Tree Spanners for Bipartite Graphs and Probe Interval Graphs

Andreas Brandstädt[1], Feodor F. Dragan[2], Hoang-Oanh Le[1], Van Bang Le[1], and Ryuhei Uehara[3]

[1] Institut für Theoretische Informatik, Fachbereich Informatik, Universität Rostock, 18051 Rostock, Germany.
{ab,hoang-oanh.le,le}@informatik.uni-rostock.de
[2] Dept. of Computer Science, Kent State University, Ohio, USA.
dragan@cs.kent.edu
[3] Natural Science Faculty, Komazawa University, Tokyo, Japan.
uehara@komazawa-u.ac.jp

Abstract. A tree t-spanner T in a graph G is a spanning tree of G such that the distance between every pair of vertices in T is at most t times their distance in G. The tree t-spanner problem asks whether a graph admits a tree t-spanner, given t. We first substantially strengthen the known results for bipartite graphs. We prove that the tree t-spanner problem is NP-complete even for chordal bipartite graphs for $t \geq 5$, and every bipartite ATE–free graph has a tree 3-spanner, which can be found in linear time. The best known before results were NP-completeness for general bipartite graphs, and that every convex graph has a tree 3-spanner. We next focus on the tree t-spanner problem for probe interval graphs and related graph classes. The graph classes were introduced to deal with the physical mapping of DNA. From a graph theoretical point of view, the classes are natural generalizations of interval graphs. We show that these classes are tree 7-spanner admissible, and a tree 7-spanner can be constructed in $O(m \log n)$ time.

Keywords: Chordal bipartite graph, Interval bigraph, NP-completeness, Probe interval graph, Tree spanner

1 Introduction

A *tree t-spanner* T in a graph G is a spanning tree of G such that the distance between every pair of vertices in T is at most t times their distance in G. The *tree t-spanner problem* asks whether a graph admits a tree t-spanner, given t. The notion is introduced by Cai and Corneil [8,9], which finds numerous applications in distributed systems and communication networks; for example, it was shown that tree spanners can be used as models for broadcast operations [1] (see also [23]). Moreover, tree spanners were used in the area of biology [2], and approximating the bandwidth of graphs [27]. We refer to [24,26,6] for more background information on tree spanners.

H.L. Bodlaender (Ed.): WG 2003, LNCS 2880, pp. 106–118, 2003.

The tree t-spanner problem is NP-complete in general [9] for any $t \geq 4$. However, it can be solved efficiently for some particular graph classes. Especially, the complexity of the tree t-spanner problem is well investigated for chordal graphs and its subclasses. For $t \geq 4$ the problem is NP-complete for chordal graphs [6], strongly chordal graphs are tree 4-spanner admissible [3] (i.e., every strongly chordal graph has a tree 4-spanner), and the following graph classes are tree 3-spanner admissible: interval graphs [18], directed path graphs [17], split graphs [27] (see also [6]).

We first focus on the tree t-spanner problem for bipartite graphs and its subclasses. The class of bipartite graphs is wide and important from both practical and theoretical points of view. However, the known results for the complexity of the tree t-spanner problem for bipartite graph classes are few comparing to chordal graph classes. The NP-completeness is only known for general bipartite graphs (this result can be deduced from the construction in [9]), and the problem can be solved for regular bipartite graphs, and convex graphs as follows; a regular bipartite graph is tree 3-spanner admissible if and only if it is complete [18]; and any convex graph is tree 3-spanner admissible [27].

We substantially strengthen the known results for bipartite graph classes, and reduce the gap. We show that the tree t-spanner problem is NP-complete even for chordal bipartite graphs for $t \geq 5$. The class of chordal bipartite graphs is a bipartite analog of chordal graphs, introduced by Golumbic and Goss [13], and has applications to nonsymmetric matrices [12]. We also show that every bipartite asteroidal-triple-edge–free (ATE–free) graph has a tree 3-spanner, and such a tree spanner can be found in linear time. The class of ATE–free graphs was introduced by Müller [22] to characterize interval bigraphs. The class of interval bigraphs is a bipartite analog of interval graphs and was introduced by Harary, Kabell, and McMorris [14].

Our results reduce the gap between the upper and lower bounds of the complexity of the tree t-spanner problem for bipartite graph classes since the following proper inclusions are known [22,7]; convex graphs $\subset$ interval bigraphs $\subset$ bip. ATE–free graphs $\subset$ chordal bipartite graphs $\subset$ bipartite graphs.

We next focus on the tree t-spanner problem on probe interval graphs and related graph classes. The class of probe interval graphs was introduced by Zhang to deal with the physical mapping of DNA, which is a problem arising in the sequencing of DNA (see [28,21,20,29] for background). A probe interval graph is obtained from an interval graph by designating a subset P of vertices as *probes*, and removing the edges between pairs of vertices in the remaining set N of *nonprobes*. In the original papers [28,29], Zhang introduced two variations of probe interval graphs. An enhanced probe interval graph is the graph obtained from a probe interval graph by adding the edges joining two nonprobes if they are adjacent to two independent probes. The class of STS-probe interval graphs is a subset of the probe interval graphs; in those graphs all probes are independent.

From the graph theoretical point of view, it has been shown that all probe interval graphs are weakly chordal [21], and enhanced probe interval graphs are chordal [28,29]. In full version, we show that (1) the class of STS-probe interval

graphs is equivalent to the class of convex graphs (hence the class is tree 3-spanner admissible), and (2) the class of the (enhanced) probe interval graphs is incomparable with the classes of strongly chordal graphs and rooted directed path graphs.

Hence, from both viewpoints of graph theory and biology, the tree t-spanner problem for (enhanced) probe interval graphs is worth investigating. Especially, it is natural to ask that if those graph classes are tree t-spanner admissible for fixed integer t. We give the positive answer to that question: The classes of probe interval graphs and enhanced probe interval graphs are tree 7-spanner admissible. A tree 7-spanner of a (enhanced) probe interval graph can be constructed in $O(m + n \log n)$ time if it is given with an interval model. Therefore, using the recognition algorithms in [15,19], we can construct a tree 7-spanner for a given (enhanced) probe interval graph $G = (P, N, E)$ in $O(m \log n)$ time.

Due to space limitation, proofs are omitted, and can be found in full version[1].

2 Preliminaries

Given a graph $G = (V, E)$ and a subset $U \subseteq V$, the *subgraph of G induced by U* is the graph (U, F), where $F = \{\{u, v\} | \{u, v\} \in E \text{ for } u, v \in U\}$, and denoted by $G[U]$. For a subset F of E, we sometimes unify the edge set F and its *edge induced subgraph* (U, F) with $U = \{v | \{u, v\} \in F \text{ for some } u \in V\}$. For two vertices u and v on G, the *distance* of the vertices is the minimum length of the paths joining u and v, and denoted by $d_G(u, v)$. The *disk* of radius k centered at v is the set of all vertices with distance at most k to v, $D_k(v) = \{w \in V : d_G(v, w) \leq k\}$, and the *$k$th neighborhood* $N_k(v)$ of v is defined as the set of all vertices at distance k to v, that is $N_k(v) = \{w \in V : d_G(v, w) = k\}$. By $N(v)$ we denote the *neighborhood* of v, i.e., $N(v) := N_1(v)$. More generally, for a subset $S \subseteq V$ let $N(S) = \cup_{v \in S} N(v)$ denote the *neighborhood* of S.

A *tree t-spanner* T in a graph G is a spanning tree of G such that for each pair u and v in G, $d_T(u, v) \leq t \cdot d_G(u, v)$. We say that G is *tree t-spanner admissible* if it contains a tree t-spanner. The *tree t-spanner problem* is to determine, for given graph and positive integer t, if the graph admits a tree t-spanner. A class C of graphs is *tree t-spanner admissible* if every graph in C is tree t-spanner admissible. On the tree t-spanner problem, the following result plays an important role:

Lemma 1. [9] *A spanning tree T of G is a tree t-spanner if and only if for every edge $\{u, v\}$ of G, $d_T(u, v) \leq t$.*

It is well known that a graph G is bipartite if and only if G contains no cycle of odd length [16]. Thus, for each positive integer k, a tree $2k$-spanner of a bipartite graph G is also a tree $(2k - 1)$-spanner. Hence we will consider a tree t-spanner for each odd number t for bipartite graphs.

A graph (V, E) with $V = \{v_1, v_2, \cdots, v_n\}$ is an *interval graph* if there is a set of intervals $\mathcal{I} = \{I_1, I_2, \cdots, I_n\}$ such that $\{v_i, v_j\} \in E$ if and only if $I_i \cap I_j \neq \emptyset$

[1] Full version is available at `http://www.komazawa-u.ac.jp/~uehara/ps/t-span.pdf`

for each $1 \leq i, j \leq n$. We call the set $\mathcal{I}$ *interval representation* of the graph. For each interval I, we denote by $R(I)$ and $L(I)$ the right and left endpoints of the interval, respectively. A bipartite graph (X, Y, E) with $X = \{x_1, x_2, \cdots, x_{n_1}\}$ and $Y = \{y_1, y_2, \cdots, y_{n_2}\}$ is an *interval bigraph* if there are families of intervals $\mathcal{I}_X = \{I_1, I_2, \cdots, I_{n_1}\}$ and $\mathcal{I}_Y = \{J_1, J_2, \cdots, J_{n_2}\}$ such that $\{x_i, y_j\} \in E$ if and only if $I_i \cap J_j \neq \emptyset$ for each $1 \leq i \leq n_1$ and $1 \leq j \leq n_2$. We also call the families of intervals $(\mathcal{I}_X, \mathcal{I}_Y)$ *interval representation* of the graph. We sometimes unify a vertex v_i and its corresponding interval I_i; I_v denotes the interval corresponding to the vertex v, and $R(v)$ and $L(v)$ denote $R(I_v)$ and $L(I_v)$, respectively.

An edge which joins two vertices of a cycle but is not itself an edge of the cycle is a *chord* of that cycle. A graph is *chordal* if each cycle of length ≥ 4 has a chord. A graph G is *weakly chordal* if G and $\bar{G}$ contain no induced cycle C_k with $k \geq 5$. A bipartite graph G is *chordal bipartite* if each cycle of length ≥ 6 has a chord. Let the neighborhood $N(e)$ of an edge $e = \{v, w\}$ be the union $N(v) \cup N(w)$ of the neighborhoods of the end-vertices of e. Three edges of a graph G form an *asteroidal triple of edges* (ATE) if for any two of them there is a path from the vertex set from one to the vertex set of the other that avoids the neighborhood of the third edge. *ATE–free* graphs are those graphs which do not contain any ATE. This class of graphs was introduced in [22], where it was also shown that any interval bigraph is an ATE–free graph, and any bipartite ATE–free graph is chordal bipartite.

A graph $G = (V, E)$ is a *probe interval graph* if V can be partitioned into subsets P and N (corresponding to the *probes* and *nonprobes*) and each $v \in V$ can be assigned to an interval I_v such that $\{u, v\} \in E$ if and only if both $I_u \cap I_v \neq \emptyset$ and at least one of u and v is in P. In this paper, we assume that P and N are given, and we denote the considered probe interval graph by $G = (P, N, E)$. Let $G = (P, N, E)$ be a probe interval graph. Let E^+ be a set of edges $\{u_1, u_2\}$ with $u_1, u_2 \in N$ such that there are two probes v_1 and v_2 in P such that $\{v_1, u_1\}$, $\{v_1, u_2\}$, $\{v_2, u_1\}$, $\{v_2, u_2\} \in E$, and $\{v_1, v_2\} \notin E$. Each edge in E^+ is called an *enhanced edge*, and the resulting graph $G^+ = (P, N, E \cup E^+)$ is said to be an *enhanced probe interval graph*. See [28,21,20,29] for further details.

3 NP-Completeness for Chordal Bipartite Graphs

In this section we show that, for any $t \geq 5$, the tree t-spanner problem is NP-complete for chordal bipartite graphs. The proof is a reduction from Monotone 3SAT which consists of instances of 3SAT such that each clause contains either only negated variables or only non-negated variables (see [11, LO2]). For which the following family of chordal bipartite graphs will play an important role.

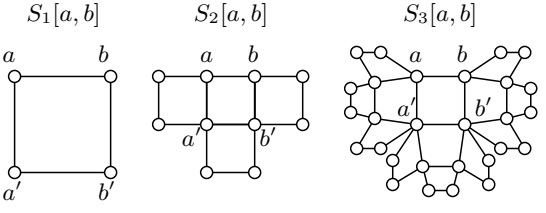

Fig. 1. The graph $S_\ell[a, b]$

First, $S_0[a,b]$ is an edge $\{a,b\}$, and $S_1[a,b]$ is the 4-cycle (a,a',b',b,a). Next, for a fixed integer $\ell > 1$, $S_{\ell+1}[a,b]$ is obtained from one cycle (a,b,b',a',a), $S_\ell[a,a']$, $S_\ell[b,b']$, and $S_\ell[a',b']$ by identifying the corresponding vertices (see Fig. 1). We will connect the vertices a and b to other graphs, and use $S_\ell[a,b]$ as a subgraph of bigger graphs. Sometimes, when the context is clear, we simply write S_ℓ for $S_\ell[a,b]$. In case $\ell > 0$ we write (a,a',b',b,a) for the 4-cycle in $S_\ell[a,b]$ containing the edge $\{a,b\}$. Each of the edges $\{a,a'\},\{a',b'\},\{b,b'\}$ belongs to a unique $S_{\ell-1}$, the *corresponding* $S_{\ell-1}$ in $S_\ell[a,b]$ to $\{a,a'\},\{a',b'\},\{b,b'\}$, respectively. The following observations collect basic facts on S_ℓ used in the reduction later.

Observation 1. *For every integer $\ell \geq 0$, $S_\ell[a,b]$ has a tree $(2\ell+1)$-spanner containing the edge $\{a,b\}$.*

Observation 2. *Let H be an arbitrary graph and let e be an arbitrary edge of H. Let K be an $S_\ell[a,b]$ disjoint from H. Let G be the graph obtained from H and K by identifying the edges e and $\{a,b\}$; see Fig. 2. Suppose that T is a tree t-spanner in G, $t > 2\ell$, such that the (a,b)-path in T belongs to H. Then $d_T(a,b) \leq t - 2\ell$.*

Observation 2 indicates a way to force an edge $\{x,y\}$ to be a tree edge: Choosing $\ell = \lfloor \frac{t-1}{2} \rfloor$ shows that $\{a,b\}$ must be an edge of T.

We now describe the reduction. Let $k \geq 2$ be an integer, and let F be a 3SAT formula with m clauses C_j for $1 \leq j \leq m$, over n variables x_i for $1 \leq i \leq n$.

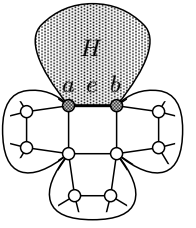

Fig. 2. The graph obtained from H and $S_\ell[a,b]$ by identifying the edge $e = \{a,b\}$

Definition 1. *In a graph G, an edge $\{a,b\}$ is said to be* forced by an S_ℓ *if $\{a,b\}$ appears in some $S_\ell[a,b]$ (as induced subgraph in G) such that $\{a,b\}$ disconnects $S_\ell[a,b]$ from the rest. We require that each two $S_\ell[a,b]$ and $S_{\ell'}[c,d]$ have at most 2 vertices in $\{a,b,c,d\}$ in common. An edge $\{a,b\}$ is said to be* strongly forced *if it is forced by two $S_k[a,b]$.*

By Observation 2, if G has a tree $(2k+1)$-spanner T every strongly forced edge must belong to T.

For each variable x_i create the gadget $G(x_i)$ as follows: (1) Take $2m+4$ vertices $x_i^1,\ldots,x_i^m$, $\overline{x_i}^1,\ldots,\overline{x_i}^m$, p_i,q_i,r_i,s_i, and (2) for $1 \leq j,j' \leq m$, add the edges $\{x_i^j,\overline{x_i}^{j'}\}$, $\{q_i,x_i^j\}$, $\{r_i,x_i^j\}$, $\{p_i,\overline{x_i}^j\}$, $\{s_i,\overline{x_i}^j\}$, and $\{p_i,r_i\},\{r_i,s_i\},\{s_i,q_i\}$. Furthermore, (3) each of the edges $\{p_i,r_i\},\{r_i,s_i\},\{s_i,q_i\}$, and $\{x_i^j,\overline{x_i}^j\}$ with $1 \leq j \leq m$, is a strongly forced edge, (4) force each edge $\{a,b\} \in \{\{q_i,x_i^j\} : 1 \leq j \leq m\} \cup \{\{r_i,x_i^j\} : 1 \leq j \leq m\} \cup \{\{p_i,\overline{x_i}^j\} : 1 \leq j \leq m\} \cup \{\{s_i,\overline{x_i}^j\} : 1 \leq j \leq m\} \cup \{\{x_i^j,\overline{x_i}^{j'}\} : 1 \leq j,j' \leq m, j \neq j'\}$ by an $S_{k-1}[a,b]$. (See Fig.3; in the figure, the S_k and S_{k-1} are omitted, and thick edges are strongly forced).

The vertex x_i^j ($\overline{x_i}^j$, respectively) will be connected to the clause gadget of clause C_j if x_i ($\overline{x_i}$, respectively) is a literal in C_j. All edges $\{r_i,x_i^j\}$ $(1 \leq j \leq m)$

or else all edges $\{s_i, \overline{x_i}^j\}$ $(1 \leq j \leq m)$ will belong to any tree $(2k+1)$-spanner (if any) of the graph G which we are going to describe.

Definition 2. *A clause is* positive *(*negative*, respectively) if it contains only variables (negation of variables).*

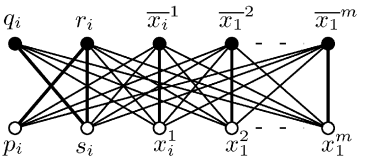

Fig. 3. $G(x_i)$

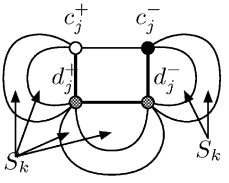

Fig. 4. $G(C_j)$

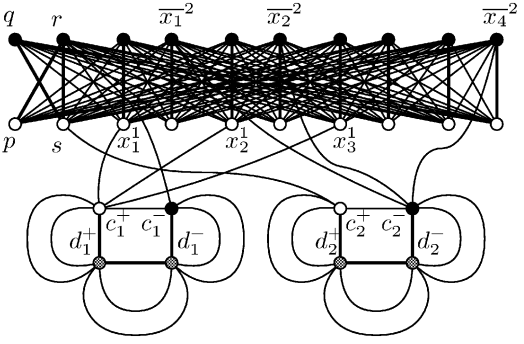

Fig. 5. The reduction given $C_1 = (x_1, x_2, x_3)$ and $C_2 = (\overline{x_1}, \overline{x_2}, \overline{x_4})$

We note that each clause is either positive or negative since given formula is an instance of Monotone 3SAT. For each clause C_j, $G(C_j)$ is the 4-cycle $(c_j^+, d_j^+, d_j^-, c_j^-, c_j^+)$ where $\{c_j^+, d_j^+\}$, $\{d_j^+, d_j^-\}$, and $\{d_j^-, c_j^-\}$ are strongly forced edges (see Fig.4).

Finally, the graph $G = G(F)$ is obtained from all $G(v_i)$ and $G(C_j)$ by identifying all vertices p_i, q_i, r_i and s_i to a single vertex p, q, r, and s, respectively (thus, $\{p, r\}$, $\{r, s\}$ and $\{s, q\}$ are edges in G), and adding the following edges: (1) Connect every x_i^j with every $\overline{x_{i'}}^{j'}$ $(i \neq i')$. (2) For every positive clause C_j: If x_i is in C_j then connect x_i^j with c_j^+ and force the edge $\{x_i^j, c_j^+\}$ by an $S_{k-2}[x_i^j, c_j^+]$. Connect c_j^- with r and force the edge $\{c_j^-, r\}$ by an $S_{k-2}[c_j^-, r]$. (3) For every negative clause C_j: If $\overline{x_i}$ is in C_j then connect $\overline{x_i}^j$ with c_j^- and force the edge $\{\overline{x_i}^j, c_j^-\}$ by an $S_{k-2}[\overline{x_i}^j, c_j^-]$. Connect c_j^+ with s and force the edge $\{c_j^+, s\}$ by an $S_{k-2}[c_j^+, s]$.

The description of the graph $G = G(F)$ is complete. Clearly, G can be constructed in polynomial time. See Fig. 5 for an example.

Lemma 2. *G is chordal bipartite.*

Lemma 3. *Suppose G admits a tree $(2k+1)$-spanner. Then F is satisfiable.*

Lemma 4. *Suppose F is satisfiable. Then G admits a tree $(2k+1)$-spanner.*

Theorem 3. *For every fixed $k \geq 2$, the Tree $(2k+1)$-Spanner problem is* NP*-complete for chordal bipartite graphs.*

4 Tree 3-Spanners for Bipartite ATE-Free Graphs

In this section we show that any bipartite ATE–free graph admits a tree 3-spanner.

We say that a vertex u of a graph G has a *maximum neighbor* if there is a vertex w in G such that $N(N(u)) = N(w)$. We will need the following result from [5].

Lemma 5. [5] *Any chordal bipartite graph G has a vertex with a maximum neighbor.*

It is easy to deduce from results [4, Lemma 4.4], [5, Corollary 5] and [10, Corollary 1] that a vertex with a maximum neighbor of a chordal bipartite graph can be found in linear time by the following procedure.

PROCEDURE NICE-VERTEX. Find a vertex with a maximum neighbor
Input: A chordal bipartite graph $G = (X \cup Y, E)$.
Output: A vertex with a maximum neighbor.
Method:
initially all vertices $v \in X \cup Y$ are unmarked;
repeat
among unmarked vertices of X select a vertex x such that $N(x)$ contains the maximum number of marked vertices;
mark x and all its unmarked neighbors;
until all vertices in Y are marked;
output the vertex of Y marked last.

Now let $G = (V, E)$ be a connected bipartite ATE–free graph and u be a vertex of G which has a maximum neighbor (recall that G is chordal bipartite and therefore such a vertex u exists).

Lemma 6. *Let S be a connected component of a subgraph of G induced by set $V \setminus D_{k-1}(u)$ ($k \geq 1$). Then, there is a vertex $w \in N_{k-1}(u)$ such that $N(w) \supset S \cap N_k(u)$.*

Lemma 6 suggests the following algorithm for constructing a spanning tree of G.

PROCEDURE SPAN-ATEG. Tree 3-spanners for bip. ATE–free graphs
Input: A bipartite ATE–free graph $G = (V, E)$ and a vertex u of G with a maximum neighbor.
Output: A spanning tree $T = (V, E')$ of G (rooted at u).
Method:
set $E' := \emptyset$;
set $q := \max\{d_G(u, v) : v \in V\}$;
let s_i^q, $i \in \{1, \ldots, p_q\}$ be the vertices of $N_q(u)$;
for every $i \in \{1, \ldots, p_q\}$ **do**
pick a neighbor w of s_i^q in $N_{q-1}(u)$;
add edge $\{s_i^q, w\}$ to E';
for $k := q - 1$ **downto** 1 **do**

compute the connected components $S_1^k, \ldots, S_{p_k}^k$ of
$G[N_k(u) \cup \{s_i^{k+1}, i \in \{1, \ldots, p_{k+1}\}\}]$;
for every $i \in \{1, \ldots, p_k\}$ **do**
set $S := S_i^k \cap N_k(u)$;
pick a vertex w in $N_{k-1}(u)$ such that $N(w) \supset S$;
for each $v \in S$ add the edge $\{v, w\}$ to E';
shrink component S_i^k to a vertex s_i^k and make s_i^k adjacent in G to
all vertices from $N(S_i^k) \cap N_{k-1}(u)$.

It is easy to see that the graph $T = (V, E')$ constructed by this procedure is a spanning tree of G and its construction takes only linear time. Moreover, T is a shortest path tree of G rooted at u since for any vertex $x \in V$, $d_G(x, u) = d_T(x, u)$ holds.

Theorem 4. *Let $T = (V, E')$ be a spanning tree of a bipartite ATE–free graph $G = (V, E)$ output by PROCEDURE SPAN-ATEG. Then, for any $x, y \in V$, we have $d_T(x, y) \leq 3 \cdot d_G(x, y)$ and $d_T(x, y) \leq d_G(x, y) + 2$.*

Since any interval bigraph is a bipartite ATE–free graph, and any convex graph is an interval bigraph, we have the following corollaries.

Corollary 1. *Any interval bigraph $G = (V, E)$ admits a spanning tree T such that $d_T(x, y) \leq 3 \cdot d_G(x, y)$ and $d_T(x, y) \leq d_G(x, y) + 2$ hold for any $x, y \in V$. Moreover, such a tree T can be constructed in linear time.*

Corollary 2. [27] *Any convex graph $G = (V, E)$ admits a spanning tree T such that $d_T(x, y) \leq 3 \cdot d_G(x, y)$ and $d_T(x, y) \leq d_G(x, y) + 2$ hold for any $x, y \in V$. Moreover, such a tree T can be constructed in linear time.*

5 Tree 7-Spanners for (Enhanced) Probe Interval Graphs

In this section we show that any (enhanced) probe interval graph admits a tree 7-spanner.

Let $G = (P, N, E)$ be a connected probe interval graph. We assume that an interval representation of G is given (if not, an interval model for G can be constructed by a method described in [19] in $O(m \log n)$ time, where $n = |P| + |N|$ and $m = |E|$). Let $\mathcal{I} = \{I_x : x \in P\}$ be the intervals in the interval model representing the probes and $\mathcal{J} = \{J_y : y \in N\}$ be the intervals representing the nonprobes.

First we discuss two simple special cases. If $N = \emptyset$ then clearly $G = (P, E)$ is an interval graph. It is known (see [25]) that for any interval graph G and any vertex u of G there is a shortest path spanning tree T of G rooted at u such that $d_T(x, y) \leq d_G(x, y) + 2$ holds for any x, y. In fact, a procedure similar to PROCEDURE SPAN-ATEG produces such a spanner in linear time for any interval graph G and any start vertex u. Evidently, T is a tree 3-spanner of G.

To describe other special case, we will need the following notion. A connected probe interval graph $G = (P, N, E)$ is *superconnected* if for any two intersecting

intervals $I_v, I_w \in \mathcal{I}$ there is always an interval $J_y \in \mathcal{J}$ such that $I_v \cap I_w \cap J_y \neq \emptyset$. For a superconnected probe interval graph G, a tree 4-spanner can be constructed easily. First we ignore all edges in $G[P]$ to get an interval bigraph $G' = (X = P, Y = N, E')$ and then run PROCEDURE SPAN-ATEG on G'. We claim that a spanning tree T of G', produced by that procedure, is a tree 4-spanner of G. Indeed, for any edge $\{x, y\}$ of G such that $x \in P$ and $y \in N$, $d_T(x, y) \leq 3$ holds by Corollary 1; it is an edge of G', too. Now consider an edge $\{v, w\}$ of G with $v, w \in P$. Since G is superconnected, there is a vertex $y \in N$ such that $I_v \cap I_w \cap J_y \neq \emptyset$, i.e., $d_{G'}(v, w) = 2$. Then, by Corollary 1, we have $d_T(v, w) \leq d_{G'}(v, w) + 2 = 2 + 2 = 4$. Consequently, T is a tree 4-spanner of G.

To get a tree 7-spanner for an arbitrary connected probe interval graph $G = (P, N, E)$, we will use the following strategy. First we decompose the graph G into subgraphs $G_0, G_1, \ldots, G_k$ such that G_i and G_j $(i \neq j)$ share at most one common vertex and each G_i is either an interval graph or a superconnected probe interval graph. Then iteratively, given a tree 7-spanner T^i for $G_0 \cup G_1 \cup \ldots \cup G_i$ $(i < k)$ and a tree t-spanner T_{i+1} $(t \leq 4)$ of G_{i+1}, we will extend T^i to a tree 7-spanner T^{i+1} for $G_0 \cup G_1 \cup \ldots \cup G_i \cup G_{i+1}$ by either making all vertices of G_{i+1} adjacent in T^{i+1} to a common neighbor in $G_0 \cup G_1 \cup \ldots \cup G_i$ (if it exists) or by gluing trees T^i and T_{i+1} at a common vertex.

Now we give a formal description of the decomposition algorithm. Let $S_0, S_1, \ldots, S_q$ be segments of the union $\cup_{y \in N} J_y$. (see Fig. 3 for an illustration).

Clearly, all probe interval graphs G_{2i+1} $(i = 1, \ldots, q)$ are superconnected and a decomposition of G into G_0,G_1,..., G_{2q+2} can be done in linear time if endpoints of the intervals $\mathcal{I} \cup \mathcal{J}$ are sorted.

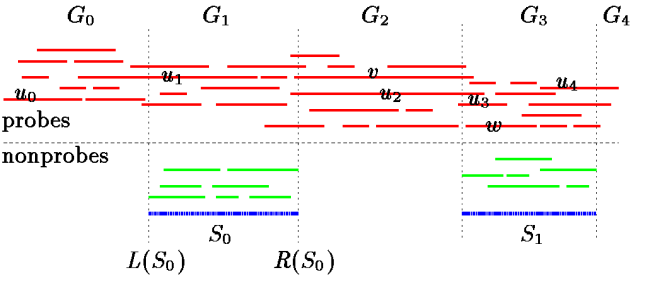

Fig. 3. Segments and a decomposition of a probe interval graph

PROCEDURE DECOMP. A decomposition of a probe interval graph

Input: A probe interval graph G and its interval representation $(\mathcal{I}, \mathcal{J})$.

Output: Subgraphs $G_0, G_1, \ldots, G_{2q+2}$ of G, where G_{2i} $(i \in \{0, \ldots, q+1\})$ is an interval graph and G_{2i+1} $(i \in \{0, \ldots, q\})$ is a superconnected probe interval graph, and special vertices u_j $(j = 1, \ldots, 2q+2)$, where u_j belongs to G_{j-1} and G_j.

Method:

for $i = 0$ **to** q **do**
/* define an interval graph */
set $\mathcal{X} := \{I_x \in \mathcal{I} : L(x) \leq L(S_i)\}$;
on intervals $\mathcal{X}$ define an interval graph G_{2i};
let I^* be an interval from $\mathcal{X}$ with maximum $R(\cdot)$ value;
set u_{2i+1} := a vertex of G corresponding to I^*;
set $\mathcal{I} := \mathcal{I} \setminus (\mathcal{X} \setminus \{I^*\})$;

/* define a superconnected probe interval graph */
set $\mathcal{Y} := \{I_y \in \mathcal{J} : I_y \subseteq S_i\}$;
set $\mathcal{X} := \{I_x \in \mathcal{I} : L(x) \leq R(S_i)\}$;
define a probe interval graph G_{2i+1} with probes $\mathcal{X}$ and nonprobes $\mathcal{Y}$;
let I^* be an interval from $\mathcal{X}$ with maximum $R(\cdot)$ value;
set u_{2i+2} := a vertex of G corresponding to I^*;
set $\mathcal{I} := \mathcal{I} \setminus (\mathcal{X} \setminus \{I^*\})$;
define on $\mathcal{I}$ an interval graph G_{2q+2}.

Lemma 7. *For any $i = 2, \ldots, 2q+2$, $R(u_i) \geq R(u_{i-1})$ holds.*

Now, for an interval graph G_0 (if it is not empty), we can construct a tree 3-spanner $T_0 = T_0(u_0)$ rooted at any vertex u_0 of G_0. For an interval graph G_{2i} $(i = 1, \ldots, q+1)$, we can construct a tree 3-spanner $T_{2i} = T_{2i}(u_{2i})$ rooted at vertex u_{2i} (see PROCEDURE DECOMP). Since all those trees are shortest path trees, the neighborhoods of vertex u_{2i} in G_{2i} and T_{2i} coincide.

Let G^-_{2i+1} be an interval bigraph obtained from a superconnected probe interval graph G_{2i+1} by ignoring all edges between probes and deleting all probes I_v such that $I_v \subset I_{u_{2i+1}}$.

Lemma 8. *For any $i = 0, \ldots, q$, vertex u_{2i+1} has a maximum neighbor in G^-_{2i+1}.*

Let $T^-_{2i+1} = T^-_{2i+1}(u_{2i+1})$ be a tree 3-spanner of an interval bigraph G^-_{2i+1} constructed starting at vertex u_{2i+1}, $i \in \{0, \ldots, q\}$ (see PROCEDURE SPANATEG). Clearly, the neighborhoods of vertex u_{2i+1} in G^-_{2i+1} and T^-_{2i+1} coincide. We can extend tree T^-_{2i+1} to a spanning tree $T_{2i+1} = T_{2i+1}(u_{2i+1})$ of G_{2i+1} by adding, for each probe I_v of G_{2i+1} such that $I_v \subset I_{u_{2i+1}}$, a pendant vertex v adjacent to u_{2i+1}.

Lemma 9. *$T_{2i+1}(u_{2i+1})$ is a tree 4-spanner for G_{2i+1}, $i \in \{0, \ldots, q\}$. Moreover, for any edge $\{w, u_{2i+1}\}$ of G_{2i+1}, $d_{T_{2i+1}}(w, u_{2i+1}) \leq 2$ holds.*

Now we are ready to construct a spanning tree T for the original probe interval graph $G = (P, N, E)$. We say that a vertex v of G *dominates* a subgraph G_k of G if every vertex of G_k, different from v, is adjacent to v in G.

PROCEDURE SPAN-PIG. Tree 7-spanner for probe interval graphs
Input: A probe interval graph $G = (P, N, E)$, its interval representation $(\mathcal{I}, \mathcal{J})$ and a decomposition of G into graphs $G_0, G_1, \ldots, G_{2q+2}$.
Output: A spanning tree $T = (P \cup N, E')$ of G.
Method:
set $E' = \emptyset$ and $k := 0$;
while $k \leq 2q+2$ **do**
  if there is an index j such that $k \leq j$ and u_k dominates G_j **then do**
    find the largest index j with that property;
    for each v in $G_k \cup \ldots \cup G_j$ $(v \neq u_k)$ add edge $\{v, u_k\}$ to E';
    set $k := j + 1$;
  else do

if k is even **then do**
find a tree 3-spanner $T_k(u_k)$ of an interval graph G_k;
add all edges of $T_k(u_k)$ to E';
if k is odd **then do**
find a tree 4-spanner $T_k(u_k)$ of a superconnected probe interval graph G_k;
add all edges of $T_k(u_k)$ to E';
set $k := k + 1$.

It is easy to see that the tree T constructed by PROCEDURE SPAN-PIG is a spanning tree of G and its construction takes only linear time.

Lemma 10. *If for graph G_k ($k \in \{0, \ldots, 2q+2\}$) there exists a vertex $u_i \in \{u_0, \ldots, u_k\}$ which dominates G_k, then there is a vertex $u_s \in \{u_0, \ldots, u_k\}$ such that $d_T(x, u_s) \leq 1$ holds for any x in G_k. Otherwise, if such vertex u_i does not exist, then for any vertices x, y of G_k, $d_T(x, y) = d_{T_k}(x, y)$ holds.*

Corollary 3. *For any vertices x, y of G_k ($k \in \{0, \ldots, 2q+2\}$), $d_T(x, y) \leq \max\{2, d_{T_k}(x, y)\}$ holds.*

Lemma 11. *T is a tree 7-spanner for G.*

Theorem 5. *Any probe interval graph G admits a tree 7-spanner. Moreover, such a tree 7-spanner can be constructed in $O(m \log n)$ time, or in $O(m+n \log n)$ time if the intersection model of G is given in advance.*

Now let $G = (P, N, E)$ be an enhanced probe interval graph with probes P and nonprobes N.

Corollary 4. *Any enhanced probe interval graph $G = (P, N, E)$ admits a tree 7-spanner. Moreover, such a tree spanner can be constructed in $O(m \log n)$ time.*

6 Concluding Remarks

In the paper, we have shown that the tree t-spanner problem is NP-complete even for chordal bipartite graphs for $t \geq 5$. The complexity of the tree 3-spanner problem is still open. We have also shown that every (enhanced) probe interval graph has a tree 7-spanner. However, it is also open whether the graph classes are tree t-spanner admissible for smaller t.

Acknowledgements. The authors are greatful to an anonymous referee for simplifying the reduction in Section 3.

References

1. B. Awerbuch, A. Baratz, and D. Peleg. Efficient broadcast and light-weighted spanners. manuscript, 1992.
2. H.-J. Bandelt and A. Dress. Reconstructing the Shape of a Tree from Observed Dissimilarity Data. *Advances in Applied Mathematics*, 7:309–343, 1986.
3. A. Brandstädt, V. Chepoi, and F. Dragan. Distance Approximating Trees for Chordal and Dually Chordal Graphs. *J. of Algorithms*, 30(1):166–184, 1999.
4. A. Brandstädt, V.D. Chepoi, and F.F. Dragan. The Algorithmic Use of Hypertree Structure and Maximum Neighbourhood Orderings. *Disc. Appl. Math.*, 82:43–77, 1998.
5. A. Brandstädt, F. Dragan, V. Chepoi, and V. Voloshin. Dually Chordal Graphs. *SIAM J. Disc. Math.*, 11(3):437–455, 1998.
6. A. Brandstädt, F.F. Dragan, H.-O. Le, and V.B. Le. Tree Spanners on Chordal Graphs: Complexity, Algorithms, Open Problems. In *ISAAC 2002*, pages 163–174. LNCS Vol. 2518, Springer-Verlag, 2002.
7. A. Brandstädt, V.B. Le, and J.P. Spinrad. *Graph Classes: A Survey*. SIAM, 1999.
8. L. Cai and D.G. Corneil. Tree Spanners: an Overview. *Congressus Numerantium*, 88:65–76, 1992.
9. L. Cai and D.G. Corneil. Tree Spanners. *SIAM J. Disc. Math.*, 8(3):359–387, 1995.
10. F.F. Dragan and V.I. Voloshin. Incidence Graphs of Biacyclic Hypergraphs. *Disc. Appl. Math.*, 68:259–266, 1996.
11. M.R. Garey and D.S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness*. Freeman, 1979.
12. M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.
13. M.C. Golumbic and C.F. Goss. Perfect Elimination and Chordal Bipartite Graphs. *J. of Graph Theory*, 2:155–163, 1978.
14. F. Harary, J.A. Kabell, and F.R. McMorris. Bipartite intersection graphs. *Comment. Math. Univ. Carolin.*, 23:739–745, 1982.
15. J.L. Johnson and J.P. Spinrad. A Polynomial Time Recognition Algorithm for Probe Interval Graphs. In *Proc. 12th SODA*, pages 477–486. ACM, 2001.
16. D. König. *Theorie der endlichen und unendlichen Graphen (in German)*. Akademische Verlagsgesellschaft, 1936.
17. H.-O. Le and V.B Le. Optimal Tree 3-Spanners in Directed Path Graphs. *Networks*, 34:81–87, 1999.
18. M.S. Madanlal, G. Venkatesan, and C. P. Rangan. Tree 3-Spanners on Interval, Permutation and Regular Bipartite Graphs. *IPL*, 59:97–102, 1996.
19. R.M. McConnell and J.P. Spinrad. Construction of Probe Interval Models. In *Proc. 13th SODA*, pages 866–875. ACM, 2002.
20. T.A. McKee and F.R. McMorris. *Topics in Intersection Graph Theory*. SIAM, 1999.
21. F.R. McMorris, C. Wang, and P. Zhang. On Probe Interval Graphs. *Disc. Appl. Math.*, 88:315–324, 1998.
22. H. Müller. Recognizing Interval Digraphs and Interval Bigraphs in Polynomial Time. *Disc. Appl. Math.*, 78:189–205, 1997. Erratum is available at `http://www.comp.leeds.ac.uk/hm/pub/node1.html`.
23. D. Peleg. *Distributed Computing: A Locally-Sensitive Approach*. Monographs on Discrete Mathematics and Applications. SIAM, 2000.

24. D. Peleg and A.A. Schäffer. Graph Spanners. *J. of Graph Theory*, 13(1):99–116, 1989.
25. E. Prisner. Distance approximating spanning trees. In *Proc. of STACS'97*, pages 499–510. LNCS Vol. 1200, Springer-Verlag, 1997.
26. J. Soares. Graph Spanners: a Survey. *Congress Numerantium*, 89:225–238, 1992.
27. G. Venkatesan, U. Rotics, M.S. Madanlal, J.A. Makowsy, and C.P. Rangan. Restrictions of Minimum Spanner Problems. *Inf. and Comp.*, 136:143–164, 1997.
28. P. Zhang. Probe Interval Graphs and Its Applications to Physical Mapping of DNA. manuscript, 1994.
29. P. Zhang. United States Patent. Method of Mapping DNA Fragments. [Online] Available `http://www.cc.columbia.edu/cu/cie/techlists/patents/5667970.htm`, July 3 2000.