ is a shortest path. This proved the claim. According to Lemma 32 and the fact that $(r_s(x), b_1) \notin E(G)$, (p, a_1) must be in $E(G)$. Since $|m_s(p)a_1| \leq 1, |m_s(p)p| \leq 1$, for any point y on line $m_s(p)a_1$, $|py| \leq 1$ must hold. If a_1 is q then we are done. Otherwise, there must be an edge (a_2, b_2) of T_r intersecting (a_1, b_1) with $\sigma_{a_2} < \sigma_{a_1}$. By Lemma 37, $(b_1, a_2) \in E(G)$. We know $(r_s(x), b_1)$ and $(r_s(x), a_2)$ are not in $E(G)$. Similar to above claim, one can show (b_1, a_2) must intersect with $(r_s(x), p)$. By Lemma 32, $(r_s(x), b_1) \notin E(G)$ implies (p, a_2) is in $E(G)$. Continue this process until (a_i, b_i) such that $m_s(q)$ is on it. By Procedure 2, $q = b_i$ implies $(p, q) \in E(G)$ and we are done. If $q \neq b_i$, then $m_s(q)$ must be a mirror vertex of a_i and $q = a_i$. By Corollary 14, one concludes that $(ea(m_s(p)), a_i)$ is in $E(G)$. By induction, $(r_s(x), p)$ is an edge

of G. We claim that $(r_s(x), p)$ must intersect with $(ea(m_s(p), a_i)$. Assume not. Then either $ea(m_s(p))$ or a_i is in the polygon made by part of $(r_s(x), p)$, part of $P_{x,m_s(p)}$, line segment $xr_s(x)$ and line segment $m_s(p)p$. Combine this above with Lemma 38, one of the edges in $\{(r_s(x), ea(m_s(p))), (r_s(x), a_i)\}$ must be in $E(G)$. This contradicts with our assumption and $P_G(s, r)$ is a shortest path. Hence, the claim must hold. Then since $(r_s(x), ea(m_s(p)))$ is not in $E(G)$, by Lemma 32, (p, a_i) must be in $E(G)$. Combine this with above proof, the lemma clearly holds. This completes our proof.

□

Let x be an arbitrary active vertex of T'_r and $af(x)$ be its active father. $P_{x,af(x)}$ is the path between x and $af(x)$. Let $(a, b) \in E(G)$ be an edge of G. The following lemma holds.

Lemma 40 *If (a, b) intersects with path $P_{x,af(x)}$, then at least one of the edges in $\{(a, x), (af(x), b)\}$ is in $E(G)$.*

Proof By Lemma 37, one knows that $(af(x), x) \in E(G)$. If $(af(x), x)$ intersects with (a, b), then by Lemma 32, one knows that one of the edges in $\{(a, x), (af(x), b)\}$ is in $E(G)$, and we are done.

If $(af(x), x)$ does not intersect with (a, b). Then one of the vertex in $\{a, b\}$ must be in the simple polygon formed by (part of the) edge $(x, af(x))$ and (part of the) path $P_{x,af(x)}$. Without loss of generality, assume a is in such polygon. See Figure 3.10 (b) for an illustration. By Lemma 37 and 38, one can easily deduce that (x, a) is in $E(G)$. This concludes the proof of this lemma.

□

Assume x is an arbitrary vertex on path $P_{s,r}$. As above, $r_s(x)$ and $r_s(af(x))$. Similar to Lemma 40, one can show the following lemma holds. The proof is omitted.

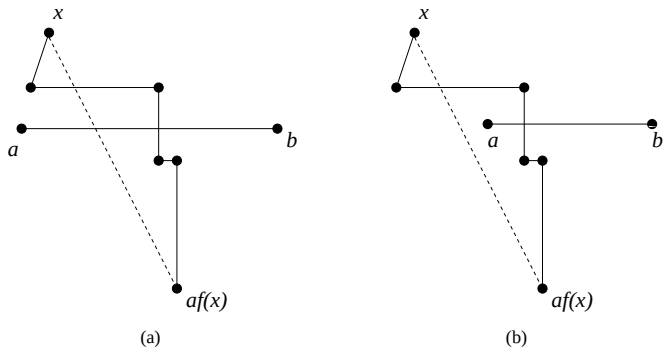


Figure 3.10: (a)The case that ab intersects with $af(x)x$. (b)The case that ab do not intersect with $af(x)x$.

Lemma 41 *If (a, b) intersects with path $P_{x,af(x)}$, then at least one of the edges in $\{(a, r_s(x)) , (r_s(af(x)), b)\}$ is in $E(G)$.*

For any path P of G and positive integer k, $N_k[P] = \{u : \exists v \in V(P) \text{ and } d_G(u, v) \leq k\}$. With this lemma, the following theorem clearly holds.

Theorem 28 *If an edge (a, b) of G intersects with the path $P_{s,r}$, then a or b must be in $N_1[P_G(s, r)]$.*

3.4.2 Finding two balanced shortest-path separators

In this subsection, we will show how to find two shortest-pathes P_1 and P_2 of a n-vertex unit disk graph G, such that any connected component in $G \setminus \{N_1[P_1] \cup N_1[P_2]\}$ contains at most $2n/3$ vertices. The following definitions are needed for our proof.

Assume T'_r is the tree returned by Procedure 1. For any two vertices $x, y \in V(T'_r)$, $P_{x,y}$ is used to denote the path between x and y in T'_r. Let a, b be two arbitrary vertices of T'_r. nca(a, b) is used to denote the *nearest common ancestor of a and b in T'_r*. Let c be nca(a, b). The edge (a, b) together with path $P_{c,a}, P_{c,b}$ divides the plane into two parts, the inner face and the out face. Let $F_{a,b}$ be the inner face. Assume x is an arbitrary vertex of T'_r. Let $a' =$nca(a, x) and $b' =$nca(b, x). Assume (a', a'') and (b', b'') are two edges on $P_{a',x}$ and $P_{b',x}$, respectively. We say x is *inside the face $F_{a,b}$*, if one of the

following is true.

(1) a' is a vertex in $V(P_{a,c}) \setminus \{a\}$ and (a', a'') is in $F_{a,b}$.

(2) b' is a vertex in $V(P_{b,c}) \setminus \{b\}$ and (b', b'') is in $F_{a,b}$.

The vertices on $P_{a,c}$ and $P_{b,c}$ are said to be the boundary of $F_{a,b}$. Any vertex which is neither inside $F_{a,b}$ nor on the boundary of $F_{a,b}$ is said to be *outside the face* $F_{a,b}$.

An edge (a, b) is *valid* if the following is true.

(1) There is no edge $(c, d) \in E(G)$ with c inside face $F_{a,b}$ and d outside $F_{a,b}$, and $\{c, d\} \cap (N_1(P_G(a, r)) \cup N_1(P_G(b, r))) = \emptyset$.

If the above does not hold, then we say edge (a, b) is *invalid*.

The following lemmata can be proved.

Lemma 42 *For any valid edge (a, b), if there are at least one vertex inside (or outside) $F_{a,b}$, then there exists a vertex c inside (or outside) $F_{a,b}$ such that both (a, c) and (c, b) are valid.*

Proof Let c be $\mathrm{nca}(a, b)$. Arbitrarily pick a vertex d which is inside $F_{a,b}$. If both (a, d) and (d, b) are valid, then we are done. Without loss of generality, assume (a, d) is invalid. This implies that there is an edge $(s, t) \in E(G)$ such that one of the vertex (say s) is inside $F_{a,d}$ and t is outside $F_{a,d}$.

We claim that both s and t are inside $F_{a,b}$. Clearly, since s is inside $F_{a,d}$, it must be inside $F_{a,b}$. If t is outside $F_{a,b}$, then (a, b) will be invalid. This contradicts with our assumption. Besides, at most one vertex (say s) in $\{s, t\}$ can be on the boundary of $F_{a,b}$. This is clear, since, if both s and t are on the boundary, then (s, t) do not make (a, d) invalid. Now, set $d = t$. If (a, d) is still invalid, continue this process. One can easily see that each edge $(s, t) \in E(G)$ will be processed at most once. Since there are finite number of edges in $E(G)$, this process will terminate. Finally, (a, d) will be valid.

If (b, d) is valid, then we are done. Otherwise, there must be an edge $(s', t') \in E(G)$ with t' outside $F_{b,d}$ and s' inside $F_{b,d}$. Note, t' can not be inside $F_{a,d}$, since otherwise (a, d) will be invalid. This are only two possibilities. First, d is an active ancestor of t'. Second, t' is on $P_{a,c}$. In both cases, set $d = s'$. Continue this process until (d, b) is valid. At this point, if (a, d) is valid, then we are done. Otherwise, repeat the process for (a, d). It is clear, that after an edge $(s, t) \in E(G)$ is processed, it will never be processed again. Hence, this process will terminate. Finally, we can find a vertex d inside $F_{a,b}$ such that both (a, d) and (d, b) are valid.

Note, for the outside case, the proof is almost the same. The only difference is, d could be a vertex such that a or b is d's ancestor. This concludes our proof.

□

With above lemma, we are ready to prove the following lemma.

Lemma 43 *There exit two active vertices a and b with the following properties. (a, b) is a valid edge. If $N_1(P_G(a, r)) \cup N_1(P_G(b, r))$ are removed from $V(G)$, then no connected connected component contains $2n/3$ vertices.*

Proof Let (a, b) be a valid edge whose face $F_{a,b}$ minimize the maximum number of vertices in a connected component of $G \setminus (N_1(P_G(a, r)) \cup N_1(P_G(b, r)))$. Break ties by choosing the valid edge whose face $F_{a,b}$ contains smallest number of active vertices on the same side as the maximum connected component. If ties remain, choose arbitrarily. As before assume $c = nca(a, b)$.

Let C be the largest connected component. Clearly, since (a, b) is valid, all the vertices of C must totally lie either inside face $F_{a,b}$ or outside face $F_{a,b}$. Without loss of generality assume the vertices of C are inside face $F_{a,b}$. If C contains less then $2n/3$ vertices, then we are done. Hence, assume $|C| > 2n/3$.

By Lemma 42, there is a vertex d inside $F_{a,b}$ such that both (a, d) and (d, b) are

valid. Since both (a, d) and (d, b) are valid, one of the following must be true.

1. $C \cap N_1(P_G(d, r)) = \emptyset$. That is, all vertices of C are either totally inside $F_{a,d}$ or $F_{d,b}$.

2. $C \cap N_1(P_G(d, r)) \neq \emptyset$.

Let's consider the first case. Without loss of generality, assume all vertices of C are totally inside $F_{a,d}$. Since (a, d) is a valid edge and the number of active vertices inside $F_{a,d}$ is strictly less then the number active vertices inside $F_{a,b}$, by our rule, (a, d) will be chosen, a contradiction. Hence, this case is impossible.

Now, consider the second case. Let n_1 be the number of vertices which are both in C and inside $F_{a,d}$, but not in $N_1(P_G(d, r))$. Let n_2 be the number of vertices which are both in C and inside $F_{d,b}$, but not in $N_1(P_G(d, r))$. Set $n_3 = |C \cap N_1(P_G(d, r))|$. Clearly, $n_1 + n_2 + n_3 \geq 2n/3$. We claim that $n_1 + n_3 \geq n/3$ or $n_2 + n_3 \geq n/3$. Assume neither of the inequalities holds. Then $n_1 + n_2 + n_3 \leq n_1 + n_2 + n_2 + n_3 < 2n/3$, a contradiction. Hence the claim is true. Without loss of generality, assume $n_1 + n_3 \geq n/3$. Consider the valid edge (a, d). Clearly, each connected component of $G \setminus \{N_1(P_G(a, r)) \cup N_1(P_G(d, r))\}$ must consist of the the vertices that is either inside $F_{a,d}$ or outside $F_{a,d}$.

There are two cases. First case is the maximum connected component of $G \setminus \{N_1(P_G(a, r)) \cup N_1(P_G(d, r))\}$ consists of vertices which are all inside $F_{a,d}$. Since both (a, b) and (a, d) are valid, any connected component of $G \setminus \{N_1(P_G(a, r)) \cup N_1(P_G(d, r))\}$ are also part of the connected components of $G \setminus \{N_1(P_G(a, r)) \cup N_1(P_G(b, r))\}$. Therefore, the number of vertices in the maximum connected components of $G \setminus \{N_1(P_G(a, r)) \cup N_1(P_G(d, r))\}$ is strictly less than $|C|$. This contradicts with our choice of (a, b). Hence, this case is impossible. The second case is the maximum connected component of $G \setminus \{N_1(P_G(a, r)) \cup N_1(P_G(d, r))\}$ consists of vertices which are all outside $F_{a,d}$. By assumption, $n_1 + n_3 \geq n/3$, this implies that the number of vertices in the maximum

connected component outside $F_{a,d}$ must be strictly less than $2n/3$ which is also strictly less than $|C|$. This also contradicts with our choice of (a, b). This concludes our proof of the lemma.

□

From the discussion above, we conclude that the two separating pathes can be found in polynomial time. Hence, we have the following theorem.

Theorem 29 *Assume G is a unit disk graph. Then, there are two shortest pathes $P_G(a, r)$ and $P_G(b, r)$ such that no connected component of $G \backslash (N_1(P_G(a, r)) \cup N_1(P_G(b, r)))$ contains more than $2n/3$ vertices. Moreover, such two vertices can be found in polynomial time.*

From Theorem 29, one can construct a system of $O(\log_2 n)$ collective (3, 4)-spanning trees for UDG. The details can be found in paper [53].

3.5 AT-free related graphs

For a set $S \subseteq V$, by $N[S] := \bigcup_{v \in S} N[v]$ we denote the *closed neighborhood* of S and by $N(S) := N[S] \setminus S$ the *open neighborhood* of S. A set $D \subseteq V$ is called a *dominating set* of a graph $G = (V, E)$ if $N[D] = V$.

An independent set of three vertices such that each pair is joined by a path that avoids the neighborhood of the third is called an *asteroidal triple*. A graph G is an *AT-free graph* if it does not contain any asteroidal triples [36]. In [80], the notion of asteroidal triple was generalized. An independent set $A \subseteq V$ of a graph $G = (V, E)$ is called an *asteroidal set* of G if for each $a \in A$ the vertices of $A \setminus \{a\}$ are contained in one connected component of $G - N[a]$, the graph obtained from G by removing vertices of $N[a]$. The maximum cardinality of an asteroidal set of G is denoted by $\mathsf{an}(G)$, and called the *asteroidal number* of G. The class of *graphs of bounded asteroidal number* extends naturally the class of AT-free graphs; AT-free graphs are exactly the graphs

with asteroidal number at most two.

Let P be a shortest u, v-path of G, for some pair of vertices u, v. If every vertex z of G belongs to the neighborhood $N[P]$ of P, then we say that P is a *dominating shortest path* of G. A graph G is called a *Dominating-Shortest-Path–graph* (or *DSP–graph*, for short), if it has a dominating shortest path. By the Dominating Pair Theorem given in [36], any AT-free graph is a DSP-graph.

The class of AT-free graphs contains many intersection families of graphs, among them the permutation graphs, the trapezoid graphs and the cocomparability graphs. These three families of graphs can be defined as follows [23, 72]. Consider two parallel lines (upper and lower) in the plane. Assume that each line contains n points, labeled 1 to n, and each two points with the same label define a segment with that label. The intersection graph of such a set of segments between two parallel lines is called a *permutation graph*. Assume now that each line contains n intervals, labeled 1 to n, and each two intervals with the same label define a trapezoid with that label (a trapezoid can degenerate to a triangle or to a segment). The intersection graph of such a set of trapezoids between two parallel lines is called a *trapezoid graph*. Clearly, every permutation graph is a trapezoid graph, but not vice versa. The class of cocomparability graphs (which contains all trapezoid graphs as a subclass) can be defined as the intersection graphs of continuous function diagrams, but for this paper it is more convenient to define them via the existence of a special vertex ordering. A graph G is a *cocomparability graph* if it admits a vertex ordering $\sigma = [v_1, v_2, \ldots, v_n]$, called a *cocomparability ordering*, such that for any $i < j < k$, if v_i is adjacent to v_k then v_j must be adjacent to at least one of v_i, v_k. According to [94], such an ordering of a cocomparability graph can be constructed in linear time. Note also that, given a permutation graph G, a *permutation model* (i.e., a set of segments between two parallel lines, defining G) can be found in linear time [94]. A *trapezoid model* for a trapezoid graph can be found in $O(n^2)$ time

[92].

3.5.1 AT-free graphs

It is known [101] that any AT-free graph admits one additive tree 3-spanner. In this subsection we show that any AT-free graph admits a system of two collective additive tree 2-spanners.

As a consequence of the Dominating Pair Theorem given in [36], any AT-free graph has a dominating shortest path that can be found in linear time by $2\times LexBFS$ [35]. The $2 \times LexBFS$ method first starts a *lexicographic breadth-first search* (*LexBFS*) from an arbitrary vertex x of G and then starts a second LexBFS from the vertex x_0 last visited by the first LexBFS. Let x_l be the vertex of G last visited by the second LexBFS. As shown in [35], every shortest path $(x_0, x_1, \ldots, x_l)$, connecting x_0 and x_l, is a dominating shortest path of G. Next we demonstrate how to use such a dominating shortest path in an AT-free graph to show that every AT-free graph admits a system of two collective additive tree 2-spanners. We will need the following result from [79].

Lemma 44 *[79] Let $P := (x_0, x_1, \ldots, x_l)$ be a dominating shortest path of an AT-free graph $G = (V, E)$ constructed by $2 \times LexBFS$. Then, for every $i = 1, 2, \ldots, l$, every vertex $z \in N_i(x_0)$ is adjacent to x_i or x_{i-1}.*

Using this lemma, we construct a first spanning tree $T_1 = (V, E_1)$ for an AT-free graph $G = (V, E)$ as follows: put into initially empty E_1 all edges of the path $P := (x_0, x_1, \ldots, x_l)$, and then for each vertex $z \in N_i(x_0)$, put edge zx_{i-1} into E_1, if z is adjacent to x_{i-1} in G, and put edge zx_i into E_1, otherwise. We call this spanning tree the *caterpillar-tree* of G (with *spine* P). According to [101], this caterpillar-tree gives already an additive tree 3-spanner for the AT-free graph G. To get a collective additive stretch factor 2 for G, we construct a second spanning tree $T_2 = (V, E_2)$ for G as follows. Set $L_i := N_i(x_0)$ for each $i = 1, 2, \ldots, l$.

set $E_2 := \{$all edges of the path $P := (x_0, x_1, \ldots, x_l)\}$;

set $dev(x_i) := 0$ for each vertex x_i of the path P;

for $i = 1$ to l do

  for each vertex $z \in L_i \setminus \{x_i\}$ do

    among all neighbors of z in L_{i-1} choose a neighbor w with minimum

    deviation $dev(w)$;

    add edge zw to E_2 and set $dev(z) := dev(w) + 1$;

  enddo

enddo.

We call spanning tree T_2 the *cactus-tree* of G (with *stem* P). It is evident, by construction, that the cactus-tree T_2 is a special kind of *breadth-first-search–tree* of G. The value $dev(z)$ (called the *deviation of z from stem P*) gives the distance in T_2 between vertex z and path P. In Figure 3.11 we show an AT-free graph G along with its caterpillar-tree T_1 and cactus-tree T_2.

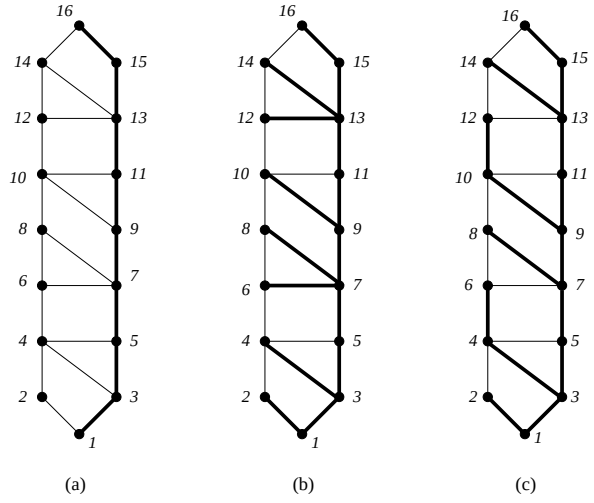


Figure 3.11: (a) An AT-free graph G with a dominating path P, (b) the caterpillar-tree T_1 of G and (c) the cactus-tree T_2 of G.

Lemma 45 *Spanning trees $\{T_1, T_2\}$ are collective additive tree 2-spanners of AT-free graph G.*

Proof Consider two arbitrary vertices $x \in L_i$ and $y \in L_j$ $(y \neq x)$ of G, where $j \leq i$. If $i = j$, i.e., both x and y lie in the same layer $L_i = L_j$, then the distance in T_1 between x and y is at most 3, since in the worst case one of them is adjacent to x_i in T_1 and the second to x_{i-1}. Thus, $d_{T_1}(x, y) \leq 3 \leq d_G(x, y) + 2$ holds when $i = j$, and therefore, we may assume that $i > j$.

We know that $d_G(x, y) \geq i-j$. By the construction of the caterpillar-tree T_1, we have $d_{T_1}(y, x_j) \leq 2$ and $d_{T_1}(x, x_{i-1}) \leq 2$. Hence, $d_{T_1}(x, y) \leq d_{T_1}(x, x_{i-1}) + d_{T_1}(x_{i-1}, x_j) + d_{T_1}(y, x_j) \leq 2 + i - 1 - j + 2 \leq d_G(x, y) + 3$, and equality $d_{T_1}(x, y) = d_G(x, y) + 3$ holds if and only if $d_G(x, y) = i - j$, vertex x is adjacent to x_i in T_1 (and thus in G, vertex x is not adjacent to x_{i-1}) and vertex y is adjacent to x_{j-1} in T_1 but does not coincide with x_j. We will show that in this case in the cactus-tree T_2, $d_{T_2}(x, y) \leq d_G(x, y) + 2$.

Consider in G a shortest path $(y = y_0, y_1, \ldots, y_{i-j} = x)$ connecting vertices y and x. Clearly, $y_k \in L_{j+k}$ for each $k = 0, 1, \ldots, i-j-1$, and since y_k is a neighbor of y_{k+1} in layer L_{j+k}, by construction of T_2, we have $dev(y_0) = 1$ and $dev(y_{k+1}) \leq dev(y_k) + 1 \leq k + 2$. Hence, the deviation of vertex x is at most $i - j + 1$. That is, there is a path in T_2 between x and a stem vertex x_s $(j - 1 \leq s \leq i - 2)$ of length $i - s$. The latter implies the existence in T_2 of a path of length $i - j + 1$ between vertices x and x_{j-1}. Therefore, $d_{T_2}(x, y) \leq d_{T_2}(x, x_{j-1}) + 1 = i - j + 1 + 1 = d_G(x, y) + 2$.

From this lemma we immediately conclude.

Theorem 30 *Any AT-free graph admits a system of two collective additive tree 2-spanners, constructable in linear time.*

In the next subsection, we will show that to get a collective additive stretch factor 2 for some AT-free graphs, one needs at least two spanning trees. Therefore, the result

given in Theorem 30 is best possible. Furthermore, to achieve a collective additive stretch factor 1 or 0 for some AT-free graphs, we will show that one needs $\Omega(n)$ spanning trees.

3.5.2 Permutation graphs and trapezoid graphs

It is known [93] that any permutation graph admits a multiplicative tree 3-spanner. In this subsection, we show that any permutation graph admits an additive tree 2-spanner and any system of collective additive tree 1–spanners must have $\Omega(n)$ spanning trees for some permutation graphs. Here also we disprove a conjecture given in [101], that any cocomparability graph admits an additive tree 2-spanner. We show that there exists even a trapezoid graph that does not admit any additive tree 2-spanner.

Let $G = (V, E)$ be a permutation graph given together with a permutation model. In what follows, "u.p." and "l.p." refer to a vertex's point on the upper and lower, respectively, line of the permutation model. Construct BFS-layers ($\{L_0, L_1, \cdots\}$) and the spine $\{x_1, x_2, \cdots\}$ of G as follows (the process continues until $L_i = \emptyset$).

set x_0 := the vertex whose u.p. is as far left as possible;
set $L_0 := \{x_0\}$;
set $L_1 :=$ {vertices whose l.p.s are to the left of the l.p. of x_0};
set x_1 := the vertex in L_1 with the u.p. as far right as possible;
set $L_2 :=$ {vertices whose u.p.s are between the u.p.s of x_0 and x_1} $\setminus L_1$;
set x_2 := the vertex in L_2 with the l.p. as far right as possible;
for $i = 3$ to n do
  if i is odd then
    set $L_i :=$ {vertices with l.p. between the l.p.s of x_{i-3} and x_{i-1}} $\setminus L_{i-1}$;
    set x_i := the vertex in L_i with the u.p. as far right as possible;
  else

set $L_i := \{$vertices with u.p. between the u.p.s of x_{i-3} and $x_{i-1}\} \setminus L_{i-1}$;

set $x_i :=$ the vertex in L_i with the l.p. as far right as possible;

enddo.

Lemma 46 *For all $i \geq 0$, if $y \in L_{i+1}$ then $x_i y \in E$.*

Proof The lemma is clearly true for $i = 0$ or $i = 1$. Now assume $i \geq 2$ and i is even. (A similar proof holds for i odd.) By the construction of the L sets, it is clear that since $y \in L_{i+1}$, the l.p. of y is between the l.p.s of x_{i-2} and x_i and the u.p. of y is between the u.p.s of x_{i-1} and x_{i+1}. Furthermore, since the u.p. of x_i is to the left of the u.p. of x_{i-1} and thus to the left of the u.p. of y, we conclude that $x_i y \in E$.

Note that as an immediate corollary of this lemma, the set $\{L_i : i = 1, 2, \cdots\}$ forms a BFS layering. This leads to the following theorem.

Theorem 31 *Every permutation graph admits an additive tree 2-spanner, constructable in linear time.*

Proof Form the tree T by choosing all edges from x_i to L_{i+1}, for all appropriate i. Now consider any two vertices u and v, where $u \in L_i$ and $v \in L_j$, $i \leq j$. If $i = j$, then $d_G(u, v) = 1$ or 2 and $d_T(u, v) = 2$ (because of vertex x_{i-1} and Lemma 46). If $i < j$, then $d_G(u, v) \geq j - i$. If u is a spine vertex, then $d_T(u, v) = j - i$. Otherwise, by following the path $v, x_{j-1}, \cdots, x_{i-1}, u$, we see that $d_T(u, v) = j - i + 2$. Thus in all cases, $d_G(u, v) \leq d_T(u, v) + 2$, as required.

We now show that there exists a trapezoid graph that does not admit any additive tree 2-spanner, thereby disproving a conjecture from [101] that any cocomparability graph admits an additive tree 2-spanner.

Consider the trapezoid graph G depicted in Figure 3.12, and assume that it has an additive tree 2-spanner T. We claim then, that all the cut edges (edges whereby

the removal of their endpoints disconnects G) of G must belong to T. Indeed, if, for example, neither edge $(7,6)$ nor $(7,8)$ is an edge of T, then in T, vertices 7 and 6 must be connected by path $(7,5,4,6)$ and vertices 7 and 8 must be connected by path $(7,9,10,8)$. Now $d_T(8,6) = 6$ whereas $d_G(8,6) = 1$. If exactly one of $(7,6),(7,8)$ (without loss of generality $(7,6)$) is an edge of T, then at most three of the edges $(7,9),(9,10),(10,8),(8,6)$ may be in T. For at least one of the remaining edges, the distance in T between its endpoints is at least 4 contradicting T being an additive 2-spanner.

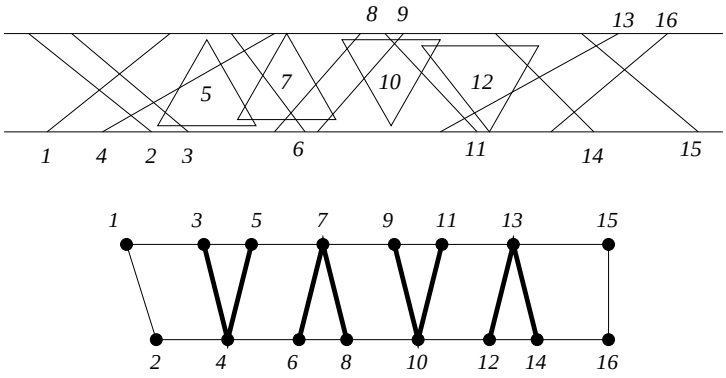


Figure 3.12: A trapezoid graph (with a trapezoid model) that does not admit an additive tree 2-spanner.

Thus, if G has an additive tree 2-spanner T, then all the cut edges of G must belong to T (see Figure 3.12). In T, paths $(9,10,11)$ and $(6,7,8)$ must be connected either by edge $(7,9)$ or by edge $(8,10)$. If $(7,9)$ is a tree edge, then we get $d_T(8,14) = 6 = d_G(8,14)+3$ (independent of whether edge $(11,13)$ or edge $(10,12)$ is a tree edge). Otherwise, if $(8,10)$ is a tree edge, then we get $d_T(3,9) = 6 = d_G(3,9) + 3$. Contradictions with T being an additive tree 2-spanner of G prove the following result.

Observation 4 *There are trapezoid graphs that do not admit any additive tree 2-spanners.*

The next observation gives a lower bound on the number of spanning trees that

guarantee a collective additive stretch factor 1 for bipartite permutation graphs (in fact, for all graph families containing complete bipartite graphs).

Observation 5 *There are bipartite permutation graphs on $2n$ vertices for which any system of collective additive tree 1–spanners needs to have at least $\Omega(n)$ spanning trees.*

Proof Consider the complete bipartite graph $G = K_{n,n}$ on $2n$ vertices (which is clearly a permutation graph), and let $\mathcal{T}(G)$ be a system of μ collective additive tree 1-spanners of G. Then, for any two adjacent vertices x and y of G there must exist a spanning tree T in $\mathcal{T}(G)$ such that $d_T(x, y) \leq 2$. If $d_T(x, y) = 2$ then a common neighbor z of x and y in G would form a triangle with vertices x and y, which is impossible for $G = K_{n,n}$. Hence, $d_T(x, y) = 1$ must hold. Thus, every edge xy of G is an edge of some tree $T \in \mathcal{T}(G)$. Since there are n^2 graph edges to cover by spanning trees from $\mathcal{T}(G)$, we conclude $\mu \geq n^2/(2n-1) > n/2$.

3.5.3 DSP-graphs

It follows from a result in [101] that any DSP-graph admits one additive tree 4-spanner. In this subsection we show that any DSP-graph admits a system of two collective additive tree 3-spanners and a system of five collective additive tree 2-spanners.

Let $G = (V, E)$ be a DSP-graph and let $P := (v = x_0, x_1, \ldots, x_l = u)$ be a dominating shortest path of G. We will build five spanning trees $\{T_1, T_2, T_3, T_4, T_5\}$ for G, all containing the edges of P, in such a way that for any two vertices $x, y \in V$, there will be a tree $T' \in \{T_1, T_2, T_3, T_4, T_5\}$ with $d_{T'}(x, y) \leq d_G(x, y) + 2$.

Our first three trees T_1, T_2, T_3 are very similar to the trees constructed for AT-free graphs. The tree $T_1 = (V, E_1)$ is constructed as follows. Add to initially empty set E_1 all edges of path P. Then, for each vertex $z \in V \setminus P$ choose an arbitrary neighbor w_z in P and add edge zw_z to E_1. The tree T_1 is an analog of the caterpillar-tree constructed for an AT-free graph. The second and third trees are analogs of the cactus-tree considered

for an AT-free graph. The tree $T_2 = (V, E_2)$ is a special breadth-first-search-tree T_v with vertex v as the root, the tree $T_3 = (V, E_3)$ is a special breadth-first-search-tree T_u with vertex u as the root. We show here only how to construct the tree T_3. For construction of T_2 we can use the algorithm given in Subsection 3.5.1 with one additional line at the end: for each $z \in N_{l+1}(v)$, add edge zu to E_2 and set $dev(z) := 1$. T_3 is constructed similarly, we simply reverse the order of vertices of P and consider u instead of v and E_3 instead of E_2.

set $E_3 := \{$all edges of the path $P := (v = x_0, x_1, \ldots, x_l = u)\}$;
set $dev(x_i) := 0$ for each vertex x_i of the path P;
for $i = 1$ to l do
  for each vertex $z \in N_i(u) \setminus \{x_{l-i}\}$ do
    among all neighbors of z in $N_{i-1}(u)$ choose a neighbor w with minimum
    deviation $dev(w)$;
    add edge zw to E_3 and set $dev(z) := dev(w) + 1$;
  enddo
enddo
for each $z \in N_{l+1}(u)$, add edge zv to E_3 and set $dev(z) := 1$.

Our tree $T_4 = (V, E_4)$ is a generalization of the tree T_2 and is constructed as follows.

set $E_4 := \{$all edges of the path $P := (v = x_0, x_1, \ldots, x_l = u)\}$;
set $dev(x_i) := 0$ for each vertex x_i of the path P;
for $i = 1$ to l do
  for each vertex $z \in N_i(v) \setminus \{x_i\}$ do case
    case (z is adjacent to x_{i-1} in G)
      add edge zx_{i-1} to E_4 and set $dev(z) := 1$;
    case (z is adjacent to x_i in G)

add edge zx_i to E_4 and set $dev(z) := 1$;

case (z is adjacent to a vertex $w \in N_i(v)$ that is adjacent to x_{i-1})

add such a w and add edge zw to E_4 and set $dev(z) := 2$;

otherwise /* none of above */

among all neighbors of z in $N_{i-1}(v)$ choose a neighbor w with

minimum deviation $dev(w)$ (break ties arbitrarily);

add edge zw to E_4 and set $dev(z) := dev(w) + 1$;

endcase

enddo

enddo

for each $z \in N_{l+1}(v)$, add edge zu to E_4 and set $dev(z) := 1$.

It is an easy exercise to show by induction that for any vertex $z \in N_i(v)$, the vertex of P closest to z in T_4 is either x_s or x_{s-1} with $s = i - dev(z) + 1$. Moreover, the length of the path of T_4 between z and P is $dev(z)$. Our last tree $T_5 = (V, E_5)$ is a version of the tree T_4, constructed downwards.

set $E_5 := \{$all edges of the path $P := (v = x_0, x_1, \ldots, x_l = u)\}$;

set $dev(x_i) := 0$ for each vertex x_i of the path P and $dev(z) := \infty$ for any $z \in V \setminus P$;

for $i = l - 1$ down to 1 do

for each vertex $z \in N_i(v) \setminus \{x_i\}$ do case

case (z is adjacent to x_{i+1} in G)

add edge zx_{i+1} to E_5 and set $dev(z) := 1$;

case (z is adjacent to x_i in G)

add edge zx_i to E_5 and set $dev(z) := 1$;

case (z is adjacent to a vertex $w \in N_i(v)$ that is adjacent to x_{i+1})

choose such a w and add edge zw to E_5 and set $dev(z) := 2$;

otherwise /* none of the above */

if z has neighbors in $N_{i+1}(v)$ then

among all neighbors of z in $N_{i+1}(v)$ choose a neighbor w with

minimum deviation $dev(w)$ (break ties arbitrarily);

if $dev(w) < \infty$ then add edge zw to E_5 and set $dev(z) := dev(w) + 1$;

endcase

enddo

enddo

for each vertex z with $dev(z)$ still ∞ do

let $z \in N_i(v)$;

if $i = l$ and z is adjacent to x_l then add edge zx_l to E_5 and set $dev(z) := 1$;

else add edge zx_{i-1} to E_5;

/* this edge exists in G since P is a dominating path

and z is adjacent in G neither to x_{i+1} nor x_i */

enddo.

Again, it is easy to see that for any vertex $z \in N_i(v)$ with finite deviation $dev(z)$, the vertex of P closest to z in T_5 is either x_s or x_{s+1} with $s = i + dev(z) - 1$. The length of the path of T_5 between z and P is $dev(z)$. In Figure 3.13 we show a DSP-graph G along with a shortest path P and spanning trees T_1, T_2, T_3, T_4, T_5.

We are ready to present the main result of this subsection.

Lemma 47 *Let G be a DSP-graph with a dominating shortest path $P := (v = x_0, x_1, \ldots, x_l = u)$ and spanning trees T_1, T_2, T_3, T_4, T_5 constructed starting from P as described above. Then, for any two vertices $x, y \in V$ the following is true.*

1. $d_{T_1}(x, y) \leq d_G(x, y) + 4$;

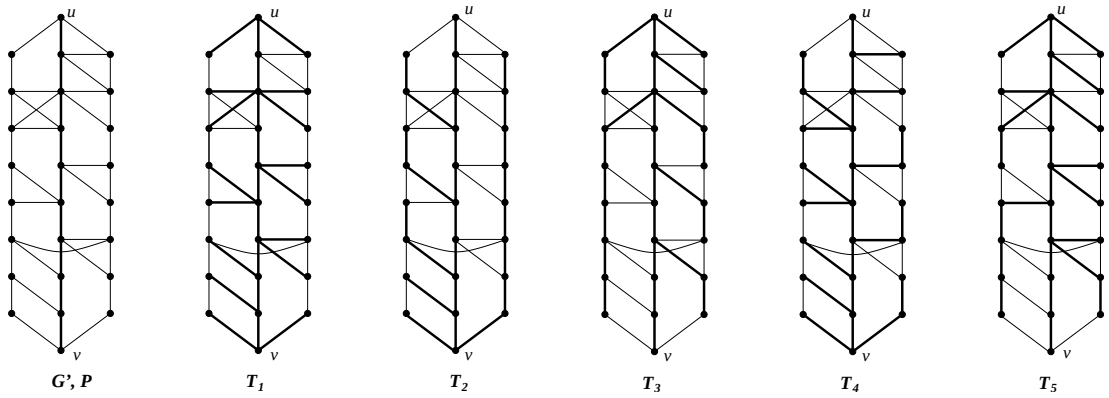


Figure 3.13: A DSP-graph G with a dominating path P and spanning trees T_1, T_2, T_3, T_4, T_5.

2. *there is a tree* $T' \in \{T_1, T_2\}$ *such that* $d_{T'}(x, y) \leq d_G(x, y) + 3$*;*

3. *there is a tree* $T'' \in \{T_1, T_2, T_3, T_4, T_5\}$ *such that* $d_{T''}(x, y) \leq d_G(x, y) + 2$.

Proof Consider two arbitrary vertices $x, y \in N[P]$ $(y \neq x)$, and let $x \in N_i(v)$, $y \in N_j(v)$ and $i \geq j$. We know that $d_G(x, y) \geq i - j$. By the construction of tree T_1, $d_{T_1}(y, x_j) \leq 2$ and $d_{T_1}(x, x_i) \leq 2$. Hence, $d_{T_1}(x, y) \leq d_{T_1}(x, x_i) + d_{T_1}(x_i, x_j) + d_{T_1}(y, x_j) \leq 2 + i - j + 2 \leq d_G(x, y) + 4$, thereby proving claim 1 of the lemma.

Equality $d_{T_1}(x, y) = d_G(x, y) + 4$ holds if and only if $d_G(x, y) = i - j$, vertex x is adjacent to x_{i+1} in T_1, vertex y is adjacent to x_{j-1} in T_1 and $x \neq x_i, y \neq x_j$. As in the proof of Lemma 45, we now show that in this case (or more generally, when $d_G(x, y) = i - j$ and y is adjacent to x_{j-1} in G), $d_{T_2}(x, y) \leq d_G(x, y) + 2$.

Consider in G a shortest path $(y = y_0, y_1, \ldots, y_{i-j} = x)$ connecting vertices y and x. Clearly, $y_k \in N_{j+k}(v)$ for each $k = 0, 1, \ldots, i - j - 1$, and since y_k is a neighbor of y_{k+1} in layer $N_{j+k}(v)$, by construction of T_2, we have $dev(y_0) = 1$ and $dev(y_{k+1}) \leq dev(y_k) + 1 \leq k + 2$. Hence, the deviation of vertex x is at most $i - j + 1$. That is, there is a path in T_2 between x and a vertex $x_s \in P$ $(j - 1 \leq s \leq i - 1)$ of length $i - s$. The latter implies the existence in T_2 of a path of length $i - j + 1$ between vertices x and

x_{j-1}. Therefore, $d_{T_2}(x,y) \le d_{T_2}(x,x_{j-1}) + 1 = i - j + 1 + 1 = d_G(x,y) + 2$.

Together with claim 1 of the lemma, this proves claim 2. We may assume in what follows that $d_G(x,y) > i-j$ or y is not adjacent to x_{j-1} in G (since otherwise, $d_{T_2}(x,y) \le d_G(x,y) + 2$).

Case 1: y is not adjacent to x_{j-1} in G.

Then y is adjacent to x_j or x_{j+1} in T_1. Hence, $d_{T_1}(x,y) \le d_{T_1}(x,x_i) + d_{T_1}(x_i,x_{j+1}) + d_{T_1}(y,x_{j+1}) \le 2 + i - j - 1 + 2 \le d_G(x,y) + 3$, and equality $d_{T_1}(x,y) = d_G(x,y) + 3$ holds if and only if $d_G(x,y) = i - j$, vertex x is adjacent to x_{i+1} in T_1, vertex y is adjacent to x_j in T_1 and $x \ne x_i$. Since $y \in N_j(v)$, $x \in N_i(v)$ and $u = x_l$ belongs to $N_l(v)$, we get $d_G(u,x) = l - i$ and $d_G(u,y) \ge l - j$. On the other hand, $d_G(y,u) \le d_G(y,x) + 1 + d_G(x_{i+1},u) = i - j + 1 + l - (i+1) = l - j$. Hence, $x \in N_{l-i}(u), y \in N_{l-j}(u)$ and $x_{i+1} \in N_{l-i-1}(u)$. From this and since x is adjacent to x_{i+1} in G and there exists a shortest path of length $i - j$ in G between vertices x and y, it is easy to show that for the special breadth-first-search–tree T_3 rooted at u, $d_{T_3}(x,y) \le d_{T_3}(y,x_{i+1}) + 1 = i + 1 - j + 1 = d_G(x,y) + 2$ (the proof is similar to the proof above for the tree T_2).

Case 2: $d_G(x,y) \ge i - j + 1$.

Again, for the tree T_1 we have $d_{T_1}(x,y) \le d_{T_1}(x,x_i) + d_{T_1}(x_i,x_j) + d_{T_1}(y,x_j) \le 2 + i - j + 2 \le d_G(x,y) + 3$, and equality $d_{T_1}(x,y) = d_G(x,y) + 3$ holds if and only if $d_G(x,y) = i - j + 1$, vertex x is adjacent to x_{i+1} in T_1, vertex y is adjacent to x_{j-1} in T_1 and $x \ne x_i$, $y \ne x_j$. We show that in this case, $\min\{d_{T_4}(x,y), d_{T_5}(x,y)\} \le d_G(x,y) + 2$.

Consider in G a shortest path $(y = y_0, y_1, \ldots, y_{i-j+1} = x)$ connecting vertices y and x. Clearly, there is only one index h $(0 \le h \le i - j)$ such that both vertices y_h and y_{h+1} lie in $N_{j+h}(v)$. For index k $(0 \le k \le h - 1)$, $y_k \in N_{j+k}(v)$, $y_{k+1} \in N_{j+k+1}(v)$.

If y_{h+1} is adjacent to x_{j+h-1} or to x_{j+h} then, by construction of tree T_4, we have $dev(y_{h+1}) = 1$ and $dev(y_{t+1}) \le dev(y_t) + 1 \le t - (h+1) + 2$ for any t, $(h + 1 \le t \le i - j)$.

Hence, the deviation of vertex x is at most $i-(j+h-1)$, and there is a path in T_4 of length $dev(x)$ from vertex x to either x_s or x_{s-1}, where $s = i-dev(x)+1 \geq i-i+(j+h-1)+1 = j+h$. In any case $d_{T_4}(x, x_{s-1}) \leq dev(x)+1$ and therefore $d_{T_4}(x, x_{j-1}) \leq d_{T_4}(x, x_{s-1})+d_{T_4}(x_{s-1}, x_{j-1}) \leq dev(x)+1+s-j = dev(x)+1+i-dev(x)+1-j = i-j+2$. The latter implies $d_{T_4}(x, y) \leq d_{T_4}(x, x_{j-1})+1 \leq i-j+2+1 = d_G(x, y)+2$.

So, we may assume that y_{h+1} is adjacent neither to x_{j+h-1} nor to x_{j+h}, i.e., y_{h+1} is adjacent to x_{j+h+1}. Assume that y_h is adjacent to x_{j+h-1} and $h \neq 0$. Then, by construction of tree T_4, $dev(y_{h+1}) = 2$ and $dev(y_{t+1}) \leq dev(y_t)+1 \leq t-(h+1)+3$ for any t, $(h+1 \leq t \leq i-j)$. Hence, the deviation of vertex x is at most $i-(j+h-1)+1$, and there is a path in T_4 of length $dev(x)$ from vertex x to either x_s or x_{s-1}, where $s = i-dev(x)+1 \geq i-i+(j+h-1)-1+1 = j+h-1$. Recall that $h \neq 0$ and hence $s-1 \geq j-1$. Again, in any case $d_{T_4}(x, x_{s-1}) \leq dev(x)+1$ and therefore $d_{T_4}(x, x_{j-1}) \leq d_{T_4}(x, x_{s-1})+d_{T_4}(x_{s-1}, x_{j-1}) \leq dev(x)+1+s-j = dev(x)+1+i-dev(x)+1-j = i-j+2$. The latter implies $d_{T_4}(x, y) \leq d_{T_4}(x, x_{j-1})+1 \leq i-j+2+1 = d_G(x, y)+2$.

If now y_h is adjacent to x_{j+h+1} or to x_{j+h} then, by construction of tree T_5, $dev(y_h) = 1$ and $dev(y_{t-1}) \leq dev(y_t)+1 \leq h-t+2$ for any t, $(1 \leq t \leq h)$. Hence, the deviation of vertex y is at most $h+1$, and there is a path in T_5 of length $dev(y)$ from vertex y to either x_s or x_{s+1}, where $s = j+dev(y)-1 \leq j+h+1-1 = j+h$. In any case $d_{T_5}(y, x_{s+1}) \leq dev(y)+1$ and therefore $d_{T_5}(y, x_{i+1}) \leq d_{T_5}(y, x_{s+1})+d_{T_5}(x_{s+1}, x_{i+1}) \leq dev(y)+1+i-s = dev(y)+1+i-j-dev(y)+1 = i-j+2$. The latter implies $d_{T_5}(x, y) \leq d_{T_5}(y, x_{i+1})+1 \leq i-j+2+1 = d_G(x, y)+2$. Thus, we may assume that y_h is adjacent to neither x_{j+h+1} nor x_{j+h}.

It remains then to consider only the last case when $h = 0$, y_1 is adjacent to x_{j+1} but not to x_j,x_{j-1}, and $y = y_0$ is adjacent to x_{j-1} but not to x_j,x_{j+1}. Recall that $y_0, y_1 \in N_j(v)$. By construction, in T_5, y is adjacent to y_1, y_1 is adjacent to x_{j+1}, and x is adjacent to x_{i+1}. Therefore, $d_{T_5}(x, y) \leq 1+d_{T_5}(x_{i+1}, x_{j+1})+2 = i-j+3 = d_G(x, y)+2$,

completing the proof of the lemma.

Theorem 32 *Any DSP-graph admits one additive tree 4-spanner, a system of two collective additive tree 3-spanners and a system of five collective additive tree 2-spanners. Moreover, given a dominating shortest path of G, all trees are constructable in linear time.*

Note that an induced cycle on six vertices gives a DSP-graph that does not admit an additive tree 3-spanner. Therefore, two trees are necessary to get a collective additive stretch factor 3. However, it is an open question whether to achieve a collective additive stretch factor 2, one really needs five spanning trees.

3.5.4 Graphs with bounded asteroidal number

It is known [80] that any graph G with asteroidal number $\mathsf{an}(G)$ admits an additive tree $(3\mathsf{an}(G)-1)$-spanner. In this subsection we show that any graph with asteroidal number $\mathsf{an}(G)$ admits a system of $\mathsf{an}(G)(\mathsf{an}(G)-1)/2$ collective additive tree 4-spanners and a system of $\mathsf{an}(G)(\mathsf{an}(G)-1)$ collective additive tree 3-spanners.

In what follows we will use the following two definitions and two theorems from [80]. An asteroidal set A of a graph G is *repulsive* if for every vertex $v \in V \setminus N[A]$, not all vertices of A are contained in one connected component of $G - N[v]$.

Theorem 33 *[80] Any graph has a repulsive asteroidal set.*

A set $D \subseteq V$ in a graph G is said to be a *dominating target*, if $D \cup S$ is a dominating set in G for every set $S \subseteq V$ for which the subgraph of G induced by $D \cup S$ is connected.

Theorem 34 *[80] Any graph G has a dominating target D with $|D| \leq \mathsf{an}(G)$. Furthermore, every repulsive asteroidal set of G is such a dominating target of G.*

We will need a stronger version of the above theorem. Let $G = (V, E)$ be a graph and $D \subseteq V$ be a repulsive asteroidal set of G.

Lemma 48 *For every $x, y \in V$ and dominating target D, there exist $a, b \in D$ such that $x, y \in N[P]$ for any path P of G between a and b.*

Proof The claim is evident if $x, y \in N[D]$. Therefore assume, without loss of generality, that $x \in V \setminus N[D]$. Then, by the definition of a repulsive asteroidal set, there must exist vertices a, b in D that lie in different connected components of $G - N[x]$. Let $CC_x(a)$ and $CC_x(b)$ be those connected components. Clearly, any path P between a and b intersects $N[x]$ and therefore $x \in N[P]$. We may assume that $y \notin N[\{a, b\}]$, since otherwise the lemma follows. Let, however, $y \in N[D]$ and c be a vertex from $N[y] \cap D$. Since c cannot be adjacent with x (recall that $x \in V \setminus N[D]$), c must lie in some connected component $CC_x(c)$ of $G - N[x]$. Clearly, $CC_x(c)$ cannot coincide with both $CC_x(a)$ and $CC_x(b)$. Assuming $CC_x(c) \neq CC_x(a)$, we conclude that any path connecting a and c dominates both x and y.

Now let $y \in V \setminus N[D]$. Then, there must exist vertices c, d in D that lie in different connected components of $G - N[y]$. Hence, for any path P between c and d, $y \in N[P]$. If c and d lie in different connected components of $G - N[x]$ as well, then the lemma follows, since any path connecting c and d must dominate both x and y. Therefore assume, without loss of generality, that neither c nor d belongs to $CC_x(a)$. Now any path $P(a, c)$ between a and c and any path $P(a, d)$ between a and d must intersect $N[x]$. Since the union of $P(a, c)$ and $P(a, d)$ connects vertices c and d, $N[y]$ must intersect every path between a and c or every path between a and d. Let a and c be vertices lying in different connected components of $G - N[y]$. Then, since a and c lie in different connected components of $G - N[x]$ too, we are done; any path between a and c dominates x and y.

Consider two arbitrary vertices a, b of D and a shortest path $P(a,b) := (a = x_0, x_1, \ldots, x_l = b)$ connecting a and b in G. We can build two spanning trees $T_1(a,b)$ and $T_2(a,b)$ for G, both containing the edges of $P(a,b)$, in such a way that for any two vertices $x, y \in N[P(a,b)]$, $d_{T_1(a,b)}(x,y) \leq d_G(x,y)+4$ and $\min\{d_{T_1(a,b)}(x,y), d_{T_2(a,b)}(x,y)\} \leq d_G(x,y)+3$.

Our trees $T_1(a,b)$ and $T_2(a,b)$ are very similar to the trees constructed for AT-free graphs. The tree $T_1(a,b) = (V, E_1)$ is constructed as follows. Add to initially empty set E_1 all edges of path $P(a,b)$. Then, for each vertex $z \in N(P(a,b))$ choose an arbitrary neighbor w in $P(a,b)$ and add edge zw to E_1. The obtained subtree of G (which covers so far only vertices from $N[P(a,b)]$) extends to a spanning tree $T_1(a,b)$ arbitrarily. The tree $T_1(a,b)$ is an analog of the caterpillar-tree constructed for an AT-free graph. The second tree is an analog of the cactus-tree considered for an AT-free graph. The tree $T_2(a,b) = (V, E_2)$ is a special breadth-first-search-tree T_a with vertex a as the root and is constructed as follows.

set $E_2 := \{$all edges of the path $P := (a = x_0, x_1, \ldots, x_l = b)\}$;
set $dev(x_i) := 0$ for each vertex x_i of the path $P(a,b)$;
for $i = 1$ to $ecc(a)$ do
  if $i \leq l$ then set $A := N_i(a) \setminus \{x_i\}$ else set $A := N_i(a)$;
  for each vertex $z \in A$ do
    among all neighbors of z in $N_{i-1}(a)$ choose a neighbor w with minimum
    deviation $dev(w)$;
    add edge zw to E_2 and set $dev(z) := dev(w) + 1$;
  enddo
enddo.

Lemma 49 *For any two vertices $x, y \in N[P(a,b)]$ the following is true.*

1. $d_{T_1(a,b)}(x,y) \leq d_G(x,y) + 4$;

2. *there is a tree* $T' \in \{T_1(a,b), T_2(a,b)\}$ *such that* $d_{T'}(x,y) \leq d_G(x,y) + 3$.

Proof The proof is similar to the proof of claims 1 and 2 of Lemma 47 and therefore is omitted.

If we construct trees $T_1(a,b)$ and $T_2(a,b)$ for each pair of vertices $a, b \in D$, from Lemma 48 and Lemma 49, we get (recall that $|D| \leq \mathsf{an}(G)$):

Theorem 35 *Any graph* G *with asteroidal number* $\mathsf{an}(G)$ *admits a system of* $\mathsf{an}(G)(\mathsf{an}(G)-1)/2$ *collective additive tree 4-spanners and a system of* $\mathsf{an}(G)(\mathsf{an}(G)-1)$ *collective additive tree 3-spanners.*

Corollary 15 *Any graph* G *with asteroidal number bounded by a constant admits a system of a constant number of collective additive tree 3-spanners. Moreover, given a repulsive asteroidal set of* G*, all trees are constructable in total linear time.*

3.5.5 Routing labeling schemes in AT-free related graphs

In this section, we use the results obtained above to design compact and efficient routing labeling schemes for graphs from the AT-free hierarchy. We first describe a direct routing labeling scheme for all graphs admitting a system of μ collective additive tree r-spanners. This scheme uses the efficient routing labeling scheme developed in [63, 108] for arbitrary trees. Then we show that, using the special structure of the trees constructed in Section 3.5, better routing labeling schemes can be designed for AT-free and related families of graphs. Among other results we show that any AT-free graph with diameter D and maximum vertex degree Δ admits a $(3\log_2 D + 6\log_2 \Delta + 3)$-bit routing labeling scheme of deviation at most 2. Moreover, the scheme is computable in linear time, and the routing decision is made in constant time per vertex.

It is worth mentioning that any AT-free graph admits a $(\log_2 D + 1)$-bit distance labeling scheme of deviation at most 2 (see [66]). That is, there is a function L labeling the vertices of each AT-free graph G with (not necessarily distinct) labels of up to $\log_2 D + 1$ bits such that given two labels $L(v), L(u)$ of two vertices v, u of G, it is possible to compute in constant time, by merely inspecting the labels of u and v, a value $\widehat{d}(u, v)$ such that $0 \leq \widehat{d}(u, v) - d_G(u, v) \leq 2$. To the best of our knowledge, the method of [66] cannot be used (at least directly) to design a routing labeling scheme for AT-free graphs.

A direct routing labeling scheme for all graphs admitting a system of μ collective additive tree r-spanners

We will need the following two results on distance and routing labeling schemes for arbitrary trees.

Theorem 36 *[96] There is a function DL labeling in $O(n \log n)$ total time the vertices of an n-vertex tree T with labels of up to $O(\log^2 n)$ bits such that given two labels $DL(v), DL(u)$ of two vertices v, u of T, it is possible to compute in constant time the distance $d_T(v, u)$, by merely inspecting the labels of u and v.*

Theorem 37 *[63, 108] There is a function RL labeling in $O(n)$ total time the vertices of an n-vertex tree T with labels of up to $O(\log^2 n / \log \log n)$ bits such that given two labels $RL(v), RL(u)$ of two vertices v, u of T, it is possible to determine in constant time the port number, at u, of the first edge on the path in T from u to v, by merely inspecting the labels of u and v.*

Now consider a graph G admitting a system $\mathcal{T}(G) = \{T_1, T_2, \ldots, T_\mu\}$ of μ collective additive tree r-spanners. We can preprocess each tree T_i using the algorithms from [96] and [63] and assign to each vertex v of G a distance label $DL_i(v)$ of size $O(\log^2 n)$

bits and a routing label $RL_i(v)$ of size $O(\log^2 n/\log\log n)$ bits associated with the tree T_i. Then we can form a label $L(v)$ of v of size $O(\mu \log^2 n)$ bits by concatenating the μ tree-labels:

$$L(v) := DL_1(v) \circ \ldots \circ DL_\mu(v) \circ RL_1(v) \circ \ldots \circ RL_\mu(v).$$

Assume now that a vertex v wants to send a message to a vertex u. Given the labels $L(v)$ and $L(u)$, v first uses the substrings $DL_1(v)\circ\ldots\circ DL_\mu(v)$ and $DL_1(u)\circ\ldots\circ DL_\mu(u)$ to compute in $O(\mu)$ time the distances $d_{T_i}(v,u)$ $(i = 1,\ldots,\mu)$ and an index k such that $d_{T_k}(v,u) = \min\{d_{T'}(v,u) : T' \in \mathcal{T}(G)\}$. Then, v extracts from $L(u)$ the substring $RL_k(u)$ and forms a header of the message $H(u) := k \circ RL_k(u)$. Now, the initiated message with the header $H(u) = k \circ RL_k(u)$ is routed to the destination using the tree T_k: when the message arrives at an intermediate vertex x, vertex x using its own substring $RL_k(x)$ and the string $RL_k(u)$ from the header makes a constant time routing decision.

Thus, the following result is true.

Theorem 38 *Let G be an n-vertex graph admitting a system of μ collective additive tree r-spanners. Then G has a $O(\mu \log^2 n)$-bit routing labeling scheme of deviation r. Once computed by the sender in $O(\mu)$ time, headers never change, and the routing decision is made in constant time per vertex.*

A better routing labeling scheme for AT-free graphs

In subsection 3.5.1, we showed that any AT-free graph $G = (V,E)$ admits a system of two collective additive tree 2-spanners. During the construction of the cactus-tree T_2 for G, each vertex $z \in V$ received a deviation number $dev(z)$ that is the distance in T_2 between z and the stem $P := (x_0, x_1, \ldots, x_l)$ of T_2 using only "down" edges. To simplify the routing decision, it will be useful to construct one more spanning tree $T' = (V,E')$ for G. Let $P := (x_0, x_1, \ldots, x_l)$ be the dominating path of G described in Lemma 44.

set $E' := \{$all edges of the path $P := (x_0, x_1, \ldots, x_l)\}$;

set $dev'(x_i) := 0$ for each vertex x_i of the path P and $dev'(z) := l + 1$ for any $z \in V \setminus P$;

for each vertex $z \in N_l(x_0)$ that is adjacent to x_l, set $dev'(z) := 1$ and add edge zx_l to E';

for $i = l - 1$ down to 1 do

  for each vertex $z \in N_i(x_0) \setminus \{x_i\}$ do

    if z is adjacent to x_i in G then add edge zx_i to E' and set $dev'(z) := 1$;

    else if z has neighbors in $N_{i+1}(x_0)$ then

      among all neighbors of z in $N_{i+1}(x_0)$, choose a neighbor w with minimum deviation $dev'(w)$ (break ties arbitrarily);

      if $dev'(w) < l + 1$ then add edge zw to E' and set $dev'(z) := dev'(w) + 1$;

  enddo

enddo

for each vertex z with $dev'(z)$ still $l + 1$ do

  let $z \in N_i(x_0)$;

  add edge zx_{i-1} to E';

enddo.

We name tree T' the *willow-tree* of G. As a result of its construction, each vertex $z \in V$ got a second deviation number $dev'(z)$, which is either $l + 1$ or the distance in T' (using only "horizontal" or "up" edges) between z and the path $P := (x_0, x_1, \ldots, x_l)$ of T'.

Now we are ready to describe the routing labels of the vertices of G. For each vertex $x_i \in P$ $(i = 0, 1, \ldots, l)$, we have

$$Label(x_i) := (b(x_i), level(x_i), port_{up}(x_i), port_{down}(x_i)),$$

where

- $b(x_i) := 1$, a bit indicating that x_i belongs to P;
- $level(x_i)$ $(= i)$ is the index of x_i in P, i.e., the distance $d_G(x_i, x_0)$;
- $port_{up}(x_i)$ is the port number at vertex x_i of the edge $x_i x_{i+1}$ (if $i = l$, $port_{up}(x_i)$ $:= \mathsf{nil}$);
- $port_{down}(x_i)$ is the port number at vertex x_i of the edge $x_i x_{i-1}$ (if $i = 0$, $port_{down}(x_i)$ $:= \mathsf{nil}$).

For each vertex $z \in V \setminus P$, we have

$$Label(z) := (b(z), level(z), a_{v\downarrow}(z), port_{v-in}(z), port_{v-out}(z), a_h(z), port_{h-in}(z), port_{h-out}(z),$$

$$dev(z), port_{down}(z), dev'(z), port_{up}(z)),$$

where

- $b(z) := 0$, a bit indicating that z does not belong to P;
- $level(z)$ is the distance $d_G(z, x_0)$;
- $a_{v\downarrow}(z)$ is a bit indicating whether z is adjacent to $x_{level(z)-1}$;
- $port_{v-in}(z)$ is the port number at vertex $x_{level(z)-1}$ of the edge $x_{level(z)-1}z$ (if z and $x_{level(z)-1}$ are not adjacent in G, then $port_{v-in}(z) := \mathsf{nil}$);
- $port_{v-out}(z)$ is the port number at vertex z of the edge $z x_{level(z)-1}$ (if z and $x_{level(z)-1}$ are not adjacent in G, then $port_{v-out}(z) := \mathsf{nil}$);

- $a_h(z)$ is a bit indicating whether z is adjacent to $x_{level(z)}$;
- $port_{h-in}(z)$ is the port number at vertex $x_{level(z)}$ of the edge $x_{level(z)}z$ (if z and $x_{level(z)}$ are not adjacent in G, then $port_{h-in}(z) := \mathsf{nil}$);
- $port_{h-out}(z)$ is the port number at vertex z of the edge $zx_{level(z)}$ (if z and $x_{level(z)}$ are not adjacent in G, then $port_{h-out}(z) := \mathsf{nil}$);
- $dev(z)$ is the deviation of z in tree T_2;
- $port_{down}(z)$ is the port number at vertex z of the edge zw, where w is the father of z in T_2;
- $dev'(z)$ is the deviation of z in tree T';
- $port_{up}(z)$ is the port number at vertex z of the edge zw, where w is the father of z in T' (if $dev'(z) = l + 1$, $port_{up}(z) := \mathsf{nil}$).

Clearly, the label size of each vertex of G is at most $3\log_2 l + 6\log_2 \Delta + 3 \leq 3\log_2 D + 6\log_2 \Delta + 3$ bits.

The routing decision algorithm is obvious. Suppose that a packet with the header (address of destination) $Label(y)$ arrives at vertex x. Vertex x can use the following constant time algorithm to decide where to submit the packet. Note that each vertex v of G is uniquely identified by its label $Label(v)$.

function routing_decision_AT-free($Label(x), Label(y)$)

if $Label(x) = Label(y)$ then return "packet reached its destination";

else do case

case ($b(x) = 1$) /* x belongs to P and routing is performed on the

caterpillar-tree T_1 of G */

do case

case ($level(x) > level(y)$)

send packet via $port_{down}(x)$;

case ($level(x) < level(y)$)

if $b(y) = 1$ then send packet via $port_{up}(x)$;

else if $level(y) = level(x) + 1$ and $a_{v\downarrow}(y) = 1$ then send packet via $port_{v-in}(y)$;

else send packet via $port_{up}(x)$;

case ($level(x) = level(y)$)

if $a_h(y) = 1$ then send packet via $port_{h-in}(y)$;

else send packet via $port_{down}(x)$;

endcase;

/* now x does not belong to P */

case ($level(x) > level(y)$)

do case

case ($a_{v\downarrow}(x) = 1$)

send packet via $port_{v-out}(x)$; /* routing is performed on T_1 */

case ($b(y) = 1$ or $b(y) = 0$ and $a_h(y) = 1$)

send packet via $port_{h-out}(x)$; /* routing is performed on T_1 */

otherwise /* here we have $d_{T_1}(x, y) = level(x) - level(y) + 3$ */

if $dev(x) \leq level(x) - level(y) + 1$ then send packet via $port_{down}(x)$;

/* the cactus-tree T_2 of G is used for routing */

else send packet via $port_{h-out}(x)$; /* routing is performed on T_1 */

endcase;

case ($level(x) < level(y)$)

do case

case ($a_h(x) = 1$)

send packet via $port_{h-out}(x)$; /* routing is performed on T_1 */

case ($b(y) = 1$ or $b(y) = 0$ and $a_{v\downarrow}(y) = 1$)

send packet via $port_{v-out}(x)$; /* routing is performed on T_1 */

otherwise /* here we have $d_{T_1}(x, y) = level(y) - level(x) + 3$ */

if $dev'(x) \leq level(y) - level(x) + 1$ then send packet via $port_{up}(x)$;

/* the willow-tree T' of G is used for routing */

else send packet via $port_{v-out}(x)$; /* routing is performed on T_1 */

endcase;

case ($level(x) = level(y)$) /* routing is performed on T_1 */

if $a_h(x) = 1$ then send packet via $port_{h-out}(x)$;

else send packet via $port_{v-out}(x)$;

endcase.

Thus, we have the following result.

Theorem 39 *Every AT-free graph of diameter $D := diam(G)$ and of maximum vertex degree Δ admits a $(3\log_2 D + 6\log_2 \Delta + 3)$-bit routing labeling scheme of deviation at most 2 (and also a $(\log_2 D + 2\log_2 \Delta + 2)$-bit routing labeling scheme of deviation at most 3). Moreover, the scheme is computable in linear time, and the routing decision is made in constant time per vertex.*

A $(\log_2 D + 2\log_2 \Delta + 2)$-bit routing labeling scheme of deviation at most 3 can easily be derived from the structure of tree T_1.

Permutation graphs.

The above idea can be used to design a compact routing labeling scheme for permutation graphs. Note that, by Theorem 31, any permutation graph admits an additive

tree 2-spanner. According to the tree construction algorithm, the tree has a spine (shortest path) $(x_0, x_1, \cdots, x_l)$. For every vertex $y \in N_i(x_0)$, $0 < i \leq l+1$, we have $yx_{i-1} \in E(G) \cap E(T)$. Each vertex x_i on the spine needs only to memorize the port numbers to x_{i+1} and to x_{i-1}. Any other vertex y needs only to memorize the port numbers from x_{i-1} to y and from y to x_{i-1}. Moreover, each vertex w will memorize also one extra bit, indicating whether w is on the spine, and the distance $d_G(w, x_0)$ which is the index of the level $L_i = N_i(x_0)$ it belongs to. Other details are left to the reader. We only formulate the final result for the permutation graphs.

Theorem 40 *Every permutation graph of diameter D and maximum vertex degree Δ admits a $(\log_2 D + 2\log_2 \Delta + 1)$-bit routing labeling scheme of deviation at most 2. Moreover, the scheme is computable in linear time, and the routing decision is made in constant time per vertex.*

DSP-graphs.

Based on Lemma 47, the following result can be proved for DSP-graphs.

Theorem 41 *Every n-vertex DSP-graph of diameter D and of maximum vertex degree Δ admits the following routing labeling schemes.*

Scheme	*Label Size*	*Decision time*	*Deviation*
1	$(\log_2 D + 2\log_2 \Delta + 3)$	$O(1)$	*4*
2	$(3\log_2 D + 8\log_2 \Delta + 4)$	$O(1)$	*3*
3	$O(\log^2 n / \log\log n)$	$O(1)$	*2*

The first routing labeling scheme can easily be constructed using the special structure of additive tree 4-spanner T_1 (see Section 3.5.3). The second routing labeling scheme can be constructed using the special structure of the system of two collective additive tree 3-spanners $\{T_1, T_2\}$. The construction is very similar to that used for AT-free

graphs (see the scheme of deviation 2 in Section 3.5.5), and its details are left to the reader (consider T_3 instead of T'; since in DSP-graphs one needs to handle also the case when vertex z from $N_i(x_0) \setminus P$ is adjacent to x_{i+1}, $i \in \{1, \ldots, l-1\}$, where we have a $(2\log_2 \Delta + 1)$-bit surplus in the label size). Here we will give a brief description only of the third routing labeling scheme (of deviation 2).

In Section 3.5.3, we have shown how to construct a system of five collective additive tree 2-spanners for a DSP-graph G with a dominating shortest path $P := (x_0, x_1, \ldots, x_l)$. From Theorem 38, one can conclude that G has a $O(\log^2 n)$-bit routing labeling scheme with deviation 2 and constant routing decision. In what follows, we will show that, to find distances in those trees, shorter labels are sufficient.

We preprocess each tree T_i $(i = 1, 2, 3, 4, 5)$ using the algorithm from [**?**] and assign to each vertex v of G a routing label $RL_i(v)$ of size $O(\log^2 n/\log\log n)$ bits associated with the tree T_i. Then we form a label $L(v)$ of v of size $6\log_2 D + O(\log^2 n/\log\log n) = O(\log^2 n/\log\log n)$ bits as follows:

$$L(v) := level(v) \circ level'(v) \circ dev_2(v) \circ dev_3(v) \circ dev_4(v) \circ dev_5(v) \circ RL_1(v) \circ \ldots \circ RL_5(v),$$

where $level(v) := d_G(x_0, v)$, $level'(v) := d_G(x_l, v)$ and $dev_2(v), dev_3(v), dev_4(v), dev_5(v)$ are the deviation numbers of v in T_2, T_3, T_4 and T_5, respectively.

Assume now that a vertex x wants to send a message to a vertex y. Given the labels $L(x)$ and $L(y)$, x first uses the substrings $(level(x) \circ level'(x) \circ dev_2(x) \circ dev_3(x) \circ dev_4(x) \circ dev_5(x))$ and $(level(y) \circ level'(y) \circ dev_2(y) \circ dev_3(y) \circ dev_4(y) \circ dev_5(y))$ and the function create_header_DSP (described below) to form in $O(1)$ time the header $H(y) := k \circ RL_k(y)$ of the message by finding an index $k \in \{1, 2, 3, 4, 5\}$ such that $d_{T_k}(x, y) = \min\{d_{T'}(x, y) : T' \in \{T_1, \ldots, T_5\}\}$ and extracting from $L(y)$ the substring $RL_k(y)$. Then, the initiated message with the header $H(y) = k \circ RL_k(y)$ can be routed to the destination using the tree T_k.

function create_header_DSP($L(x), L(y)$)

if $level(x) = level(y)$ then

do case

case ($dev_2(x) = 1$ and $dev_2(y) = 1$), set $k := 2$;

case ($dev_3(x) = 1$ and $dev_3(y) = 1$), set $k := 3$;

case ($dev_4(x) \leq 2$ and $dev_2(y) = 1$), set $k := 4$;

case ($dev_5(x) \leq 2$ and $dev_3(y) = 1$), set $k := 5$;

otherwise, set $k := 1$;

endcase

else

if $level(x) > level(y)$ then set $v := x$ and $u := y$;

else set $v := y$ and $u := x$;

do case

case ($dev_2(u) = 1$ and $dev_2(v) \leq level(v) - level(u) + 1$), set $k := 2$;

case ($dev_3(v) = 1$ and $dev_3(u) \leq level'(u) - level'(v) + 1$), set $k := 3$;

case ($dev_4(u) = 1$ and $dev_4(v) \leq level(v) - level(u) + 1$), set $k := 4$;

case ($dev_5(v) = 1$ and $dev_5(u) \leq level(v) - level(u) + 1$), set $k := 5$;

otherwise, set $k := 1$;

endcase

set the header of the message to be $(k, RL_k(y))$.

The correctness of this procedure follows from the proof of Lemma 47.

Graphs with bounded asteroidal number.

Let $G = (V, E)$ be a graph and let $D \subseteq V$ be a repulsive asteroidal set of G. We have $|D| \leq \mathsf{an}(G)$. For each pair of vertices a_i, b_i ($i = 1, \ldots, \mu, \mu := |D|(|D|-1)/2$) from D we construct three trees $T_1^i := T_1(a_i, b_i)$, $T_2^i := T_2(a_i, b_i)$ and $T_3^i := T_2(b_i, a_i)$ starting

from a shortest path $P(a_i, b_i)$ as described in Section 3.5.4. With each vertex v of G we associate a characteristic vector $\chi(v) = (\chi_1(v), \chi_2(v), \ldots, \chi_\mu(v))$, where $\chi_i(v) = 1$ if v belongs to $N[P(a_i, b_i)]$ and 0 otherwise.

For each index i $(i = 1, \ldots, \mu)$, a vertex z of G will store in its label the following routing information. If z belongs to $P_i := P(a_i, b_i)$ then

$$L_i(z) := (b(z), level(z), port_{up}(z), port_{down}(z)),$$

where

- $b(z) := 1$, a bit indicating that z belongs to P_i;
- $level(z) := d_G(z, a_i)$;
- $port_{up}(z)$ is the port number at vertex z of the edge of P_i leading towards b_i (if $z = b_i$, $port_{up}(z) := \mathsf{nil}$);
- $port_{down}(z)$ is the port number at vertex z of the edge of P_i leading towards a_i (if $z = a_i$, $port_{down}(z) := \mathsf{nil}$).

If z does not belong to $P_i := P(a_i, b_i) := (a_i = x_0, x_1, \ldots, x_l = b_i)$ then

$$L_i(z) := (b(z), level(z), a_{v\downarrow}(z), port_{v\downarrow-in}(z), port_{v\downarrow-out}(z), a_h(z), port_{h-in}(z), port_{h-out}(z),$$

$$a_{v\uparrow}(z), port_{v\uparrow-in}(z), port_{v\uparrow-out}(z), dev(z), port_{down}(z), dev'(z), port_{up}(z)),$$

where

- $b(z) := 0$, a bit indicating that z does not belong to P_i;
- $level(z)$ is the distance $d_G(z, a_i)$;
- $a_{v\downarrow}(z)$ is a bit indicating whether z is adjacent to $x_{level(z)-1}$;

- $port_{v\downarrow -in}(z)$ is the port number at vertex $x_{level(z)-1}$ of the edge $x_{level(z)-1}z$ (if z and $x_{level(z)-1}$ are not adjacent in G, then $port_{v\downarrow -in}(z) := \mathsf{nil}$);
- $port_{v\downarrow -out}(z)$ is the port number at vertex z of the edge $zx_{level(z)-1}$ (if z and $x_{level(z)-1}$ are not adjacent in G, then $port_{v\downarrow -out}(z) := \mathsf{nil}$);
- $a_h(z)$ is a bit indicating whether z is adjacent to $x_{level(z)}$;
- $port_{h-in}(z)$ is the port number at vertex $x_{level(z)}$ of the edge $x_{level(z)}z$ (if z and $x_{level(z)}$ are not adjacent in G, then $port_{h-in}(z) := \mathsf{nil}$);
- $port_{h-out}(z)$ is the port number at vertex z of the edge $zx_{level(z)}$ (if z and $x_{level(z)}$ are not adjacent in G, then $port_{h-out}(z) := \mathsf{nil}$);
- $a_{v\uparrow}(z)$ is a bit indicating whether z is adjacent to $x_{level(z)+1}$;
- $port_{v\uparrow -in}(z)$ is the port number at vertex $x_{level(z)+1}$ of the edge $x_{level(z)+1}z$ (if z and $x_{level(z)+1}$ are not adjacent in G, then $port_{v\uparrow -in}(z) := \mathsf{nil}$);
- $port_{v\uparrow -out}(z)$ is the port number at vertex z of the edge $zx_{level(z)+1}$ (if z and $x_{level(z)+1}$ are not adjacent in G, then $port_{v\uparrow -out}(z) := \mathsf{nil}$);
- $dev(z)$ is the deviation of z in tree T_2^i;
- $port_{down}(z)$ is the port number at vertex z of the edge zw, where w is the father of z in T_2^i;
- $dev'(z)$ is the deviation of z in tree T_3^i;
- $port_{up}(z)$ is the port number at vertex z of the edge zw, where w is the father of z in T_3^i.

Now we define label $Label(z)$ of a vertex z of G as

$$Label(z) := (\chi(z), L_1(z), L_2(z), \ldots, L_\mu(z)).$$

Clearly, the label size of any vertex of G is at most $\mu + \mu(3\log_2 D + 8\log_2 \Delta + 4) \leq (\mathsf{an}(G)(\mathsf{an}(G)-1)/2)(3\log_2 D + 8\log_2 \Delta + 5)$ bits.

Assume now that a vertex v wants to send a message to a vertex u. Given the labels $Label(v)$ and $Label(u)$, v first uses the characteristic vectors $\chi(v)$ and $\chi(u)$ to find an index k such that $\chi_k(v) = \chi_k(u) = 1$. Then, v extracts from $Label(u)$ the substring $L_k(u)$ and forms a header of the message $H(u) := k \circ L_k(u)$. Now, the initiated message with the header $H(u) = k \circ L_k(u)$ is routed to the destination using the trees T_1^k, T_2^k, T_3^k: when the message arrives at an intermediate vertex x, vertex x using own substring $L_k(x)$ and the string $L_k(u)$ from the header makes a constant time routing decision by using a function similar (even simpler since we are targeting now deviation 3 only, not 2) to function routing_decision_AT-free (details are omitted again). Thus, we have the following result.

Theorem 42 *Every n-vertex graph G with asteroidal number $\mathsf{an}(G)$, diameter D and maximum vertex degree Δ admits an $(\mathsf{an}(G)(\mathsf{an}(G)-1)/2)(3\log_2 D + 8\log_2 \Delta + 5)$-bit labeling scheme of deviation 3 (and an $(\mathsf{an}(G)(\mathsf{an}(G)-1)/2)(\log_2 D + 2\log_2 \Delta + 4)$-bit routing labeling scheme of deviation 4). Once computed by the sender in $(\mathsf{an}(G)(\mathsf{an}(G)-1)/2)$ time, headers never change. Moreover, given a repulsive asteroidal set of G the scheme is computable in linear time, and the routing decision is made in constant time per vertex.*

CHAPTER 4

DISTANCE APPROXIMATING TREES

4.1 Introduction

In tree and collective tree spanners, the underlying spanning sub-graphs must use the edges in the original graph. For many applications (e.g. in numerical taxonomy or in phylogeny reconstruction), this requirement can be dropped [12, 103, 106]. This motivated us to define a new notion - *distance approximating trees*.

In this chapter, continuing the line of research started in [19, 30] on distance (Δ, δ)-approximating trees, we will be interested in two special cases, when either $\Delta = 1$ or $\delta = 0$. A tree $T = (V, E')$ is a distance $(\Delta, 0)$-approximating tree of $G = (V, E)$ if $1/\Delta \cdot d_G(u, v) \leq d_T(u, v) \leq \Delta \cdot d_G(u, v)$ for all $u, v \in V$. A tree $T = (V, E')$ is a distance $(1, \delta)$-approximating tree of $G = (V, E)$ (or, simply, a distance δ-approximating tree of G) if $|d_G(u, v) - d_T(u, v)| \leq \delta$ for all $u, v \in V$. The distance (Δ, δ)-approximating tree problem asks for a given graph G to decide whether G has a distance (Δ, δ)-approximating tree.

In this chapter, we consider unweighted graphs and show that the distance $(\Delta, 0)$-approximating tree problem is NP-complete for any $\Delta \geq 5$ and the distance $(1, 1)-$approximating tree problem is polynomial time solvable. The latter solves (algorithmically) the problem posed in [30] which asked to characterize/recognize the graphs admitting distance (1, 1)-approximating trees.

Results of this chapter were published in [47].

4.1.1 Basic notions, notation and facts

All graphs $G = (V, E)$ occurring in this chapter are connected, finite, undirected, loopless and without multiple edges. The *length* of a path from a vertex u to a vertex v is the number of edges in this path. The *distance* $d_G(u, v)$ between the vertices u and v in G is the length of a shortest (u, v)-path. The *eccentricity* $ecc_G(v)$ of a vertex v is the maximum distance from v to any vertex in G. The *radius* $rad(G)$ of a graph G is the minimum eccentricity of a vertex in G and the *diameter* $diam(G)$ of G is the maximum eccentricity of a vertex in G.

For a subset $S \subseteq V$ of vertices of a graph G, by $G(S)$ we denote the subgraph of G induced by S. Let, for simplicity, $G - v := G(V \setminus \{v\})$ and $G - v - u := G(V \setminus \{v, u\})$, where v and u are vertices of G. Let also $G - uv$ denote the graph obtained from G by removing edge uv of G, i.e., $G - uv := (V, E \setminus \{uv\})$. A graph G is said to be *3-connected* if $G - u - v$ is connected for any pair of vertices $u, v \in V$. A graph G is said to be *2-connected* if $G - u$ is connected for any vertex $u \in V$. In a 2-connected graph G, if for some pair of vertices $x, y \in V$ the graph $G - x - y$ is disconected, then we say that $\{x, y\}$ is a *2-cut* of G. In a connected graph G, if for some vertex $x \in V$ the graph $G - x$ is disconected, then we say that x is a *1-cut vertex* (or, simply, *1-cut*) of G.

It is easy to see from the definitions of distance approximating trees that the following holds.

Lemma 50 *A tree $T = (V, F)$ is a distance $(\Delta, 0)$-approximating tree of a graph $G = (V, E)$ if and only if $d_T(x, y) \leq \Delta$ holds for each edge $xy \in E$ and $d_G(u, v) \leq \Delta$ holds for each edge $uv \in F$.*

Lemma 51 *If T is a distance $(0, \delta)$-approximating tree for G, then T is a distance $(\Delta, 0)-$ approximating tree for G with $\Delta = \delta + 1$.*

4.2 Distance δ-approximating trees with $\delta = 1$

In this section, we show that the distance δ-approximating tree problem is polynomial time solvable for $\delta = 1$. For simplicity, in what follows, we will use the notion "distance 1-approximating tree" as a synonym to "distance $(1, 1)$-approximating tree".

4.2.1 3-Connected graphs

A *star* is a tree with a vertex adjacent to all other vertices of the tree. We call that vertex *the center of the star*. Equivalently, a *star* is a tree of diameter at most 2.

Lemma 52 *For a 3-connected graph G, the following three statements are equivalent.*

1. *G has a distance 1-approximating tree.*
2. *G has a distance 1-approximating tree which is a star.*
3. *$diam(G) \leq 3$ and $rad(G) \leq 2$.*

Proof (1$\Longleftrightarrow$2) Let T be a distance 1-approximating tree of G. If T is not a star, then there exists a path in T with length 3. Let (x', x, y, y') be such a path. Consider subtrees T_x and T_y obtained from T by removing edge xy, and assume that x belongs to T_x and y belongs to T_y. Since for any $u \in V(T_x) \setminus \{x\}$ and $v \in V(T_y) \setminus \{y\}$, $d_T(u, v) \geq 3$, we have $uv \notin E(G)$. This implies that $\{x, y\}$ is a 2-cut of G, contradicting with the 3-connectedness of G. Hence, T must be a star.

(2$\Rightarrow$3) Let T be a distance 1-approximating tree of G which is a star. Then, for any $x, y \in V$, we have $d_T(x, y) \leq 2$ and, therefore, $d_G(x, y) \leq 3$. Hence, $diam(G) \leq 3$. Let now u be the center of T. Then, for each $x \in V$, $d_T(x, u) \leq 1$, and therefore $d_G(x, u) \leq 2$. The latter implies $rad(G) \leq 2$.

(3$\Rightarrow$2) If $rad(G) \leq 2$, then, by definition, there exists a vertex $u \in V$ such that $d_G(x, u) \leq 2$, for any $x \in V$. Pick such a vertex u and construct a tree $T = (V, E')$ where each vertex $v \in V \setminus \{u\}$ is adjacent to u, i.e., construct a star on vertices V

with the center u. Obviously, $0 \leq d_G(x,y) - d_T(x,y) \leq 1$, for any $x \in V \setminus \{u\}$. Moreover, since $diam(G) \leq 3$, we have $d_G(x,y) \leq 3$ for any $x, y \in V \setminus \{u\}$. As, for those vertices x and y, $d_T(x,y) = 2$, we conclude $d_G(x,y) - d_T(x,y) \leq 3 - 2 = 1$ and $d_G(x,y) - d_T(x,y) \geq 1 - 2 = -1$. Hence, T is a distance 1-approximating tree of G. □

The following corollary is immediate from this proof.

Corollary 16 *Let G be an arbitrary (not necessarily 3-connected) graph. Then, G has a distance 1-approximating tree which is a star if and only if $diam(G) \leq 3$ and $rad(G) \leq 2$.*

4.2.2 2-Connected graphs

A vertex of a tree is *inner* if it is not a leaf. An edge of a tree is an *inner edge* if it is not incident to a leaf.

Lemma 53 *If T is a distance 1-approximating tree of a connected graph G, then any inner edge of T is a 2-cut of G.*

Proof For any inner edge xy of T, let T_x and T_y be the two subtrees of T obtained from T by removing edge xy. Let also x belong to T_x and y belong to T_y. Then, since T is a distance 1-approximating tree of G, for all $u \in V(T_x) \setminus \{x\}$ and $v \in V(T_y) \setminus \{y\}$, $uv \notin E(G)$. This implies that $\{x, y\}$ is a 2-cut of G separating $V(T_x) \setminus \{x\}$ from $V(T_y) \setminus \{y\}$. □

A *bistar* is a tree with only one inner edge. Equivalently, a *bistar* is a tree of diameter 3. The proof of the following lemma can be found in the [47].

Lemma 54 *If T is a distance 1-approximating tree of a 2-connected graph G, then $diam(T) \leq 3$, i.e., T is a star or a bistar.*

To characterize 2-connected graphs admitting distance 1-approximating trees, we will need also the following easy observations (proof can be found in the [47]).

Lemma 55 *Assume a graph G has a distance 1-approximating bistar and let T be such a bistar for G with the inner edge c_1c_2. Then, the following properties hold:*

1. *$diam(G) \leq 4$ and $rad(G) \leq 3$;*
2. *for any $j = 1, 2$ and $x, y \in V(T_{c_j}) \cup \{c_1, c_2\}$, $d_G(x, y) \leq 3$ and $d_G(x, c_j) \leq 2$;*
3. *if $A_1, \ldots, A_k$ are the connected components of the graph $G - c_1 - c_2$ and T_{c_1}, T_{c_2} are the connected components of $T - c_1c_2$, then, for any $i = 1, \ldots, k$, $V(A_i)$ is entirely contained either in $V(T_{c_1})$ or in $V(T_{c_2})$.*

Let now G be a graph with a 2-cut $\{a, b\}$ and $A_1, \ldots, A_k$ be the connected components of the graph $G - a - b$. For given 2-cut $\{a, b\}$ of G we can construct a new graph $H_{a,b}$ as follows. The vertex set of $H_{a,b}$ is $\{a, b, a_1, \ldots, a_k\}$. Edge aa_i $(i = 1, \ldots, k)$ exists in $H_{a,b}$ if and only if for each $x, y \in V(A_i) \cup \{b\}$, $d_G(x, y) \leq 3$ and $d_G(x, a) \leq 2$ hold. Edge ba_i $(i = 1, \ldots, k)$ exists in $H_{a,b}$ if and only if for each $x, y \in V(A_i) \cup \{a\}$, $d_G(x, y) \leq 3$ and $d_G(x, b) \leq 2$ hold. Edge a_ia_j $(i, j = 1, \ldots, k,\ i \neq j)$ exists in $H_{a,b}$ if and only if for each vertex $x \in V(A_i)$ and each vertex $y \in V(A_j)$, $d_G(x, y) \leq 3$ holds. No other edges exist in $H_{a,b}$.

The following lemma gives a characterization of those 2-connected graphs that admit distance 1-approximating trees. Denote the complement of a graph H by $\overline{H}$.

Lemma 56 *For a 2-connected graph G, the following three statements are equivalent.*

1. *G has a distance 1-approximating tree.*
2. *G has a distance 1-approximating tree which is a star or a bistar.*
3. *$diam(G) \leq 3$ and $rad(G) \leq 2$ or $diam(G) \leq 4$ and there exists a 2-cut $\{a, b\}$ in G such that the graph $\overline{H_{a,b}}$ is bipartite.*

Proof (1$\Longleftrightarrow$2) is given by Lemma 54.

(2$\Rightarrow$3) If G has a distance 1-approximating tree which is a star, then, by Corollary 16, $diam(G) \leq 3$ and $rad(G) \leq 2$. Assume now that a distance 1-approximating tree T of G is a bistar. Then, by Lemma 55, $diam(G) \leq 4$. Lemma 55 (together with Lemma 53) implies also that G has a 2-cut $\{a, b\}$ (which is the inner edge of T) such that for any connected component A_i ($i \in \{1, \ldots, k\}$) of $G - a - b$, either $V(A_i) \subset V(T_a)$ or $V(A_i) \subset V(T_b)$ holds. Since vertices $V(T_a) \cup \{b\}$ form a star in T with the center a, we have $d_G(x, y) \leq 3$ and $d_G(x, a) \leq 2$ for any $x, y \in V(T_a) \cup \{b\}$. By construction of $H_{a,b}$, vertices $\{a\} \cup \{a_i : V(A_i) \subset V(T_a)\}$ of $H_{a,b}$ will form a clique. Analogously, vertices $\{b\} \cup \{a_i : V(A_i) \subset V(T_b)\}$ form a clique in $H_{a,b}$. Since these two cliques cover all vertices of $H_{a,b}$, the complement $\overline{H_{a,b}}$ of $H_{a,b}$ is bipartite.

(3$\Rightarrow$2) Clearly, if $diam(G) \leq 3$ and $rad(G) \leq 2$ then, by Corollary 16, G has a distance 1-approximating star. Assume now that $diam(G) \leq 4$ and there exists a 2-cut $\{a, b\}$ in G such that the graph $\overline{H_{a,b}}$ is bipartite. Let $A_1, \ldots, A_k$ be the connected components of the graph $G-a-b$. Vertices of $H_{a,b}$ can be partitioned into two cliques C_1 and C_2. Since a and b are not adjacent in $H_{a,b}$, they must be in different cliques. Assume, $a \in C_1$ and $b \in C_2$. By construction of $H_{a,b}$, for all $x, y \in \cup\{V(A_i) : a_i \in C_1\} \cup \{b\}$, $d_G(x, y) \leq 3$ and $d_G(x, a) \leq 2$ holds. Similarly, for all $x, y \in \cup\{V(A_i) : a_i \in C_2\} \cup \{a\}$, $d_G(x, y) \leq 3$ and $d_G(x, b) \leq 2$ holds. Hence, we can construct a bistar T of G as follows. Vertices a and b will form the inner edge of T. Vertices of A_i with $a_i \in C_1$ will be attached (i.e., made adjacent in T) to a. Vertices of A_i with $a_i \in C_2$ will be attached to b. It is easy to see that T is a distance 1-approximating tree of G. The only interesting case to mention here is when $x \in V(A_i)$, where $a_i \in C_1$, and $y \in V(A_j)$, where $a_j \in C_2$. For those x and y, we have $d_T(x, y) = 3$ and $2 \leq d_G(x, y) \leq 4$ (since $diam(G) \leq 4$ and x and y are separated by $\{a, b\}$ in G). Thus, $-1 \leq c_T(x, y) \leq 1$ holds. □

Conditions $diam(G) \leq 4$ and $rad(G) \leq 3$ are not sufficient for a 2-connected graph G to have a distance 1-approximating tree.

The following corollary is immediate from the proof of Lemma 56.

Corollary 17 *Let G be an arbitrary (not necessarily 2-connected) graph. Then, G has a distance 1-approximating tree which is a star or a bistar if and only if $diam(G) \leq 3$ and $rad(G) \leq 2$ or $diam(G) \leq 4$ and there exists a 2-cut $\{a, b\}$ in G such that the graph $\overline{H_{a,b}}$ is bipartite.*

Lemma 56 implies also that the problem of checking whether a given 2-connected graph G has a distance 1-approximating tree is polynomial time solvable. More specifically, we have

Corollary 18 *It is possible, for a given 2-connected graph $G = (V, E)$, to check in $O(|V|^4)$ time whether G has a distance 1-approximating tree and, if such a tree exists, construct one within the same time bound.*

Proof We can find in $O(|V||E|)$ time the distance matrix of G and all 2-cuts [**?**, **?**] of G. Then, to check whether $diam(G) \leq 3$ and $rad(G) \leq 2$ and, if so, to construct a distance 1-approximating star of G as described in the proof of Lemma 52, one needs at most $O(|V|^2)$ time in total. To check if $diam(G) \leq 4$ and whether there exists a 2-cut $\{a, b\}$ of G with $\overline{H_{a,b}}$ bipartite, one needs $O(|V|^4)$ total time. We just need, for each 2-cut $\{a, b\}$, to construct the graph $\overline{H_{a,b}}$ and check if it is bipartite. Construction of $\overline{H_{a,b}}$ for a given 2-cut $\{a, b\}$ and checking whether it is bipartite will take no more than $O((|V|^2)$ time (given the distance matrix of G). Since any graph G has at most $O(|V|^2)$ 2-cuts, to check if G has a distance 1-approximating bistar, one needs at most $O(|V|^4)$ time. If G admits such a bistar, then we can find one in linear time as described in the proof of Lemma 56. □

4.2.3 Connected graphs

In this subsection, we assume that G is a connected graph but not 2-connected. Therefore, there exists a vertex $v \in V(G)$, such that $G-v$ contains at least two connected components.

From Lemma 54 and its proof, the following lemma is obvious.

Lemma 57 *Let T be a distance 1-approximating tree of a connected graph G and (a, b, c) be a path in T. If both a and c are inner vertices of T, then at least one of these vertices is a 1-cut of G. Moreover, assuming c is a 1-cut, c separates vertices $V(T_c) \setminus \{c\}$ from other vertices of G, where T_c is the subtree of $T - bc$ containing c.*

A *2-connected component* of a graph G is a maximal by inclusion 2-connected subgraph of G or an edge uv of G such that both u and v are 1-cuts of G (such an edge is called a *bridge* of G). Two 2-connected components of G are neighbors if they share a common vertex (a 1-cut) of G.

Lemma 58 *Assume a connected graph G admits a distance 1-approximating tree T and let A be a 2-connected component of G. Then, for any two vertices $x, y \in V(A)$, $d_T(x, y) \leq 3$. Moreover, if there exist vertices $x, y \in V(A)$, such that $d_T(x, y) = 3$, then $T(V(A))$ is a bistar.*

Proof Assume that, for some vertices $x, y \in V(A)$, $d_T(x, y) \geq 4$ holds. Then, one can connect x and y in T with a path $P_T(x, y)$ of length at least 4. Pick three consecutive inner vertices a, b, c of path $P_T(x, y)$, they necessarily exist. According to Lemma 57, a or c is a 1-cut of G separating x from y in G. The latter is in contradiction with the assumption that $x, y \in V(A)$ and A is a 2-connected component of G. Hence, $d_T(x, y) \leq 3$, for any $x, y \in V(A)$, is proven.

Assume now that there exist vertices $x, y \in V(A)$, such that $d_T(x, y) = 3$. Then, one can find two vertices $\{c_1, c_2\}$ in G such that $T(V(A) \cup \{c_1, c_2\})$ is a bistar with the inner edge c_1c_2. Let $xc_1, yc_2 \in E(T)$. We will show that both c_1 and c_2 are in A.

Suppose, neither c_1 nor c_2 is in A. Assume $c_1 \in V(B)$, $c_2 \in V(C)$, where B and C are 2-connected components of G. Let $V(B) \cap V(A) = \{v\}$ and $V(C) \cap V(A) = \{u\}$. We claim that $B = C$ or at least $v = u$. Suppose $B \neq C$ and $v \neq u$. Then, since $V(B) \cap V(C) = \emptyset$ (otherwise, A, B and C will be parts of one 2-connected component of G), $d_G(c_1, c_2) \geq 3$. As $d_T(c_1, c_2) = 1$, a contradiction with T being a distance 1-approximating tree of G arises. So, c_1, c_2 must be either in one 2-connected component of G or in two 2-connected components B and C such that $V(B) \cap V(A) = V(C) \cap V(A)$.

Without loss of generality, assume v is attached (i.e., adjacent in T) to c_1. Since $d_T(y, c_2) = 1$, we have $d_G(y, c_2) \leq 2$ and, hence, $yv \in E(G)$. On the other hand, $d_T(y, v) = 3$, contradicting the assumption that T is a distance 1-approximating tree of G.

Assume now that $c_1 \in V(A)$ and $c_2 \in V(B) \setminus \{v\}$. For any vertex $x' \in V(A)$ which is attached to c_1 and any vertex $y' \in V(A) \setminus \{c_1\}$ which is attached to c_2, $x'y' \notin E(G)$ must hold. Moreover, since $V(A) \cap V(B) = \{v\}$, one concludes that for all $x' \in V(A) \setminus \{v\}$, $x'c_2 \notin E(G)$. Hence, any path of A connecting a vertex attached to c_1 with a vertex attached to c_2 must use vertex c_1. Since there exist vertices $x, y \in V(A)$ such that $xc_1, yc_2 \in E(T)$, this is in contradiction with the assumption that A is 2-connected.

Thus, we conclude that $T(V(A))$ is a bistar. □

Corollary 19 *Assume a connected graph G admits a distance 1-approximating tree T and let A be a 2-connected component of G. Then, either $T(V(A))$ is a bistar or $T(V(A) \cup \{c\})$ is a star centered at some vertex c of G.*

In what follows, we will show that among all possible distance 1-approximating trees

of G there is a tree T such that, for any 2-connected component A of G, $T(V(A))$ is connected, i.e., if $T(V(A) \cup \{c\})$ is a star for some vertex c of G, then c must be in A. To show that, we will need two lemmata (proofs can be found in the [47]).

A sequence $(B_0 := B, B_1, \ldots, B_{k-1}, B_k := A)$ is called *the chain of 2-connected components of G between A and B* if each B_i is a 2-connected component of G, B_i and B_j are different for $j \neq i$, B_{i-1}, B_i are neighbors sharing a 1-cut $v_i := V(B_{i-1}) \cap V(B_i)$ of G for any $i \in \{1, \ldots, k\}$, and $v_i \neq v_j$ for any $i \neq j$. Clearly, this chain is unique for any A and B.

Lemma 59 *Assume a connected graph G admits a distance 1-approximating tree T, A and B be 2-connected components of G and $(B_0 := B, B_1, \ldots, B_{k-1}, B_k := A, Z)$ be the chain of 2-connected components of G between Z and B. If $T(V(A) \cup \{c\})$ is a star with the center c belonging to $V(Z) \setminus V(A)$, then for any $i \in \{0, \ldots, k-1\}$, $T(V(B_i))$ is a star centered at a 1-cut $v_{i+1} := V(B_{i+1}) \cap V(B_i)$ of G. Moreover, for any $i \in \{0, \ldots, k-1\}$ and any $x \in V(B_i)$, $xv_{i+1} \in E(G)$ must hold.*

Lemma 60 *Assume a connected graph G admits a distance 1-approximating tree T and let A, Z be 2-connected components of G such that $V(A) \cap V(Z) = \{v\}$. Let also A' be that connected component of the graph $G - v$ which contains $A - v$. If $T(V(A) \cup \{c\})$ is a star centered at $c \in V(Z) \setminus \{v\}$, then for any vertices $x \in V(A')$, $y \in (V(G) \setminus V(A')) \setminus \{c, v\}$, $xy \notin E(T)$ holds. In particular, for any two vertices $y, z \in V(G) \setminus V(A')$, the path $P_T(x, y)$ between x and y in T does not contain any vertices of A'.*

In what follows, let G be a connected graph admitting a distance 1-approximating tree and let T denote a distance 1-approximating tree of G with minimum $|E(T) \setminus E(G)|$, i.e., with minimum number of non-graph edges. We will show that this tree T has a number of nice properties.

Theorem 43 *If T is a distance 1-approximating tree of G with minimum $|E(T)\setminus E(G)|$, then $T(V(A))$ is a star or a bistar for any 2-connected component A of G.*

Proof Since A is a 2-connected component of G, by Corollary 19, either $T(V(A))$ is a bistar or $T(V(A)\cup\{c\})$ is a star centered at some vertex c of G. By way of contradiction, assume that for A, $T(V(A)\cup\{c\})$ is a star centered at a vertex c of G not belonging to A. Let c belong to some 2-connected component Z of G. Necessarily, A and Z are neighbor (2-connected) components. Let $v := V(A) \cap V(Z)$ and A' be a connected component of $G - v$ containing $V(A) \setminus \{v\}$. By Lemma 59, for any 2-connected component B of G, which is different from A and belongs to A', $T(V(B))$ is a star centered at a 1-cut of G lying in B and closest to A. Moreover, if v' is that 1-cut, then for any $x \in V(B)$, $xv' \in E(G)$ holds (see Figure 4.1). We have also that v is adjacent in G to c and to any vertex a ($a \neq v$) of A (see Lemma 59).

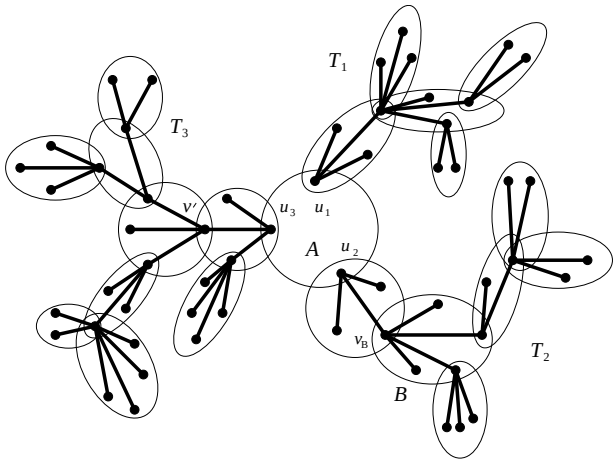


Figure 4.1: Illustration to the proof of Theorem 43. A part of the tree T is shown using thick edges. Thin edges show some graph edges.

We can transform tree T into a new tree T' as follows. Set $E(T') := E(T)$ and $V(T') := V(T)$. For each vertex $a \in V(A) \setminus \{v\}$, let $E(T') := (E(T') \setminus \{ac\}) \cup \{av\}$ (i.e., replace edge ac with edge av). We claim that T' is a distance 1-approximating tree

of G, too. We need to show that $|d_{T'}(x,y) - d_G(x,y)| \leq 1$ holds for any two vertices $x, y \in V(G)$.

If $x, y \in V(A')$ then, by Lemma 59 and the way we transformed T into T', $d_{T'}(x,y) = d_T(x,y)$. If $x, y \in V(G) \setminus V(A')$ then, by Lemma 60 and the way T was transformed into T', $d_{T'}(x,y) = d_T(x,y)$. Hence, in these cases, $|d_{T'}(x,y) - d_G(x,y)| = |d_T(x,y) - d_G(x,y)| \leq 1$.

Consider now the case when $x \in V(A')$ and $y \in V(G) \setminus V(A')$. By Lemma 59, $d_{T'}(x,v) = d_G(x,v)$. Since v is a 1-cut of G, $d_G(x,y) = d_G(x,v) + d_G(v,y)$. By Lemma 60 and the way we transformed T into T', one concludes that $d_{T'}(x,y) = d_{T'}(x,v) + d_{T'}(v,y)$. Combining these equalities, we get $|d_{T'}(x,y) - d_G(x,y)| = |d_{T'}(x,v) + d_{T'}(v,y) - (d_G(x,v) + d_G(v,y))| = |d_{T'}(v,y) - d_G(v,y)|$. But, by Lemma 60, $d_{T'}(v,y) = d_T(v,y)$. Hence, we get $|d_{T'}(x,y) - d_G(x,y)| = |d_T(v,y) - d_G(v,y)| \leq 1$.

Thus, T' is a distance 1-approximating tree of G. Since this tree has original graph edges more than T has ($|E(T') \setminus E(G)| < |E(T) \setminus E(G)|$), a contradiction with the choice of T arises. Hence, the center c of star $T(V(A) \cup \{c\})$ must belong to A. □

Lemma 61 *Let T be a distance 1-approximating tree of G with minimum $|E(T) \setminus E(G)|$ and A be a 2-connected component of G such that $T(V(A))$ is a bistar. Then, for any other 2-connected component B of G, $T(V(B))$ is a star centered at a 1-cut of G which is closest to A (among all 1-cuts of G located in B).*

Corollary 20 *If T is a distance 1-approximating tree of G with minimum $|E(T) \setminus E(G)|$, then there is at most one 2-connected component A in G such that $T(V(A))$ is a bistar.*

The following lemma (its proof can be found in [47]) and its corollaries show that a distance 1-approximating tree T of G with $T(V(A))$ being a star for any 2-connected component A of G has also a very deterministic structure.

Lemma 62 *Let T be a distance 1-approximating tree of G with minimum $|E(T) \setminus E(G)|$ and A and B be two neighbor 2-connected components of G with $v := V(A) \cap V(B)$. If $T(V(A))$ is a star centered not at v, then $T(V(B))$ is a star centered at v.*

Corollary 21 *Let T be a distance 1-approximating tree of G with minimum $|E(T) \setminus E(G)|$ and A be a 2-connected component of G such that $T(V(A))$ is a star. If the center of this star $T(V(A))$ is not a 1-cut of G, then for any other 2-connected component B of G, $T(V(B))$ is a star centered at a 1-cut of G which is closest to A (among all 1-cuts of G located in B).*

Corollary 22 *Let T be a distance 1-approximating tree of G with minimum $|E(T) \setminus E(G)|$. If for every 2-connected component A of G, $T(V(A))$ is a star centered at a 1-cut of G, then there exists a 1-cut v in G such that*

a) for any 2-connected component A of G containing v, $T(V(A))$ is a star centered at v,

b) for any 2-connected component B of G not containing v, $T(V(B))$ is a star centered at a 1-cut of G which is closest to v (among all 1-cuts of G located in B).

Clearly, if $T(V(A))$ is a star for a 2-connected component A of G, then $diam(A) \leq 3$ and $rad(A) \leq 2$. And, if $T(V(B))$ is a bistar for a 2-connected component B of G, then $diam(B) \leq 4$ and $rad(B) \leq 3$.

4.2.4 Algorithm for connected graphs

Recall that, for a 2-connected graph $G = (V, E)$, one can use the $O(|V|^4)$ time algorithm, described in the proof of Corollary 18, to check whether G has a distance 1-approximating tree and, if such a tree exists, construct one within the same time bound. Hence, we may assume here that graph $G = (V, E)$ is connected but not 2-connected.

From Lemma 61, Corollary 21 and Corollary 22 one can design the following algorithm for constructing a distance 1-approximating tree of a connected graph $G = (V, E)$, if G has one.

Algorithm Dist-1-appr-tree.

compute the distance matrix of G;
find all 1-cuts, 2-cuts and 2-connected components of G;
for each 2-connected component A of G, check its radius and diameter;
if $diam(A) > 4$ or $rad(A) > 3$ for some A,
  then return "G does not have a distance 1-approximating tree".
if G has two or more 2-connected components with diameter four or radius three,
  then return "G does not have a distance 1-approximating tree".
if G has a 2-connected component with diameter four or radius three,
  then call function Tree-with-a-bistar.
else do
/* PHASE I: attempt to grow a tree from a non–1-cut of G */
  for each 2-connected component A of G do
    search A for a vertex $u \in A$ such that u is not a 1-cut of G and $ecc_A(u) \leq 2$;
    for each such vertex u found do
      set $E' := \{ux : x \in V(A) \setminus \{u\}\}$;
      for each 2-connected component B of G not containing u
        set $E' := E' \cup \{vx : x \in V(B) \setminus \{v\}\}$,where v is a 1-cut of G from B
        which is closest to A (among all 1-cuts of G located in B);
      check whether tree $T_u := (V, E')$ is a distance 1-approximating tree of G;
      if "yes", then output the tree T_u and stop.
/* PHASE II: attempt to grow a tree from a 1-cut of G */
  for each 1-cut u of G do
    set $E' := \emptyset$;
    for each 2-connected component A of G containing u

set $E' := E' \cup \{ux : x \in V(A) \setminus \{u\}\}$;

for each 2-connected component B of G not containing u

set $E' := E' \cup \{vx : x \in V(B) \setminus \{v\}\}$,

where v is a 1-cut of G from B which is closest to u

(among all 1-cuts of G located in B);

check whether tree $T_u := (V, E')$ is a distance 1-approximating tree of G;

if "yes", then output the tree T_u and stop.

/* PHASE III: attempt to grow a tree from a 2-cut of G */

call function Tree-with-a-bistar.

If G has a distance 1-approximating tree T such that $T(V(A))$ is a star for every 2-connected component A of G, then, by Corollary 21 and Corollary 22, such a tree T will be found in PHASE I or in PHASE II of the above algorithm. If neither PHASE I nor PHASE II did produce a distance 1-approximating tree of G, then one can hope only for a distance 1-approximating tree T of G such that $T(V(A))$ is a bistar for some (and only one) 2-connected component A of G (see Lemma 61). The existence of such a tree is checked in the function Tree-with-a-bistar.

Function Tree-with-a-bistar iterates over all 2-connected components of G having 2-cuts. We know, by Lemma 61, that if $T(V(A))$ is a bistar, where A is a 2-connected component of G and T is a distance 1-approximating tree of G with minimum $|E(T) \setminus E(G)|$, then, for any other 2-connected component B of G, $T(V(B))$ is a star centered at a 1-cut of G from B which is closest to A. Moreover, by Lemma 53, the inner edge c_1c_2 of $T(V(A))$ is a 2-cut of A (and of G).

Thus, given a 2-connected component A of G with a 2-cut, we need, for every other 2-connected component B of G, to check if the 1-cut v_B of G from B closest to A has eccentricity $ecc_B(v_B) \leq 2$. If for some B, $ecc_B(v_B) > 2$ holds, then we cancel

the consideration of this 2-connected component A and move to another 2-connected components A of G with a 2-cut. Otherwise (i.e., if for all B, $ecc_B(v_B) \leq 2$), we do the following. Create a star $S_B := (V(B), \{bv_B : b \in V(B), b \neq v_B\})$ for each 2-connected component B of G different from A (v_B is a 1-cut of G from B closest to A) and consider the forest $F := (V, E_F)$, where $E_F := \bigcup_B E(S_B)$. We hope to extend this forest F to a distance 1-approximating tree T of G by defining a suitable bistar $T(V(A))$ on A.

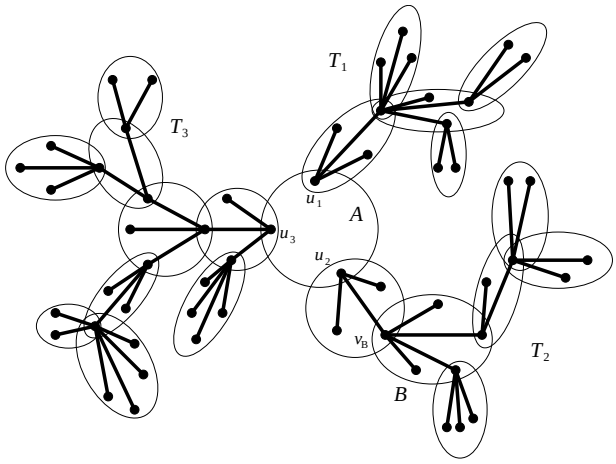


Figure 4.2: A 2-connected component A of G and the corresponding trees T_1, T_2, T_3.

Let $u_1, \ldots, u_k$ be all 1-cuts of G located in A and $T_1, \ldots, T_k$ be the trees from F containing these 1-cuts, i.e., $u_i \in V(T_i)$, $i = 1, \ldots, k$ (see Figure 4.2 for an illustration). We check for each i whether tree T_i is a distance 1-approximating tree of $G(V(T_i))$, where $G(V(T_i))$ is the subgraph of G induced by vertices $V(T_i)$. If not, then we terminate the consideration of this 2-connected component A. If yes, then for each 1-cut u_i, $i \in \{1, \ldots, k\}$, we compute the *tail* of u_i as follows: $tail(u_i) = \min\{d_{T_i}(x, u_i) - d_G(x, u_i) : x \in V(T_i)\}$. Clearly, as T_i is a distance 1-approximating tree of $G(V(T_i))$ and $d_{T_i}(x, u_i) \leq d_G(x, u_i)$ (by construction of T_is), we have $tail(u_i) \in \{-1, 0\}$.

Now, to get the desired tree T from F, we need to construct a bistar $T(V(A))$ with the inner edge c_1c_2, where $\{c_1, c_2\}$ is a 2-cut of A (i.e., we need to decide, for a given

2-cut $\{c_1, c_2\}$ of A, which edge, vc_1 or vc_2, will go to T for each vertex v of $A \setminus \{c_1, c_2\}$). For this, we will make some modifications to the method described in Subsection 4.2.2 so that the requirements dictated by tails are taken into account (the constructed subtrees T_i, $i = 1, \ldots, k$, have their influence to the construction of a bistar $T(V(A))$).

Notice that, for all $x \in V(T_i)$ and $y \in V(T_j)$, $d_G(x, y) = d_G(x, u_i) + d_A(u_i, u_j) + d_G(u_j, y)$ and $d_T(x, y) = d_{T_i}(x, u_i) + d_{T(V(A))}(u_i, u_j) + d_{T_j}(u_j, y)$. That is, $d_T(x, y) - d_G(x, y) = d_{T(V(A))}(u_i, u_j) - d_A(u_i, u_j) + (d_{T_i}(x, u_i) - d_G(x, u_i)) + (d_{T_j}(u_j, y) - d_G(u_j, y))$. Hence, $-1 \leq d_T(x, y) - d_G(x, y) \leq 1$ holds for all $x \in V(T_i)$ and $y \in V(T_j)$ if and only if $-1 \leq d_{T(V(A))}(u_i, u_j) - d_A(u_i, u_j) + tail(u_i) + tail(u_j) \leq 1$, i.e., $d_{T(V(A))}(u_i, u_j) - 1 \leq d_A(u_i, u_j) - tail(u_i) - tail(u_j) \leq d_{T(V(A))}(u_i, u_j) + 1$, holds for all u_i, u_j. If we will set also $tail(z) := 0$ for each vertex $z \in V(A) \setminus \{u_1, \ldots, u_k\}$, then $d_{T(V(A))}(v, w) \leq 3$ for any $v, w \in V(A)$ would imply $d_A(v, w) - tail(v) - tail(w) \leq 4$. Hence, if for some vertices $v, w \in V(A)$, we detect $d_A(v, w) - tail(v) - tail(w) > 4$, then we can terminate the consideration of this 2-connected component A and move to another 2-connected component A of G with a 2-cut. In what follows, we assume that $d_A(v, w) - tail(v) - tail(w) \leq 4$ for any $v, w \in V(A)$.

Let $A_1, \ldots, A_s$ be the connected components of the graph $A - c_1 - c_2$, where $\{c_1, c_2\}$ is a 2-cut of A. For this given 2-cut $\{c_1, c_2\}$ of A we can construct a new graph $\mathcal{H}_{c_1,c_2}$ as follows. The vertex set of $\mathcal{H}_{c_1,c_2}$ is $\{c_1, c_2, a_1, \ldots, a_s\}$. Edge c_1a_i $(i = 1, \ldots, s)$ exists in $\mathcal{H}_{c_1,c_2}$ if and only if for each $x, y \in V(A_i) \cup \{c_2\}$, $1 \leq d_A(x, y) - tail(x) - tail(y) \leq 3$ and $0 \leq d_A(x, c_1) - tail(x) - tail(c_1) \leq 2$ hold. Edge c_2a_i $(i = 1, \ldots, s)$ exists in $\mathcal{H}_{c_1,c_2}$ if and only if for each $x, y \in V(A_i) \cup \{c_1\}$, $1 \leq d_A(x, y) - tail(x) - tail(y) \leq 3$ and $0 \leq d_A(x, c_2) - tail(x) - tail(c_2) \leq 2$ hold. Edge a_ia_j $(i, j = 1, \ldots, s,\ i \neq j)$ exists in $\mathcal{H}_{c_1,c_2}$ if and only if for each vertex $x \in V(A_i)$ and each vertex $y \in V(A_j)$, $d_A(x, y) - tail(x) - tail(y) \leq 3$ holds. No other edges exist in $\mathcal{H}_{c_1,c_2}$.

We claim that if $\overline{\mathcal{H}_{c_1,c_2}}$ is bipartite with parts C_1 and C_2, where $c_1 \in C_1$, $c_2 \in C_2$,

then the tree T obtained from forest F by adding the edges of a bistar $T(V(A)) := (V(A), \{xc_1 : x \in A_i \text{ such that } a_i \in C_1\} \cup \{xc_2 : x \in A_j \text{ such that } a_j \in C_2\} \cup \{c_1c_2\})$ is a distance 1-approximating tree of G. And, if $\overline{\mathcal{H}_{c_1,c_2}}$ is not bipartite, then the forest F cannot be extended to a distance 1-approximating tree of G by adding the edges of any bistar $T(V(A))$ with the inner edge c_1c_2. So, in the latter case, we need to consider other 2-cuts of A (if any left) or to move to a new 2-connected component A of G with 2-cuts (if any left). The proof of the claim follows from the construction of $\mathcal{H}_{c_1,c_2}$ and from discussions above and in Subsection 4.2.2.

A formal description of the function Tree-with-a-bistar is given below.

Function Tree-with-a-bistar.

for each 2-connected component A of G with a 2-cut $\{c_1, c_2\}$
such that $d_A(c_1, c_2) \leq 2$ do
in each 2-connected component B of G different from A find a 1-cut
v_B of G closest to A;
if $ecc_B(v_B) \leq 2$ for all 2-connected components B ($B \neq A$) of G then do
set $E_F := \emptyset$;
for each 2-connected components B ($B \neq A$) of G
set $E_F := E_F \cup \{bv_B : b \in V(B) \setminus \{v_B\}\}$;
form a forest $F := (V, E_F)$;
let $u_1, \ldots, u_k$ be all 1-cuts of G located in A;
let $T_1, \ldots, T_k$ be the trees from F containing those 1-cuts, i.e., $u_i \in V(T_i)$,
$i = 1, \ldots, k$;
for all $i = 1, \ldots, k$, check if T_i is a distance 1-approximating tree of $G(V(T_i))$;
if "yes" then do
compute $tail(u_i)$ for each $i = 1, \ldots, k$;
set $tail(z) := 0$ for each vertex $z \in V(A) \setminus \{u_1, \ldots, u_k\}$;
if $d_A(v, w) - tail(v) - tail(w) \leq 4$ for any $v, w \in V(A)$, then do

for each 2-cut $\{c_1, c_2\}$ of A such that $d_A(c_1, c_2) \leq 2$ do

construct the graph $\mathcal{H}_{c_1,c_2}$;

if $\overline{\mathcal{H}_{c_1,c_2}}$ is bipartite, then do

create a bistar $T(V(A))$ on A as described above;

create a tree T by adding edges of $T(V(A))$ to the forest F;

output T and stop.

return "G does not have a distance 1-approximating tree".

The complexity of algorithm Dist-1-appr-tree is dominated by the complexity of function Tree-with-a-bistar. Each step before PHASE I (excluding the call to function Tree-with-a-bistar) requires at most $O(|V||E|)$ time. In PHASE I and PHASE II, we construct at most $O(|V|)$ trees. Each tree can be constructed and checked for being a distance 1-approximating tree of G in at most $O(|V|^2)$ time. Hence, the complexity of algorithm Dist-1-appr-tree is $O(|V|^3)$ plus the complexity of function Tree-with-a-bistar. Function Tree-with-a-bistar essentially considers all 2-cuts of G and tries to build for each 2-cut $\{c_1, c_2\}$ a tree T with an inner edge c_1c_2. There are at most $|V|^2$ 2-cuts in G. For a given 2-connected component A and its 2-cut $\{c_1, c_2\}$, one can check all necessary conditions and build the graph $\overline{\mathcal{H}_{c_1,c_2}}$ in $O(|V|^2)$ total time. Since we build at most $O(|V|^2)$ $\overline{\mathcal{H}_{c_1,c_2}}$ graphs in total, the entire function Tree-with-a-bistar requires no more than $O(|V|^4)$ time.

Thus, we have proven the following result.

Theorem 44 *It is possible, for a given connected graph $G = (V, E)$, to check in $O(|V|^4)$ time whether G has a distance 1-approximating tree and, if such a tree exists, construct one within the same time bound.*

Because of the space limitation, we will not include our complexity result on distance

approximating trees in this dissertation. Instead, the following theorem is given without proof.

Theorem 45 *Let $\Delta \geq 5$ be a positive number. Given a graph G, to decide whether G admits a distance $(\Delta, 0)$-approximating tree is NP-complete.*

The detailed proof of this theorem will appear in [47].

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this dissertation, we studied three topics. They are *Graph Spanners, Collective Tree Spanners* and *Distance Approximating Trees*. On graph spanners, we designed efficient algorithms for constructing additive spanners for chordal graphs. These improved previous results by Peleg and Schäffer [97]. Our algorithms are much faster and the spanners constructed are better. Moreover, the algorithms can be extended to more general cases. On collective tree spanners, we designed algorithms for constructing collective tree spanners for several graph classes. They immediately give us efficient routing algorithms for these graph classes. It is worth to emphasize that as a byproduct, we can use our results of collective tree spanners to design a very efficient routing schemes for unit disk graphs. According to what we know, this is the first time that a routing scheme for UDG can achieve bounded stretch factor. On distance approximating trees, we proved that in most cases, it is NP-hard to decide whether a general graph admits a distance approximating tree. This answered the open question posted by A. Brandstädt, V. Chepoi and F. F. Dragan [19]. We also presented a polynomial time algorithm to decide whether a graph admits a distance (1, 1)-approximating tree. The algorithm will construct such a tree if it exists.

An interesting question is whether one can apply the ideas presented in this dissertation to other graph classes to get additive graph spanners? One possible candidate is the n-hypercube. We know that the n-hypercube admits a multiplicative 3-spanner with $4n$ edges [54]. Some other interesting graph classes could be the bounded tree-length graphs. We are also interested in finding better additive spanners for interval graphs,

permutation graphs, AT-free graphs. We already know that they admit additive tree spanners. Also, it is natural to ask what is the complexity of constructing collective tree spanners for general graphs? On distance approximating trees, the open questions are what are the complexities of finding a distance $(\Delta, 0)$-approximating tree for general graphs, when Δ=2, 3, 4?

BIBLIOGRAPHY

[1] I. Abraham, C. Gavoille, and D. Malkhi, Compact routing for graphs excluding a fixed minor, *19th International Symposium on Distributed Computing (DISC'05)*, Springer, Lecture Notes in Computer Science 3724, pp. 442-456.

[2] L. Aleksandrov and H. Djidjev, Linear Algorithms for Partitioning Embedded Graphs of Bounded Genus, *SIAM Journal on Discrete Mathematics* 9 (1996), 129–150.

[3] N. Alon, P. Seymour, and R. Thomas, A Separator Theorem for Graphs with an Excluded Minor and its Applications, Proceedings of the *22nd Annual ACM Symposium on Theory of Computing*, ACM, 1990, pp. 293–299.

[4] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares, On sparse spanners of weighted graphs, *Discrete Comput. Geom.,* 9 (1993), 81–100.

[5] K. Alzoubi, X. Y. Li, Y. Wang, and P. J. Wang, Geometric spanners for wireless ad hoc networks, *IEEE Transac. on Parallel and Distributed Systems* , 14, NO. 5 (2003), 1–14.

[6] 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.

[7] B. Awerbuch, A. Bar Noy, N. Linial and D. Peleg, Improved routing strategies with succinct tables, *J. Algorithms,* 11(1990), 307–341.

[8] B. Awerbuch and D. Peleg, Routing with polynomial communication-space tradeoff, *SIAM J. Discrete Math.,* 5(1992), 151–162.

[9] H.-J. Bandelt, and A. Dress, Reconstructing the shape of a tree from observed dissimilarity data, *Adv. Appl. Math.,* 7 (1986), 309-343.

[10] Y. Bartal, Probabilistic approximations of metric spaces and its algorithmic applications, Proceedings of the *37th Annual Symposium on Foundations of Computer Science*, pp. 184–193, 1996.

[11] Y. Bartal, On approximating arbitrary metrices by tree metrics, Proceedings of the *13th Annual ACM Symposium on Theory of Computing,* pp. 161-168, 1998.

[12] J.-P. Barthélemy and A. Guénoche, Trees and Proximity Representations, *Wiley,* New York, 1991.

[13] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis, On the desirability of acyclic database schemes, *J. ACM,* 30 (1983), 479–513.

[14] C. Berge, Hypergraphs, *North Holland*, 1989.

[15] S. Bhatt, F. Chung, F. Leighton, and A. Rosenberg, Optimal simulations of tree machines, Proceedings of the *27th Annual Symposium on Foundations of Computer Science,* pp. 274–282, 1986.

[16] H.L. Bodlaender, Discovering Treewidth, *SOFSEM 2005: Theory and Practice of Computer Science, 31st Conference on Current Trends in Theory and Practice of Computer Science*, January 22-28, 2005, Springer, Lecture Notes in Computer Science 3381, pp. 1-16.

[17] R. Borie, J.L. Johnson, V. Raghavan, and J.P. Spinrad, Robust polynomial time algorithms on clique-width k graphs, *Manuscript,* 2002.

[18] U. Brandes and D. Handke, NP–Completeness Results for Minimum Planar Spanners, Preprint University of Konstanz, *Konstanzer Schriften in Mathematik und Informatik,* Nr. 16, Oktober 1996.

[19] A. Brandstädt, V. Chepoi, and F.F. Dragan. Distance Approximating Trees for Chordal and Dually Chordal Graphs, *J. Algorithms*, 30 (1999), 166-184.

[20] A. Brandstädt, F. Dragan, V. Chepoi, and V. Voloshin, Dually chordal graphs, *SIAM J. Discrete Math.,* 11 (1998), 437-455.

[21] A. Brandstädt, F.F. Dragan, H.-O. Le, and V.B. Le, Tree Spanners on Chordal Graphs: Complexity, Algorithms, Open Problems, *Theoretical Computer Science* Vol. 310, no. 1-3 (2004), 329-354.

[22] 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.

[23] A. Brandstädt, V.B. Le and J. Spinrad, Graph Classes: A Survey, *SIAM,* Philadelphia, 1999.

[24] A.E. Brouwer, P. Duchet, and A. Schrijver, Graphs whose neighbourhoods have no special cycles, *Discrete Math.,* 47 (1983), 177–182.

[25] P. Buneman, A characterization of rigid circuit graphs, *Discrete Math.*, 9 (1974), 205–212.

[26] L. Cai, Tree spanners: Spanning trees that approximate the distances, *Ph.D. dissertation*, University of Toronto, 1992.

[27] L. Cai and D.G. Corneil, Tree spanners, *SIAM J. Discrete Math.,* 8 (1995), 359–387.

[28] H. T-H. Chan, A. Gupta, B. M. Maggs, and S.H. Zhou On hierarchical routing in doubling metrics, In proceedings of the *16th Symposium on Discrete Algorithms,* (SODA'05), pp. 762-771.

[29] M. Charikar, C. Chekuri, A. Goel, S. Guha, and S. Plotkin, Approximating a Finite Metric by a Small Number of Tree Metrics, *39th Annual Symposium on Foundations of Computer Science*, pp. 379–388, 1998.

[30] V. Chepoi and F.F. Dragan, A note on distance approximating trees in graphs, *European Journal of Combinatorics* 21 (2000), 761–766.

[31] V. D. Chepoi, F. F. Dragan, and C. Yan, Additive Spanners for k-Chordal Graphs, *International Conference on Algorithm and Complexity (CIAC'03)* , May 2003, Springer, Lecture Notes in Computer Science 2653, pp. 96-107.

[32] V. D. Chepoi, F. F. Dragan, and C. Yan, Additive Spanners for Graphs with Bounded Length of Largest Induced Cycle, *Theoretical Computer Science*, Vol. 38, no. 5(2005), 623-645.

[33] L.P. Chew, There are planar graphs almost as good as the complete graph, *J. Comput. System Sci.,* 39 (1989), 205–219.

[34] D.G. Corneil, F.F. Dragan, E. Köhler, and C. Yan, Collective tree 1-spanners for interval graphs, *31st International Workshop "Graph-Theoretic Concepts in Computer Science" (WG '05)*, June 2005, Springer, Lecture Notes in Computer Science 3787, pp. 151–162.

[35] D.G. Corneil, S. Olariu, and L. Stewart, Linear time algorithms for dominating pairs in asteroidal triple–free graphs, *SIAM J. on Computing*, 28 (1999), 1284-1297.

[36] D.G. Corneil, S. Olariu, and L. Stewart, Asteroidal Triple–free Graphs, *SIAM J. Discrete Math.*, 10 (1997), 399-430.

[37] B. Courcelle and S. Olariu, Upper bounds to the clique-width of graphs, *Discrete Appl. Math.*, 101 (2000), 77-114.

[38] L. Cowen, Compact routing with minimum stretch, *10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1999)*, ACM–SIAM 1999, pp. 255–260.

[39] G.A. Dirac, On rigid circuit graphs, *Abh. Math. Sem. Univ. Hamburg,* 25 (1961), 71–76.

[40] H.N. Djidjev, On the Problem of Partitioning Planar Graphs, *SIAM J. Alg. Disc. Meth.,* 3 (1982), 229–240.

[41] H.N. Djidjev, A separator theorem for graphs of fixed genus, *Serdica*, 11 (1985), 319-329.

[42] Y. Dourisboure and C. Gavoille, Improved Compact Routing Scheme for Chordal Graphs, *16th International Conference on Distributed Computing (DISC 2002)*, October 28-30, 2002, Springer, *Lecture Notes in Computer Science* 2508, pp. 252-264.

[43] Y. Dourisboure and C. Gavoille, Spanners for bounded tree-length graphs, *11th interntaional Colloquium, (SIROCCO'04)* Springer, Lecture Notes in Computer Science, 3104(2007), 123-137.

[44] F.F. Dragan, Estimating All Pairs Shortest Paths in Restricted Graph Families: A Unified Approach, *Journal of Algorithms,* 57(2005), 1-21.

[45] F.F. Dragan and I. Lomonosov, On Compact and Efficient Routing in Certain Graph Classes, *15th Annual International Symposium on Algorithms and Computation (ISAAC 2004)*, December 20 - 22, 2004, Springer, Lecture Notes in Computer Science 3341, pp. 402-414.

[46] F.F. Dragan, and V. Voloshin, Incidence graphs of biacyclic hypergraphs, *Discrete Appl. Math.,* 68 (1996), 259-266.

[47] F.F. Dragan, C. Yan Distance approximating trees: complexity and algorithms, *6th Italian Conference on Algorithms and Complexity (CIAC'06)* Springer, Lecture Notes in Computer Science, 3998, 260-271.

[48] F.F. Dragan and C. Yan, Collective tree spanners of graphs with bounded genus, chordality tree-width or clique-width, *16th Annual International Symposium on Algorithms and Computation (ISAAC'05),* December, 2005, Springer, Lecture Notes in Computer Science 3827, 583-592.

[49] F.F. Dragan, C. Yan and D.G. Corneil, Collective Tree Spanners and Routing in AT-free Related Graphs, *Graph-Theoretic Concepts in Computer Science (WG'04)*, Junce, 2004, Springer, Lecture Notes in Computer Science, 3353, 68-80.

[50] F.F. Dragan, C. Yan and D.G. Corneil, Collective Tree Spanners and Routing in AT-free Related Graphs, *Journal of Graph Algorithms and Applications*, Vol. 10, no. 2(2007), 97-122.

[51] F.F. Dragan, C. Yan and I. Lomonosov, Collective tree spanners of graphs, *9th International Scandinavian Workshop on Algorithm Theory (SWAT'04)*, July, 2004, Springer, Lecture Notes in Computer Science, 3111, 64-76.

[52] F.F. Dragan, C. Yan and I. Lomonosov, Collective tree spanners of graphs, *SIAM J. Discrete Mathematics*, Vol. 20, no. 1(2006), 241-260.

[53] F.F. Dragan, C. Yan, and Y. Xiang Collective tree spanners for unit disk graphs, *in preparation.*

[54] W. Duckwortha and M. Zito, Sparse Hypercube 3-spanners, *Discrete Appl. Math.,* 103 (2000), 289–295.

[55] T. Eilam, C. Gavoille, and D. Peleg, Compact routing schemes with low stretch factor, *17th Annual ACM Symp. Prin. Distr. Comput. (PODC 1998),* ACM 1998, pp. 11–20.

[56] J. Engelfriet and G. Rozenberg, Node replacement graph grammars, *Handbook of Graph Grammars and Computing by Graph Transformation, Foundations,* Vol. I, World Scientific, Singapore, 1997, pp. 1-94.

[57] D.B.A. Epstein, J.W. Cannon, D.F. Holt, S.V.F. Levy, M.S. Paterson, and W.P. Thruston, Word processing in groups, *Jones and Bartlett,* Boston, 1992.

[58] M. Farber, Characterizations of strongly chordal graphs, *Discrete Math.,* 43 (1983), 173–189.

[59] R. Fagin, Degrees of acyclicity for hypergraphs and relational database schemes, *J. ACM,* 30 (1983), 514–550.

[60] J. Fakcharoenphol, S. Rao, and K. Talwar, A tight bound on approximating arbitrary metrics by tree metrics, *35th ACM Symposium on Theory of Computing*, ACM 2003, pp. 448 - 455.

[61] S.P. Fekete, J. Kremer, Tree spanners in planar graphs, *Discrete Appl. Math.*, 108 (2001), 85-103.

[62] P. Fraigniaud and C. Gavoille, Memory requirements for univesal routing schemes, *14th Annual ACM Symp. Prin. Distr. Comput. (PODC 1995),* ACM 1995, pp. 223–230.

[63] P. Fraigniaud and C. Gavoille, Routing in Trees, *28th Int. Colloquium on Automata, Languages and Programming (ICALP 2001),* Springer, 2001, Lecture Notes in Computer Science 2076, pp. 757–772.

[64] C. Gavoille, A survey on interval routing schemes, *Theoret. Comput. Sci.*, 245 (1999), 217–253.

[65] C. Gavoille and M. Gengler, Space-efficiency of routing schemes of stretch factor three, *J. Parallel and Distr. Comput.*, 61(2001), 679–687.

[66] C. Gavoille, M. Katz, N.A. Katz, C. Paul, and D. Peleg, Approximate Distance Labeling Schemes, *9th Annual European Symposium on Algorithms" (ESA'01)*, August 28-31, 2001, Springer, *Lecture Notes in Computer Science 2161*, pp. 476-487.

[67] C. Gavoille, D. Peleg, S. Pérennes, and R. Raz, Distance labeling in graphs, *12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2001)*, January 7-9, 2001, ACM-SIAM, 2001, pp. 210–219.

[68] C. Gavoille and S. Pérennès, Memory requirements for routing in distributed networks, *15th Ann. ACM Symp. on Prin. Distr. Comput. (PODC 1996),* ACM 1996, pp. 125–133.

[69] E. Ghys, and P. de la Harpe, Les groupes hyperboliques d'après M. Gromov, *Prog. Math.,* 83, 1990.

[70] J.R. Gilbert, J.P. Hutchinson, and R.E. Tarjan, A separator theorem for graphs of bounded genus, *Journal of Algorithms*, 5 (1984), 391–407.

[71] J.R. Gilbert, D.J. Rose, and A. Edenbrandt, A separator theorem for chordal graphs, *SIAM J. Alg. Discrete Meth.*, 5 (1984), 306-313.

[72] M.C.Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[73] M.C. Golumbic and U. Rotics, On the Clique-Width of Perfect Graph Classes, *25th International Workshop "Graph-Theoretic Concepts in Computer Science" (WG '99)*, June 1999, Ascona, Switzerland, Springer, Lecture Notes in Computer Science 1665, pp. 135-147.

[74] A. Gupta, Improved bandwidth approximation for trees and chordal graphs, *J. of Algorithms,* 40(1) (2001), pp. 24-36. Appeared also in SODA 2000.

[75] A. Gupta, A. Kumar and R. Rastogi, Traveling with a Pez Dispenser (or, Routing Issues in MPLS), *SIAM J. Comput.,* 34 (2005), pp. 453-474. Appeared also in FOCS 2001.

[76] O. Kariv and S.L. Hakimi, An algorithmic approach to network location problems, I: the p-centers, *SIAM J. Appl. Math.* 37 (1979), 513–538.

[77] M. Katz, N.A. Katz, and D. Peleg, Distance labeling schemes for well-separated graph classes, *17th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2000)*, February 2000, Springer, 2000, *Lecture Notes in Computer Science* 1770, pp. 516–528.

[78] T. Kloks, Treewidth: Computations and Approximations, *Lecture Notes in Computer Science* 842, Springer, Berlin, 1994.

[79] T. Kloks, D. Kratsch, and H. Müller, Approximating the bandwidth for asteroidal triple-free graphs, *J. of Algorithms*, 32 (1999), 41–57.

[80] T. Kloks, D. Kratsch, and H. Müller, On the structure of graphs with bounded asteroidal number, *Graphs and Combinatorics*, 17 (2001), 295–306.

[81] D. Kratsch, H.-O. Le, H. Müller, E. Prisner, and D. Wagner, Additive tree spanners, *SIAM J. Discrete Math.* 17 (2003), 332-340.

[82] R. Krauthgamer and J.R. Lee, The intrinsic dimensionality of graphs, *32th annual ACM Symposium on Theory of Computing (STOC'03)*, pp. 438–447, 2003.

[83] R. Krauthgamer, N. Linial, and A. Magen, Metric Embedding – Beyond one-dimensional distortion, *Discrete and Computational Geometry* 31 (2004), 339–356.

[84] H.-O Le, V.B. Le, Optimal tree 3-spanners in directed path graphs, *Networks,* 34 (1999), 81-87.

[85] A.L. Liestman and T. Shermer, Additive graph spanners, *Networks,* 23 (1993), 343–364.

[86] A.L. Linial and E. London, The geometry of graphs and some of its algorithmic applications *Combinatorica,* 15 (1995), 215–245.

[87] R. J. Lipton and R. E. Tarjan, A separator theorem for planar graphs, *SIAM J. Appl. Math.*, 36 (1979), 346-358.

[88] R.J. Lipton and R.E. Tarjan, A separator theorem for planar graphs, *SIAM J. Appl. Math.*, 36 (1979), 177-189.

[89] R.J. Lipton and R.E. Tarjan, Applications of a planar separator theorem, *SIAM J. Comput.,* 9 (1980), 615-627.

[90] L. Lovász, Normal hypergraphs and the perfect graph conjecture, *Discrete Math.*, 2 (1972), 253–267.

[91] A. Lubiw, Doubly lexical orderings of matrices, *SIAM J. Comput.*, 16 (1987), 854–879.

[92] T.H. Ma and J.P. Spinrad, On the two-chain subgraph cover and related problems, *J. of Algorithms*, 17(1994), 251–268.

[93] M.S. Madanlal, G. Venkatesan, C. Pandu Rangan, Tree 3-spanners on interval, permutation and regular bipartite graphs, *Inform. Process. Lett.,* 59 (1996), 97-102.

[94] R.M. McConnell and J.P. Spinrad, Linear-time transitive orientation, *8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1997),* 5-7 January 1997, pp. 19-25.

[95] T.A. McKee, personal communication to E. Prisner, 1995.

[96] D. Peleg, Distributed Computing: A Locality-Sensitive Approach, *SIAM Monographs on Discrete Math. Appl.*, SIAM, Philadelphia, 2000.

[97] D. Peleg, and A.A. Schäffer, Graph Spanners, *J. Graph Theory*, 13 (1989), 99-116.

[98] D. Peleg and J.D. Ullman, An optimal synchronizer for the hypercube, *6th ACM Symp. on Prin. of Distr. Comput. (PODC 1987),* ACM 1987, 77–85.

[99] D.Peleg, and E.Upfal, A tradeoff between space and efficiency for routing tables, *20th ACM Symposium on the Theory of Computing,* ACM 1988, 43-52.

[100] E. Prisner, Distance approximating spanning trees, *14th Annual Symposium on Theoretical Aspects of Computer Science (STACS 1997)*, February 1997, Springer, *Lecture Notes in Computer Science* 1200, pp. 499–510.

[101] E. Prisner, D. Kratsch, H.-O. Le, H. Müller, and D. Wagner, Additive tree spanners, *SIAM J. Discrete Math.*, 17 (2003), 332–340.

[102] N. Robertson and P.D. Seymour, Graph minors. Algorithmic aspects of tree-width, *J. Algorithms,* 7 (1986), 309–322.

[103] P.H.A. Sneath and R.R. Sokal, Numerical Taxonomy, *W.H. Freeman,* San Francisco, California, 1973.

[104] J. Soares, Graph spanners: A survey, *Congressus Numer.,* 89 (1992), 225-238.

[105] J.P. Spinrad, Doubly lexical ordering of dense 0-1– matrices, *Inform. Process. Lett.,* 45 (1993), 229–235.

[106] D.L. Swofford and G.J. Olsen, Phylogeny reconstruction, In *Molecular Systematics (D.M. Hillis and C. Moritz, editors),* Sinauer Associates Inc., Sunderland, MA., 1990, 411–501.

[107] R.E. Tarjan and M. Yannakakis, Simple linear time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs, *SIAM J. Comput.,* 13 (1984), 566–579.

[108] M. Thorup and U. Zwick, Compact routing schemes, *13th Annual ACM Symp. on Par. Alg. and Arch. (SPAA 2001)*, ACM 2001, pp. 1–10.

[109] G. Venkatesan, U. Rotics, M.S. Madanlal, J.A. Makowsky, C. Pandu Ragan, Restrictions of minimum spanner problems, *Information and Computation,* 136 (1997), 143-164.

[110] Y. Xiang, Ph.D. dissertation.