Restrictions of Minimum Spanner Problems

G. Venkatesan[1] U. Rotics[2] M.S. Madanlal[*1]
J.A. Makowsy[**2] C. Pandu Rangan[1]

[1] Department of Computer Science and Engineering
Indian Institute of Technology
Madras-600 036, INDIA.

[2] Department of Computer Science
Technion–Israel Institute of Technology
32000 Haifa, ISRAEL.

Abstract

A t-spanner of a graph G is a spanning subgraph H such that the distance between any two vertices in H is at most t times their distance in G. Spanners arise in the context of approximating the original graph by a sparse subgraph [23]. The MINIMUM t-SPANNER problem seeks to find a t-spanner with the minimum number of edges for the given graph. In this paper, we completely settle the complexity status of this problem for various values of t, on Chordal graphs, Split graphs, Bipartite graphs and Convex Bipartite graphs. Our results settle an open question raised in [7] and also greatly simplify some of the proofs presented in [7, 8]. We also give a factor two approximation algorithm for the MINIMUM 2-SPANNER problem on interval graphs. Finally, we provide approximation algorithms for the bandwidth minimization problem on Convex Bipartite graphs and Split graphs using the notion of tree spanners.

*Current Address: Department of Computer Science, Stanford University, Stanford, California.
**Partially Supported by German Israeli Foundation

1 Introduction and Motivation

We denote the vertex set and edge set of a graph G by $V(G)$ and $E(G)$ respectively. A t-spanner of a graph G is a spanning subgraph S in which the distance between every pair of vertices is at most t times their distance in G. We refer to t and $|E(S)|$ as the *stretch factor* and *size* of the spanner S. The MINIMUM t-SPANNER problem asks for a t-spanner S with fewest possible edges for a fixed t, such an S is called a minimum t-spanner of G and we denote by $s_t(G)$ the size of a minimum t-spanner of G. Clearly if G is connected, $s_t(G) \geq |V(G)| - 1$ with equality holding if and only if G admits a tree t-spanner [6].

The decision version of the MINIMUM t-SPANNER problem consists of a graph $G = (V, E)$ and a goal $K \geq 0$ and the question is whether G has a t-spanner with K or fewer edges i.e whether $s_t(G) \leq K$. The decision versions of all problems considered in this paper are in **NP**; we only prove **NP**-hardness of the optimization versions of the problems – our proofs can be trivially modified to prove **NP**-completeness of the decision versions. In what follows, all graphs we deal with are finite, simple, undirected and connected.

Motivation

The notion of t-spanners was introduced by Peleg and Ullman [24] in connection with the design of synchronizers. The synchronizer is a simulation technology introduced by Awerbuch [3] which enables the execution of a synchronous algorithm on an asynchronous network. The t-spanner is the underlying graph structure of the synchronizer, and the stretch factor and size of the t-spanner are closely related to the time and communication complexities of the synchronizer respectively. Spanners also have applications in planning efficient routing schemes while maintaining succinct routing tables [25]. Spanners also arise in computational geometry in the study of approximation of complete Euclidean graphs [10] and in computational biology in the process of reconstruction of phylogenetic trees [4].

The study of graph spanners has been very active in the last few years. Most of the work has been focused on finding spanners with few edges, small degree, light weights or small stretch factors or on finding optimal spanners on restricted classes of graphs [1, 6, 9, 17, 18, 19, 21, 23].

The key ideas behind the notion of spanners is to approximate the pairwise distances in the original graph by a *sparse* spanning subgraph. Therefore, one of the fundamental problems in the study of spanners is to find a minimum t-spanner of a graph where $t \geq 1$ is a fixed integer (stretch factor). Finding a minimum 2-spanner is **NP**-hard for general graphs [23]. Later, Cai [7] proved that minimum t-spanner is **NP**-hard for each $t \geq 3$ using a rather complicated reduction from 3SAT. MINIMUM t-SPANNER, for each fixed $t \geq 2$, remains **NP**-hard on graphs with maximum degree equal to nine [8]. Because of these intractability results, the complexity of MINIMUM t-SPANNER for useful subclasses of graphs becomes interesting. Linear algorithms exist for the MINIMUM t-SPANNER problem on interval and permutation graphs for each fixed $t \geq 3$ [20], however the case $t = 2$ remains open for both these classes of graphs. Also, a linear algorithm for the MINIMUM 2-SPANNER problem on graphs with bounded degree ≤ 4 is presented in [8].

Discussion of main results

In this paper we study the MINIMUM t–SPANNER problem for chordal, bipartite and split graphs. We also consider interval graphs (which are a subclass of chordal graphs), convex bipartite graphs and graphs of bounded degrees. We also show how to use the existence of optimal t–spanners with special properties to find approximate solutions to the bandwidth minimization problem for split graphs, permutation graphs and convex bipartite graphs.

An interesting outcome of our study consists of the fact that the difficulty of finding optimal 2–spanners may radically differ from the difficulty of finding optimal t–spanners for $t \geq 3$.

Chordal and interval graphs

We show in this paper that for **chordal** graphs the problem is **NP**-hard for $t = 2$ and $t \geq 3$. As corollary, this also holds for perfect graphs. For $t = 2$ this is new. A polynomial approximation algorithm was given for $t = 2, 3, 5$ in [23], however the exact complexity was still considered open in [7].

For its subclass of **interval** graphs it was shown in [20, 27] that the problem is polynomial for $t \geq 3$. For $t = 2$ the exact complexity of the problem remains open, but we show here a polynomial 2–approximation algorithm. Note that also for *permutation* graphs the exact complexity of finding optimal 2–spanners remains open, whereas for $t \geq 3$ the problem is polynomial, [20].

Split graphs

In the case of *split* graphs we show in this paper that the case of $t = 2$ is **NP**-hard whereas the case $t \geq 3$ is polynomial. The reason for this is that for $t \geq 3$ there is an optimal t-spanner which is a tree. As a tree is not necessarily a 2-spanner, the case for $t = 2$ is different, and, in fact, the proof of **NP**-hardness for chordal graphs can be adapted to this case.

Bipartite graphs

In contrast to split graphs, for bipartite graphs the case $t = 2$ is trivially polynomial as there are no edges which can be replaced by a path of length 2. But the case $t \geq 3$ is **NP**-hard, [7]. In this paper we present a new and simpler proof of this.

We also look at a fairly large subclass of bipartite graphs, namely the *convex* bipartite graphs. Here we prove that finding optimal t–spanners is polynomial for every $t \geq 2$.

Bounded degree graphs

Finally we look at graphs of bounded degree with bound k. In [9] it is shown that for $k \geq 9$ finding optimal t-spanners is **NP**-hard for $t \geq 2$. Our techniques used for the case of chordal graphs can be used to simplify the proof of this result. In [9] it is also shown that for $k \leq 4$ the 2–spanner problem is polynomial. For the remaining non–trivial combinations of t and k the exact complexity of the MINIMUM t–SPANNER problem still remains as open.

Organization of the paper

In detail the paper is organized as follows. In section 2 we prove a useful lemma stating that a certain edge covering problem used in the sequel is **NP**-complete. In section 3 we discuss bipartite and convex bipartite graphs. In section 4 we deal with chordal graphs and split graphs. In section 5 we present approximation algorithms for interval graphs and in section 6 we discuss the applications to the bandwidth minimization problem. In section 7 we draw conclusions and present open problems.

Notation

The notations and definitions not stated explicitly may be found in [15]. We use $u \sim v$ to mean that u is adjacent to v. The neighborhood of a vertex v in G is denoted by $N_G(v)$. When the graph being referred to is obvious, we just use $N(v)$. The degree of a vertex v in G is denoted by $deg_G(v)$. We denote by $\delta(G)$ and $\Delta(G)$ the minimum and maximum degree of a vertex in G. $d_G(u,v)$ denotes the distance between vertices u, v in graph G. $\gamma(G)$ denotes the size of a minimum vertex-cover of G, $c_3(G)$ stands for the number of triangles (3-cliques) in G and $\omega(G)$ stands for the size of a maximum clique in G.

Acknowledgments

This paper reports results which were mostly obtained independently by G. Venkatesan, M.S. Madanlal and C. Pandu Rangan in India and U. Rotics and J.A. Makowsky in Israel. [27], which was accepted for publication first, contains the **NP**-completeness for chordal graphs (the result was however proved in a different manner) and the polynomial algorithm for interval graphs, which also appears in [20]. [27] was subsequently replaced by the current paper with five authors.

We would like to thank L. Cai for bringing to our mutual attention the work of the other team and also for providing us other valuable references. The Israeli

team would like to thank also to B. Courcelle, D. Peleg, R. Shamir and K. Simon for pointing out important references and critical comments to [27].

2 A useful NP-complete problem

In this section we prove the **NP**-hardness of a useful problem which we will use for reductions to prove **NP**-hardness later on. Define the $2K_3$-edge cover of a graph G to be a set of edges S of G such that every triangle in G has at least two edges from S. Also let $t_2(G)$ denote the size of a minimum $2K_3$-edge cover of G.

Lemma 2.1 *It is* **NP***-hard to determine* $t_2(G)$ *of a general graph* G.

The above lemma can be proved through a reduction from the vertex-cover problem using some modifications in the idea used in [30] to prove the **NP**-hardness of the following problem: Given a graph $G = (V, E)$, find a set P of minimum number of edges in E such that any triangle in G has at least one edge in P.

However we prove a stronger result constraining the input graph G is degree-bounded. More precisely we prove:

Lemma 2.2 *It is* **NP***-hard to determine* $t_2(G)$ *even for a graph* G *with* $\Delta(G) = 9$.

Proof : The reduction is from the vertex cover problem on cubic graphs [12]. Let $H_1 = (V_1, E_1)$ be a cubic graph with $V_1 = \{1, 2, \ldots, n\}$, $E_1 = \{e_i = (p_i, q_i) : 1 \leq i \leq m\}$.

Form a graph $H_2 = (V_2, E_2)$ as follows: Initially take n independent edges $f_i = (x_i, y_i)$, for $1 \leq i \leq n$ as part of H_2 – these edges correspond to the vertices of H_1. Then for $1 \leq j \leq m$, corresponding to the edge e_j of H_1, add the following configuration shown in Figure 1 (called T_j) between the edges f_{p_j} and f_{q_j} of H_2. Note that a_j, b_j, c_j and u_{jk} (for $1 \leq k \leq 5$) are *new* vertices. Also $|V_2| = 8m + 2n$, $|E_2| = 19m + n$ and $c_3(H_2) = 10m$.

Clearly $\Delta(H_2) \leq 10$. Also if we ensure that, for each i, $1 \leq i \leq n$, if $\{e_{i_1}, e_{i_2}, e_{i_3}\}$ are the three edges of H_1 incident at vertex i, then x_i is adjacent to say c_{i_1} and c_{i_2} and y_i is adjacent to c_{i_3}. This will make sure that $\Delta(H_2) = 9$. Observe also that $\omega(H_2) = 3$.

Now consider the problem of finding a minimum $2K_3$-edge cover S of H_2, $|S| = t_2(H_2)$. Clearly S may be chosen so that, for the configuration of Figure 1, $P_j = \{(a_j, x_{p_j}), (c_j, y_{p_j}), (a_j, b_j), (c_j, x_{q_j}), (b_j, y_{q_j})\} \subset S$. Now in S, at least three more edges are needed from T_j so that the triangles (a_j, b_j, c_j), (a_j, x_{p_j}, y_{p_j}), (a_j, c_j, y_{p_j}), (c_j, b_j, x_{q_j}) and (b_j, x_{q_j}, y_{q_j}) have at least two edges in S and at least two of these edges which are used to *cover* the above five triangles must be from $Q_j = \{(a_j, c_j), (c_j, b_j), (a_j, y_{p_j}), (b_j, x_{q_j})\}$. Thus it is not difficult to see that the optimal way to choose S is that: $P_j \subset S$, $|Q_j \cap S| = 2$

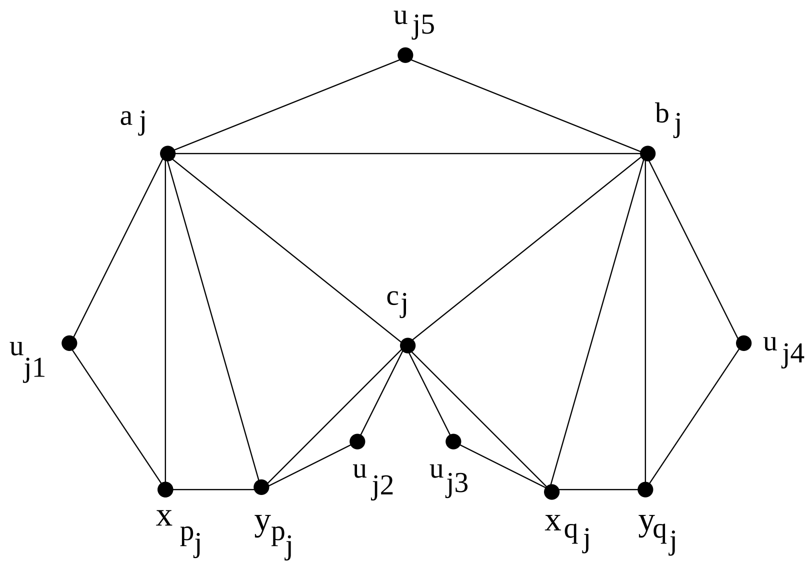

Figure 1: Configuration T_j for an edge (p_j, q_j).

for $1 \leq j \leq m$ and S includes at least one of f_{p_j} and f_{q_j} for $1 \leq j \leq m$. Hence, it follows that:

$$|S| = t_2(H_2) = 12m + \gamma(H_1)$$

where $\gamma(H_1)$ stands for the size of a minimum vertex cover of H_1.

The result now follows since H_2 can be clearly constructed in polynomial time from H_1. □

3 Minimum Spanners on Bipartite Graphs

In this section, we prove that the MINIMUM t-SPANNER problem, for $t \geq 3$ is **NP**-hard on Bipartite graphs. Since a Bipartite graph G has no odd cycle, a subgraph H of G would be a $2k$-spanner of G iff it is $(2k-1)$-spanner of G. Hence if suffices to prove the result for odd $t \geq 3$. Also since the only 1-spanner and 2-spanner of a Bipartite graph G is G itself, the MINIMUM t-SPANNER problem for Bipartite Graphs is in P for $t = 1, 2$. We prove that for $t = 3$, this problem is **NP**-hard. Moreover, we solve this problem on Convex Bipartite Graphs, which is a broad subclass of Bipartite Graphs. Specifically, we show that all Convex Bipartite Graphs have a tree 3-spanner, and such a spanner can be constructed in linear time.

3.1 Minimum Spanners on Bipartite Graphs is NP-hard

Let us first define few terms.

Definition 3.1 *An edge e of G is a t-mandatory edge if the two ends of e are connected by a path Q of length t and all internal vertices of the path have degree 2. The path Q will be called a t-compelling path.*

Definition 3.2 *An induced path P of a graph G is called a t-mandatory path if all its edges are t-mandatory edges in G.*

We now observe the following lemma that allows us to *force* an edge in a t-spanner of a graph G.

Lemma 3.1 *If S is a MINIMUM t-SPANNER of a graph G, then S may be modified without changing its cardinality so that it will include all the t-mandatory edges and exactly $t-1$ edges at each t-compelling path.*

Proof: Follows immediately from the definition of a t-mandatory edge. □

Theorem 3.1 *For any fixed $t \geq 3$, determination of $s_t(G)$ is* **NP**-*hard even for a Bipartite graph G.*

Proof : As mentioned earlier, it suffices to consider odd $t \geq 3$. The reduction is from the edge domination problem on Bipartite graphs [29]. Let $H = (X, Y, E)$ be a Bipartite graph with $X = \{x_1, x_2, \ldots, x_r\}$, $Y = \{y_1, y_2, \ldots, y_s\}$ and $|E| = m$. Let $\gamma_e(H)$ denote the size of a minimum edge dominating set of H $(MED(H))$.

Construct a graph $G = (V', E')$ as follows: G will initially have H as an induced subgraph. Add two vertices z_X, z_Y, and connect z_X (z_Y) to all vertices in X (Y) by distinct t-mandatory paths of length $(t-1)/2$ (by distinct we mean that the internal vertices of all such paths are distinct). Denote by P_i (Q_j) the t-mandatory path between z_X and x_i (z_Y and y_j). See Figure 2 that illustrates the above construction for $t = 3$.

Since t is odd and H is Bipartite, it is easy to see that G has no odd cycles and is also therefore Bipartite. Also G has $2 + (r+s)t(t-1)/2$ vertices and $(r+s)(t^2-1)/2 + m$ edges.

Let S be a minimum t-spanner of G. By lemma 3.1, we may assume that S is a minimum t-spanner of G that includes all the t-mandatory edges of G and exactly $t-1$ edges from each t-compelling path of G. Hence S will have $s_t(G)$ edges and $E(P_i) \subseteq E(S)$, $E(Q_j) \subseteq E(S)$ for $1 \leq i \leq r$, $1 \leq j \leq s$. So any two x_i's (or y_i's) will certainly be connected by a path of length $t-1$ in S and a necessary and sufficient condition to further ensure that S is a t-spanner is that $E(S) \cap E$ is an edge dominating set of H. Thus we have:

$$s_t(G) = \frac{(r+s)t(t-1)}{2} + \gamma_e(H)$$

and the result follows since G can be constructed in polynomial time from H. □

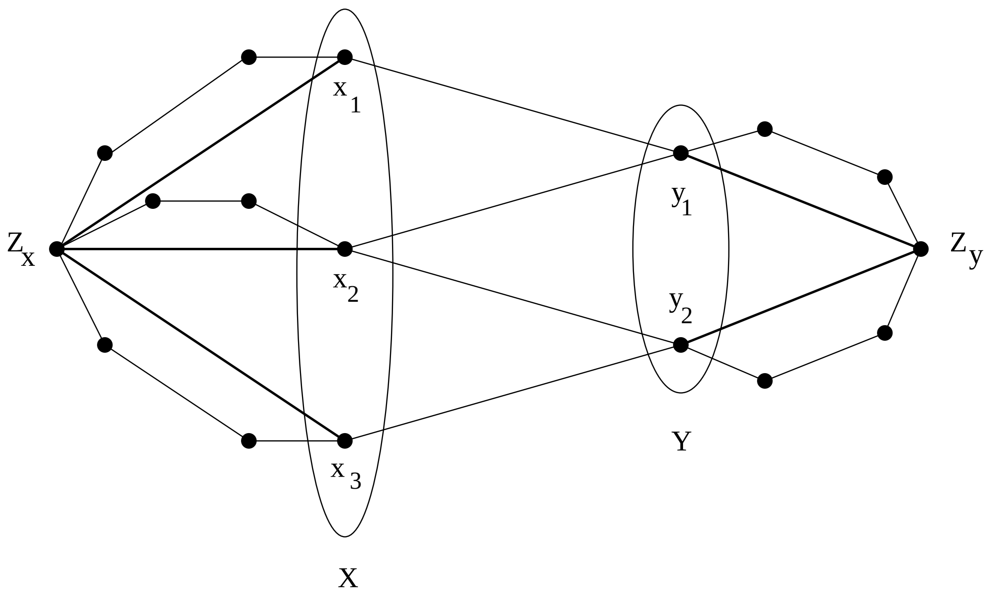

Figure 2: Construction of a Bipartite Graph.

3.2 Minimum Spanners on Convex Bipartite Graphs

As the MINIMUM t-SPANNER problem is **NP**-hard on Bipartite Graphs, the problem of determining the largest subclass of Bipartite Graphs, which admits a polynomial solution to MINIMUM t-SPANNER problem becomes interesting. We have solved this problem on Convex Bipartite Graphs. We show that all Convex Bipartite Graphs have a tree 3-spanner and such a tree can be constructed in linear time. So, in effect we have solved the MINIMUM t-SPANNER problem on Convex Bipartite Graphs (for $t \geq 3$) in linear time.

A Bipartite graph $G = (X, Y, E)$ is said to be convex if a linear ordering "$\leq$" of the elements of Y can be found so that for any x in X and distinct y_1 and y_2 in Y, with $y_1 \leq y_2$

$$x \sim y_1 \ , x \sim y_2 \Rightarrow x \sim y, \ \forall y \in Y, y_1 \leq y \leq y_2$$

In other words, G is convex iff elements of Y can be ordered such that for any $x \in X$, the set of vertices of Y adjacent to x forms an interval in this ordering.

Given the adjacency matrix A, of a Bipartite graph, it is convex if and only if A has consecutive 1's property. So, the recognition algorithm is straight forward and is linear [5]. From now on, we assume that the ordering of the vertices of Y is available. We also label the vertices of Y as $y_1, y_2, \ldots, y_{n_Y}$, where $n_Y = |Y|$, such that $y_i < y_j$ iff $i < j$. Moreover, for every vertex $x \in X$, we assume that $L(x), R(x)$ are available where $L(x)$ is the least vertex in Y adjacent to x and

$R(x)$ is the greatest vertex adjacent to x. Note that $L(x), R(x)$ for all $x \in X$ can be obtained from the adjacency list representation of the graph in linear time.

Now, given a *connected* Convex Bipartite graph G, we find a tree 3-spanner T. The basic idea is to define a sequence of vertices of Y, $\alpha_1, \alpha_2, \ldots$, such that there exists vertices $d_1, d_2, \ldots$ in X such that $d_i \sim \alpha_i$ and $d_i \sim \alpha_{i+1}$ for all i. The vertices α_i and d_i are defined recursively as follows:

Definition 3.3
$\alpha_1 = y_1$
For $i \geq 1$

$$\begin{aligned} S_i &= \{x | x \in N(\alpha_i) \textit{ such that } R(x) \textit{ is maximum}\} \\ d_i &= \textit{Any arbitrary element of } S_i \\ \alpha_{i+1} &= R(d_i) \end{aligned}$$

Note that α_{i+1} is the largest vertex that can be reached from α_i by a path of length two. It is easy to see that if G is connected, then there is a k such that $\alpha_k = y_{n_Y}$. Before we proceed with the construction of the tree, we define an imaginary vertex α_0 which is less than $\alpha_1 = y_1$. This is just for mathematical strictness in our proofs.

Lemma 3.2 *For all* $v \in X$*, there exists an* $i \geq 0$ *such that*

$$[L(v), R(v)] \subset (\alpha_i, \alpha_{i+2}]$$

Proof: Suppose $L(v) \in (\alpha_i, \alpha_{i+1}]$ for an $i \geq 0$. If $R(v) < \alpha_{i+1}$, then we have nothing to prove. Suppose $R(v) \geq \alpha_{i+1}$, then by convex Bipartite ordering $v \sim \alpha_{i+1}$. By definition of α_{i+2}, $R(v) \leq R(d_{i+1}) = \alpha_{i+2}$. Hence the lemma. □

This lemma shows that for any $v \in X$, the neighbors of v are restricted to span at most two consecutive intervals of the α sequence. This property is sufficient to construct a tree 3-spanner.

Let $y_1 = \alpha_1, \alpha_2, \ldots, \alpha_k = y_{n_Y}$ be the α sequence for a Convex Bipartite graph G. We construct a tree T as follows.

Algorithm Convex-Bip-Spanner

1. $T := \phi$;
2. For $i = 1$ to $k - 1$ do
   - Add edges (d_i, v) to T, $\forall v \in (\alpha_i, \alpha_{i+1}]$;
3. For $i = 1$ to k do
   - Add edges (α_i, v) to T, $\forall v \in N(\alpha_i)$ and $L(v) \in (\alpha_{i-1}, \alpha_i]$;

4. If there is a vertex $v \in X$ such that no edge adjacent to v is added in Steps 1,2 then add $(v, L(v))$ to T;

end.

Theorem 3.2 *All Convex Bipartite Graphs are tree 3-spanner admissible and given such a graph, its tree 3-spanner can be constructed in linear time.*

Proof: The tree T constructed by *Convex-Bip-Spanner* is obviously a spanning tree, as each edge added includes a new vertex to the graph. The proof proceeds by showing that for all $v \in X$ and for all edges (v, u), $d_T(u, v) = 1$ or 3.

Let v be any vertex in X. Let $L(v) \in (\alpha_i, \alpha_{i+1}]$. By Lemma 3.2, $R(v) \leq \alpha_{i+2}$. Obviously, if $v = d_{i+1}$ then $d(v, u) = 1$ *or* 3, $\forall u \sim v$. Otherwise, consider the following cases:

Case 1: Suppose $R(v) \geq \alpha_{i+1}$.

Then by convex Bipartite ordering $v \in N(\alpha_{i+1})$. As $L(v) \in (\alpha_i, \alpha_{i+1}]$, the edge (α_{i+1}, v) is added to T in Step 3 of the algorithm. Take any edge (v, u).

If $u \in [L(v), \alpha_{i+1})$, then v, α_{i+1}, d_i, u is a path of length three in T. So, $d_T(v, u) = 3$.

If $u = \alpha_{i+1}$, then obviously $d_T(v, u) = 1$.

If $u \in (\alpha_{i+1}, R(v)]$, then $v, \alpha_{i+1}, d_{i+1}, u$ is a path of length three in T. So, $d_T(v, u) = 3$.

Case 2: Suppose $R(v) < \alpha_{i+1}$.

ie. $[L(v), R(v)] \subset (\alpha_i, \alpha_{i+1})$. So v is not spanned after Steps 1,2, and $(v, L(v))$ is added in Step 3. For any $u \sim v$ and $u \neq L(v)$, $u, d_i, L(v), v$ is a path of length three in T.

Thus we have exhausted all possible cases and so T constructed by *Algorithm Convex-Bip-Spanner* is indeed a tree 3-spanner. The complexity of this algorithm is linear if we have the adjacency list representation of the Convex Bipartite graph. □

We now state the following result that completely settles the complexity status of the MINIMUM t-SPANNER problem on Convex Bipartite graphs:

Theorem 3.3 *The MINIMUM t-SPANNER is solvable in linear time on convex Bipartite graphs for all values of* $t \geq 1$.

Proof: Let G be a convex Bipartite graph. Since G has no odd cycles, the only t-spanner of G for $t = 1, 2$ is G itself. Also for $t \geq 3$, a tree 3-spanner that can be obtained in linear time by Theorem 3.2 will serve as a MINIMUM t-SPANNER of G. The result now follows. □

4 Minimum Spanners on Chordal Graphs

We now obtain **NP**-completeness results on Chordal graphs. However, here we cannot force the choice of an edge in a t-spanner by simply adding a path

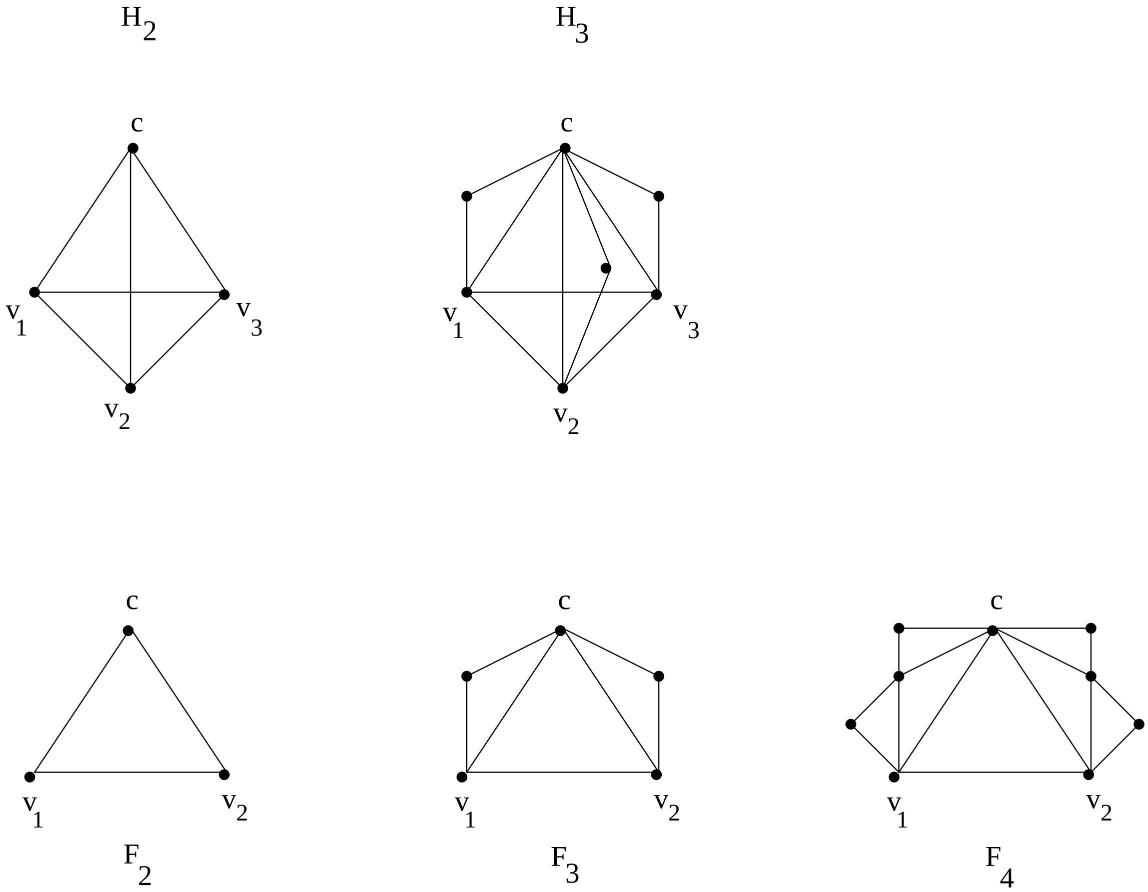

Figure 3: Construction of H_i and F_i.

of length t between its two ends as then we would have an induced cycle - a forbidden subgraph of Chordal graphs.

4.1 A useful sequence of graphs

We now define two sequences of graphs: $H_2, H_3, \cdots$ and $F_2, F_3, \cdots$ which will be used in the reductions employed in the **NP**-completeness proofs of this section.

Let H_2 be K_4, (the complete graph on 4 vertices), with a triangle (v_1, v_2, v_3) designated as the *bond triangle* and let the other three edges (c, v_i), $1 \leq i \leq 3$ be called *outer* edges. Let F_2 be K_3, (the complete graph on 3 vertices), with an edge (v_1, v_2) designated as the *bond edge* and the other two edges (c, v_1) and (c, v_2) be the *outer* edges.

Continue the sequence as follows:
Form H_{n+1} from H_n by bonding a new K_3 at each outer edge. The *bond* triangle of H_{n+1} is the same as that of H_n and the *outer* edges of H_{n+1} are the edges that are newly introduced while bonding the triangles. The newly introduced vertices become the outer vertices of H_{n+1}.
F_{n+1} is obtained from F_n in a similar way by bonding a new K_3 at each outer edge (here bond edges remain the same). See Figure 3 for an illustration of this procedure.
We observe the following:

Fact 1 *For $n \geq 2$, H_n has $3.2^{n-2}+1$ vertices and 3.2^{n-1} edges of which 3.2^{n-2} are outer edges. Moreover, H_n has 3.2^{n-3} outer vertices for $n \geq 3$.*

Fact 2 *For $n \geq 2$, F_n has $2^{n-1}+1$ vertices and $2^n - 1$ edges of which 2^{n-1} are outer edges. Moreover F_n has 2^{n-2} outer vertices for $n \geq 3$.*

For $t \geq 2$, denote by $B_e(H_t)$ ($B_v(H_t)$) the edges (vertices) of the bond triangle of H_t. Similarly $B_e(F_t)$ and $B_v(F_t)$ are defined (with bond triangle replaced by bond edge)

Lemma 4.1 *Let $G = (V, E)$ be a graph and let $H = (V', E')$ be obtained by identifying the bond edge of a* new *copy of F_t ($t \geq 2$) with an edge $e \in E$. Let S be a subgraph of H with minimum number of edges such that $d_S(x, y) \leq t d_H(x, y)$, $\forall x, y \in V(F_t)$. Then:*

1. *S has 2^{t-1} edges.*
2. *S has at least $2^{t-1} - 1$ edges from $E(F_t) - \{e\}$.*
3. *S may be chosen so that $e \in E(S)$.*

Proof : Proof proceeds by induction on t. All the three statements are clearly true for $t = 2$. For $t > 2$, S must have at least one edge incident with each outer vertex of F_t and so S must contain at least 2^{t-2} outer edges of the bonded F_t. Also if S contains both the edges incident at an outer vertex u of the bonded F_t, then one of the edges may be replaced by the third edge (not outer w.r.t F_t) of the unique triangle in F_t containing u. Thus any such S may be chosen to include a set Q of exactly 2^{t-2} outer edges of F_t.

If $O(F_t)$ denotes the set of outer vertices of F_t, then by our construction of $\{F_t\}_{t\geq 2}$, $F_t - O(F_t) \cong F_{t-1}$. Hence it is easy to see that S will satisfy the conditions of the lemma iff $S - O(F_t)$ satisfies the conditions of the lemma with t replaced by $t - 1$ everywhere in the lemma.

Hence $|E(S)| = 2^{t-2} + |E(S - O(F_t))| = 2^{t-2} + 2^{(t-1)-1} = 2^{t-1}$ by the induction hypothesis proving 1. Also by the induction hypothesis, $S - O(F_t)$ must contain at least $2^{t-2} - 1$ edges from $E(F_{t-1}) - e$ and hence

$$|E(S) \cap (E(F_t) - e)| = 2^{t-2} + |E(S - O(F_t)) \cap (E(F_t) - e)| \geq 2^{t-2} + (2^{t-2} - 1) = 2^{t-1} - 1$$

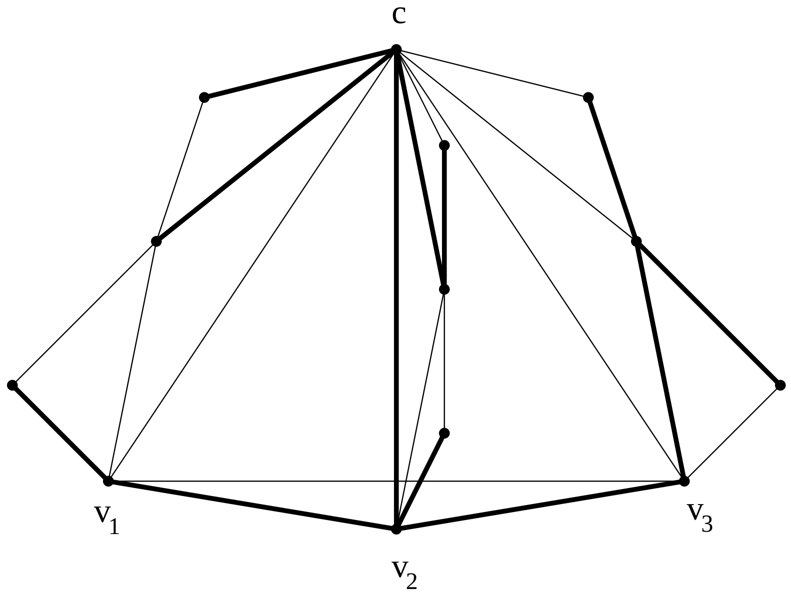

Figure 4: Spanner for $t = 4$.

proving 2. Part 3 follows from the fact that by induction hypothesis $S - O(F_t)$ may be chosen so that $e \in S - O(F_t)$. Note that part 3 implies that bonding of F_t at an edge e serves the purpose of *forcing* the edge in a t-spanner of the graph. □

Lemma 4.2 *Let $G = (V, E)$ be a graph and let $H = (V', E')$ be obtained by identifying the bond triangle of a* new *copy of H_t ($t \geq 2$) with a triangle T of G. Let S be a subgraph of H with minimum number of edges such that $d_S(x, y) \leq t d_H(x, y)$, $\forall x, y \in V(H_t)$. Then:*

1. *S has at least $3 \times 2^{t-2} - 2$ edges from $E(H_t) - B_e(H_t)$.*
2. *S has $3 \times 2^{t-2}$ edges.*
3. *S may be chosen so that it includes two edges of the triangle T.*

Proof : The proof proceeds along the same lines as that of Lemma 4.1. See Figure 4 for an illustration for the case $t = 4$ (edges present in S are marked bold). Note that this "bonding" has the effect of *forcing* the choice of at least two edges of T in a minimum spanner. □

4.2 NP-completeness on Chordal Graphs

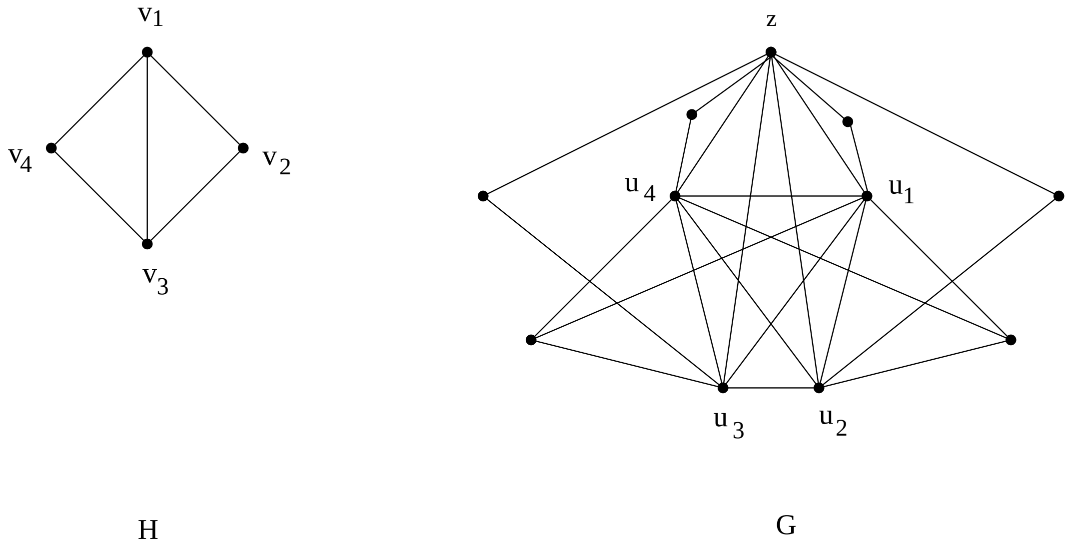

Figure 5: The Graph G for the case $t = 2$.

Theorem 4.1 *The MINIMUM t-SPANNER problem is* **NP**-*hard on Chordal graphs for all fixed* $t \geq 2$.

Proof : The reduction is from the problem of determining the size of a minimum $2K_3$-edge cover($t_2(H)$) of a general graph H, proved **NP**-hard in Lemma 2.2.

Let $H = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$. Form a graph G as follows:

Initially take a K_{n+1} containing vertices $\{z, u_1, u_2, \ldots, u_n\}$ where z is a special vertex and u_i corresponds to v_i for $1 \leq i \leq n$. For each edge $e_i = (z, u_i)$, $1 \leq i \leq n$, bond a *new* copy of F_t (using its bond edge) at e_i. Finally, for each triangle $(v_{i_1}, v_{i_2}, v_{i_3})$ in H, bond a *new* copy of H_t (using its bond triangle) at the triangle $(u_{i_1}, u_{i_2}, u_{i_3})$. The resulting graph is our desired G. See Figure 5 for an illustration of this construction for the case $t = 2$.

Since F_t and H_t are Chordal (by our construction) and all bonding operations are at either 2-cliques or 3-cliques it is straightforward to see that G is Chordal.

Let S be a minimum t-spanner of G. By Lemmas 4.1 and 4.2, it follows that S may be chosen to include each edge (z, u_i) for $1 \leq i \leq n$ (thus providing a path of length two between u_i and u_j for $1 \leq i < j \leq n$) and at least two edges from each triangle (u_i, u_j, u_k) $(1 \leq i < j < k \leq n)$ in G whenever (v_i, v_j, v_k) induces a K_3 in H. Since S is a minimum t-spanner of G, it will include exactly $t_2(H)$ edges of the form (u_p, u_q) $(1 \leq p < q \leq n)$ and using Lemmas 4.1 and 4.2 it follows that:

$$s_t(G) = |E(S)| = (2^{t-1} - 1)n + (3 \times 2^{t-2} - 2)c_3(H) + n + t_2(H)$$

where $c_3(H)$ denotes the number of triangles in H.

Clearly G can be constructed in polynomial time from H for fixed t and hence the theorem is proved. □

Corollary 1 *The MINIMUM 2-SPANNER problem is* **NP***-hard even when restricted to Split graphs.*

Proof : Follows from the fact that the graph G constructed in the proof of Theorem 4.1 will be a Split graph if $t = 2$. □

Lemma 4.3 *The MINIMUM t-SPANNER problem can be solved in linear time on Split graphs for $t \geq 3$.*

Proof : Let $G = (V, E)$ be a connected Split graph, with a partition $V = S \cup K$ where $S = \{x_1, x_2, \ldots, x_p\}$ is an independent set and $K = \{y_1, y_2, \ldots, y_q\}$ is a clique. Since G is connected, for $1 \leq i \leq p$, $\exists f_i$ such that $(x_i, y_{f_i}) \in E$. Construct a tree $T = (V, E')$ with $E' = \{(y_1, y_i) : 2 \leq i \leq q\} \cup \{(x_i, y_{f_i}) : 1 \leq i \leq p\}$. It is easy to verify that T is a tree-3-spanner of G and can be constructed in linear time. Clearly therefore, T is a minimum t-spanner of G for all $t \geq 3$ and the result follows. □

Combining the results of Theorem 4.1, corollary 1 and Lemma 4.3, we now state the following corollaries which completely settle the complexity status of the MINIMUM t-SPANNER problem on Split and Chordal graphs.

Corollary 2 *The MINIMUM t-SPANNER problem on Split graphs is* **NP***-hard if $t = 2$ and is in P otherwise.*

Corollary 3 *The MINIMUM t-SPANNER problem on Chordal graphs is in P for $t = 1$ and is* **NP***-hard for all integers $t > 1$.*

As every chordal graph is perfect we get

Corollary 4 *The MINIMUM t-SPANNER problem on perfect graphs is* **NP***-hard for all integers $t \geq 2$.*

For $t \geq 3$ this also follows from the **NP**-hardness of the MINIMUM t-SPANNER problem for bipartite graphs, [7], cf. also theorem 3.1. For $t = 2$ the result of the corollary is new.

Remark: Using a similar approach and by bonding "suitable" graphs at each triangle of the graph H_2 constructed in the reduction of lemma 2.2, one can prove in a much simpler way than that given in [8] that MINIMUM t-SPANNER is **NP**-hard on degree-bounded graphs for each fixed $t \geq 2$.

5 Minimum 2-spanner on Interval Graphs

All connected interval graphs admit a tree 3-spanner that can be computed in linear time [20, 27], hence it follows that the minimum t-spanner of an interval graph can be found in linear time for $t \geq 3$. However, the MINIMUM 2-SPANNER problem on interval graphs is an intriguing open problem and as a first step, we proceed to give a factor 2 approximation algorithm for the same. Note that a factor $O(\log(|E|/|V|))$ approximation algorithm exists for the MINIMUM 2-SPANNER problem on general graphs [17]. Recall that a *clique* of a graph $G = (V, E)$ is a set of vertices $C \subseteq V$ such that for any pair of vertices u, v in C the edge (u, v) is in E. A *maximal clique* of G is a clique C of G, such that for every vertex v of V which is not in C, the set of vertices $C \cup \{v\}$ is not a clique of G.

The following theorem is from [13]:

Theorem 5.1 (Gilmore and Hoffman) *A graph G is an interval graph if and only if the set of vertices of G can be linearly ordered such that, for every vertex x of G, the maximal cliques containing x occur consecutively.*

We now propose the following algorithm for finding a "good" sparse 2-spanner of an interval graph:

<u>ALGORITHM *Approx-Interval-2-spanner*</u>
<u>Input :</u> A connected interval graph $G = (V, E)$ with n vertices.
<u>Output :</u> A set S of edges which will induce a 2-spanner of G.
Begin

1. Find the maximal cliques of G $\{K_i : 1 \leq i \leq l\}$ ordered as in theorem 5.1 above.
2. /* Initialize */ $S = \emptyset, i = 1$.
3. While $(i \leq l)$ do
   (a) Let p = max $\{j : K_i \cap K_j \neq \emptyset\}$. Choose any point in $K_i \cap K_p$ denote it by x_i, and set
   $$S = S \cup \{(x_i, v) : v \in K_i \cup K_{i+1} ... \cup K_p\}$$
   (b) Set i = p + 1.
4. Output: the set S, which is a 2-spanner of G.

End.

Fact 3 *The set of edges S constructed by the above algorithm is a 2-spanner of G.*

Proof: Let $(u, v) \in E$, both u and v must be in some maximal clique K_j of G. From the construction of the above algorithm, all the vertices of K_j are connected in S to some vertex x_i chosen in step 2 of the algorithm. Hence $(x_i, v), (x_i, u)$ is a path of length two connecting between u and v in S. $\square$

Fact 4 *The set of edges S constructed by the above algorithm satisfies $|S| \leq 2n-2$.*

Proof: Let $x_1, x_2, ..., x_t$ be the vertices chosen in the iterations of step 2 in the algorithm, and let $R = V - \{x_i : 1 \leq i \leq t\}$. Each edge in S is of the form (v, x_i) for some i: $1 \leq i \leq t$, where $v \in R$. It is easy to see that each vertex v in R is connected in S to at most two vertices, of the form x_i, x_j for some i,j: $1 \leq i, j \leq t$. Hence, $|S| \leq 2 \times |R| \leq 2 \times (n-1)$. $\Box$.

Fact 5 *Any 2-spanner of G must contain at least $n-1$ edges.*

Fact 6 *The above algorithm can be implemented to run in linear $O(|E| + |V|)$ time.*

Proof: By classical results the maximal cliques of an interval graph can be found in linear time. Also, the time spent by the algorithm is bounded by the number of edges in S which is less than $2 \times n - 1$, by fact 4 above. Thus the algorithm can be implemented to run in linear $O(|E| + |V|)$ time. $\Box$
From facts 3-6 above it follows that:

Theorem 5.2 *The above algorithm is a factor two approximation to the MINIMUM 2-SPANNER problem on interval graphs, and it can be implemented to run in linear $O(|E| + |V|)$ time.*

We conclude this section by introducing a variant of the above algorithm, where the max function in step 3 is replaced by an arbitrary function f which satisfies the following conditions:

1. f is a function from $\{1, 2, \ldots, n-1\}$ to $\{2, 3, \ldots, n\}$.
2. For $1 \leq i < n$, it must be that:

$$i < f(i) \leq max\{j : K_i \cap K_j \neq \emptyset\}.$$

<u>ALGORITHM *f*-**SPAN-2**</u>
<u>Input :</u> A connected interval graph $G = (V, E)$ with n vertices and a function f as above.
<u>Output :</u> A set S_f of edges which will induce a 2-spanner of G.
<u>**Begin**</u>

1. Find the maximal cliques of G $\{K_i : 1 \leq i \leq l\}$ ordered as in theorem 5.1 above.

2. /* Initialize */ $S_f = \emptyset, i = 1$.
3. While $(i \leq l)$ do

(a) Let p = f(i), choose any point in $K_i \cap K_p$ denote it by x_i, and set

$$S_f = S_f \cup \{(x_i, v) : v \in K_i \cup K_{i+1} \ldots \cup K_p\}$$

(b) Set i = p + 1.

4. Output: the set S_f, which is a 2-spanner of G.

End.

It is easy to see that for every f as described above algorithm f-SPAN-2 produces a 2-spanner of G and the algorithm runs in linear time.

Problem 1 *Is it true that for every connected interval graph G there is an f_G such that the 2-spanner S_{f_G} produced by algorithm f_G-SPAN-2 above is a minimum 2-spanner for G?*

If the answer to this question is positive, we can use this to produce a polynomial algorithm for the MINIMUM 2-SPANNER problem for interval graphs. This relies on the following observation. Using dynamic programming we can optimize in polynomial time the choice of f as a function of G such that S_f is the smallest 2-spanner obtainable using f-SPAN-2.

6 Bandwidth Approximation Using Tree Spanners

We now use the notion of tree-spanners to find approximate solutions to the bandwidth minimization problem.

Definition 6.1 *A labeling L of G is a 1-1 mapping from V into $\{1, 2, \ldots, |V|\}$. The width of L is defined as*

$$b(G, L) = \max\{|L(u) - L(v)|, (u, v) \in E\}$$

The bandwidth of G is

$$bw(G) = \min\{b(G, L) | L is\ a\ labeling\ of\ G\}$$

A label L such that $b(G, L) = bw(G)$ is called an optimal labeling of G.

The bandwidth minimization problem is in general very tough. The first **NP**-hardness result on general graphs was shown by Papadimitriou [22]. Later, Garey et.al [11] showed that this problem is **NP**-hard even when restricted to trees having maximum degree of three.

A caterpillar is a special kind of tree consisting of a simple chain with "hairs" attached to each vertex. If each hair has a length $\leq k$, it is called as a caterpillar of hair length at most k.

The following is by Assmann et.al [2].

Theorem 6.1 *Bandwidth of caterpillars of hair length at most 2 can be found in $O(n \log n)$ time.*

As Bandwidth Minimization is intractable on many classes of graphs, we can try to find an approximate bandwidth of a given graph. A labeling L is said to be a d-approximation to the bandwidth of a graph G, if $b(G, L) \leq d * bw(G)$. Up to our knowledge, there is no algorithm for approximate bandwidth of general graphs, and even for trees. An approximate algorithm for Asteroidal Triple-Free Graphs was given by Kloks et.al [16]. In this section, we seek to find approximate algorithms for certain classes of graphs using the concept of tree–spanners. Recall that G^d denotes the graph obtained from G by adding an edge between any two vertices which are connected by a path of length at most d in G. A simple lemma on which all our results are based follows:

Lemma 6.1 *Let G, H be graphs such that $G \subseteq H \subseteq G^d$ or $H \subseteq G \subseteq H^d$ for an integer $d \geq 1$, and let L be an optimal labeling for H. Then L is a d-approximation for the bandwidth of G.*

Proof: First consider the case when $G \subseteq H \subseteq G^d$, then

$$\begin{aligned} b(G,L) &\leq b(H,L), \text{ As } G \subseteq H \\ &= bw(H), \text{ As } L \text{ is optimal} \\ &\leq bw(G^d), \text{ As } H \subseteq G^d \\ &\leq d.bw(G) \end{aligned}$$

Now let $H \subseteq G \subseteq H^d$, then

$$\begin{aligned} b(G,L) &\leq b(H^d,L), \text{ As } G \subseteq H^d \\ &\leq d.b(H,L), \\ &= d.bw(H), \text{ As } L \text{ is optimal} \\ &\leq d.bw(G). \text{ As } H \subseteq G \end{aligned}$$

This proves the lemma. □

Corollary 5 *If T is a tree t-spanner for G, $t > 0$, and L is an optimal labeling for T, then L is d-approximation for the bandwidth of G.*

Proof: The proof follows using lemma 6.1 once we observe that $T \subseteq G \subseteq T^t$. □

6.1 Approximating Bandwidth of Convex Bipartite Graphs

Theorem 6.2 *A factor three approximation can be found in linear time for Convex Bipartite Graphs and Permutation Graphs.*

Proof: Note that the tree 3-spanners we have constructed for convex bipartite graphs in theorem 3.2 and that constructed in [20] for permutation graphs are caterpillars of hair length at most 2. The proof of this theorem directly follows from corollary 5 and theorem 6.1.

6.1.1 Two Approximation for Bandwidth of Convex Bipartite Graphs

Now we improve the previous result on Convex Bipartite Graphs by providing a factor two approximation. Given a Convex Bipartite Graph (X, Y, E), we seek to construct an Interval Graph I such that $G \subseteq I \subseteq G^2$. Let $X = \{x_1, x_2, \ldots, x_{n_x}\}$ and $Y = \{1, 2, \ldots, n_y\}$. Recall that, in a Convex Bipartite Graph, the vertices adjacent to any x_i are consecutive in Y. Let x_i be adjacent to vertices in $[L(x_i), R(x_i)]$ for $1 \leq i \leq n_x$. Let $I = (V, E \cup E')$ where $V = X \cup Y$ and

$$E' = \{(x_i, x_j) | \exists k \in Y \text{such that} x_i \sim k, x_j \sim k\}$$

It is obvious that $G \subseteq I \subseteq G^2$. Now, we show that I is an interval graph by constructing an interval representation for I. For each $x_i \in X$ associate an interval $[L(x_i), R(x_i)]$ with it. For each $i \in Y$ associate an interval $[i, i]$ with it. Now, consider an edge (u, v) in I. If $(u, v) \in E$, then obviously without loss of generality $u \in X$ and $v \in Y$. But $u \sim v$ means that $v \in [L(u), R(u)]$. So the intervals associated with u and v intersect. If $(u, v) \in E'$, then there is a $k \in Y$ such that $u \sim k$ and $v \sim k$. So, $k \in [L(u), R(u)]$ and $k \in [L(v), R(v)]$. This means that the intervals associated with u and v intersect. Similarly, we can prove that if two intervals intersect then the corresponding vertices are adjacent in I. So, I is an interval graph. Using the fact that the bandwidth of an interval graph can be computed in $O(n \log n)$ time [28] and using lemma 6.1, the following theorem can be deduced:

Theorem 6.3 *A two approximation of bandwidth of a Convex Bipartite Graph can be found in $O(m + n \log n)$ time.*

6.2 Two Approximation for Bandwidth of Split Graphs

We now give a factor two approximation algorithm for the bandwidth on Split graphs. Note that the complexity of the bandwidth minimization is unknown even on Split graphs, but is likely to be **NP**-hard.

A Split Graph is complete if any vertex in Y is adjacent to all vertices in X. We denote a Complete Split Graph by $C_{X,Y}$. If $G = (X \cup Y, E)$ is a Split Graph then it is obvious that

$$G \subseteq C_{X,Y} \subseteq G^2$$

Lemma 6.2 *The bandwidth of $C_{X,Y}$ is $|X| + \lceil \frac{|Y|}{2} \rceil - 1$.*

For $C_{X,Y}$ we can a give a labeling L as follows. Assign labels $1, 2, \ldots, \lfloor\frac{|Y|}{2}\rfloor$ to any of the $\lfloor\frac{|Y|}{2}\rfloor$ vertices of Y. Then assign labels $\lfloor\frac{|Y|}{2}\rfloor, \ldots, \lfloor\frac{|Y|}{2}\rfloor + |X|$ to vertices of X. Then assign the labels $\lfloor\frac{|Y|}{2}\rfloor + |X|, \ldots, |X| + |Y|$ to the remaining vertices of Y. Obviously, the width of this labeling is $(|X| + |Y|) - (\lfloor\frac{|Y|}{2}\rfloor + 1) = |X| + \lceil\frac{|Y|}{2}\rceil - 1$. So, $bw(C_{X,Y}) \leq b(C_{X,Y}, L) = |X| + \lceil\frac{|Y|}{2}\rceil - 1$.

Now we prove that any labeling cannot have a width less than this value. First we claim that given any optimal labeling L, there is a labeling L' in which all vertices of X get consecutive labels and $b(C_{X,Y}, L') = b(C_{X,Y}, L)$. Let $x_l, x_r \in X$ be the vertices with smallest and largest label respectively. Suppose there is a vertex $y \in Y$ with $L(x_l) < L(y) < L(x_r)$. Obtain another labeling L' from L by interchanging the labels of x_l and y. The only vertices u, v for which $|L(u) - L(v)| < |L'(u) - L'(v)|$ is when $L(u) < L(x_l)$ and $L(x_l) \leq L(v) < L(y)$. But as $|L(x_r) - L(u)| > |L(v) - L(u)|$, the width of L' does not increase. Thus, by repeating this process all vertices of X can be made to get consecutive labels. Now,

$$\begin{aligned} bw(C_{X,Y}) &= b(C_{X,Y}, L') \\ &= \max(L'(x_r) - 1, |X| + |Y| - L'(x_l)) \\ &\geq |X| + \lceil\frac{|Y|}{2}\rceil - 1 \end{aligned}$$

Thus, we have the lemma. □

Theorem 6.4 *A two approximation to the bandwidth of Split Graphs can be found in $O(n)$ time.*

Proof: The labeling in lemma 6.2 is a two approximation for any Split Graph by lemma 6.1. The proof now follows directly. □

7 Conclusions and Open problems

We have *completely* resolved the complexity status of the MINIMUM t-SPANNER problem for all values of t on Bipartite, Convex Bipartite, Chordal, Split and Bounded Degree graphs. The results obtained or improved in this paper are summarized in the table below:

Class	$t = 1$	$t = 2$	$t \geq 3$
Bipartite [7]	P	P	NP-hard
Convex Bipartite	P	P	P
Chordal	P	NP-hard	NP-hard
Perfect	P	NP-hard	NP-hard [7]
Split	P	NP-hard	P
Bounded-degree [8]	P	NP-hard	NP-hard

For completeness we also summarize the results obtained in the companion papers [20, 27]:

Class	$t = 1$	$t = 2$	$t \geq 3$
Interval	P	?	P [27, 20]
Permutation	P	?	P [20]

It is interesting to note that the MINIMUM t-SPANNER problem ($t \geq 3$), which remains **NP**-hard on Bipartite graphs is solvable in linear time when restricted to Convex Bipartite graphs. Thus, we have narrowed down the gap between **P** and **NP** for the MINIMUM t-SPANNER problem in the hierarchy of Bipartite graphs. The same phenomenon occurs when passing from chordal graphs to interval graphs.
Many graphs in concrete applications are planar. That is why the following question, which was also left open in [7], is an extremely interesting one.

Problem 2 *Determine the complexity of the MINIMUM t–SPANNER problem ($t \geq 2$) for planar graphs (or even planar graphs of bounded degree).*

The MINIMUM t–SPANNER problem is not monotone in t. Some cases remain open for $t = 2$.

Problem 3 *Determine the exact complexity of the MINIMUM 2-SPANNER problem on permutation graphs and interval graphs.*

Problem 4 *Determine the exact complexity of the MINIMUM 2-SPANNER problem on graphs of bounded degree k with $5 \leq k \leq 8$.*

We also have seen in the tables above differences between the cases $t = 2$ and $t \geq 3$. Is there a more general feature behind this ? More precisely,

Problem 5 *Given $t_1, t_2 \geq 3$, does there exist a class $\mathcal{C}$ of graphs such that the MINIMUM t_1-SPANNER problem on $\mathcal{C}$ is polynomial-time solvable whereas the MINIMUM t_2-SPANNER problem on $\mathcal{C}$ is **NP**-hard ?*

References

[1] Althöfer, G.Das, D.Dobkin, D.Joseph and J.Soares, On sparse spanners of weighted graphs, *Discrete Comput. Geom.*, **9**(1993), 81=100.

[2] S.F.Assmann, G.W.Peck, M.M.Syslo and J.Zak, The Bandwidth of caterpillars with hairs of length 1 and 2, *SIAM Journal on Algebraic and Discrete Methods*, **2**(1981), 387-393.

[3] B.Awerbuch, Complexity of Network synchronization, *J.ACM*, **32**(1985), 804-823.

[4] H.Bandelt and A.Dress, Reconstructing the shape of a tree from observed dissimilarity data, *Adv. Appl. Math.*, **7**(1986), 309-343.

[5] K.S.Booth and G.S.Lueker, Testing for the Consecutive Ones Property, Interval Graphs and Graph Planarity using P-Q Tree Algorithms. *J. Comput. System Sciences* **13** 335-379 (1976).

[6] L.Cai, Tree Spanners: Spanning Trees that approximate distances. Tech. Report 260/92, University of Toronto, (1992).

[7] L.Cai, NP-completeness of minimum spanner problems, *Disc. Appl. Math.*, **48**(1994), 187-194.

[8] L.Cai and M.Keil, Spanners in Graphs of Bounded Degree, *Networks*, **24**(1994), 233-249.

[9] L.Cai and J.M.Keil, Degree-Bounded Spanners, *Parallel Processing Letters*, **3**(1993), 457-468.

[10] L.P.Chew, There is a planar graph almost as good as the complete graph, *Proceedings 2nd ACM Symposium on Computational Geometry*, Yorktown Heights, NY(1986), 169-177.

[11] M.R.Garey, R.L.Graham, D.S.Johnson and D.E.Knuth, Complexity results for Bandwidth Minimization, *SIAM Journal on Applied Mathematics*, **34**, 477-495,(1978).

[12] M.R.Garey and D.S.Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[13] P.C.Gilmore and A.J.Hoffman, A characterizatiion of comparability graphs and of interval graphs, *Canad. J. Math.*, **16**, 539-548, (1964).

[14] M.C.Golumbic, *Algorithmic Graph Theory and Perfect Graphs.* (Academic Press, New York, 1980).

[15] F.Harary, *Graph Theory* (Addison-Wesley, Reading, Mass., 1969).

[16] T.Kloks, D.Kratsch and H.Müller, Approximating the bandwidth for asteroidal triple-free graphs, Forschungsergebnisse Math/95/6, Friedrich Schiller University, Germany, 1995.

[17] G.Kortsarz and D.Peleg, Generating sparse 2-spanners, in: *Third SWAT*, (1992).

[18] A.L.Liestman and T.C.Shermer, Additive graph Spanners, *Networks*, **23**(1993), 343-364.

[19] A.L.Liestman and T.C.Shermer, Grid Spanners, *Networks*, **23**(2)(1993), 123-133.

[20] M.S.Madanlal, G. Venkatesan and C.Pandu Rangan, Tree 3-spanners on Interval, Permutation and Regular Bipartite Graphs, *Information Processing Letters*, **59** (2) (1996), 97-102.

[21] G. Narasimhan, B. Chandra, D. Gautam and J. Soares. New sparseness results on graph spanners. In *8th Annual ACM Symposium on Computational Geometry*, pages 192–201, 1992.

[22] C.H.Papadimitriou, The $NP-$Completeness of the bandwidth minimization problem, *Computing*, **16**, 263-270, (1976).

[23] D.Peleg and A.A.Schäffer, Graph Spanners, *Journal of Graph Theory*, **13**(1), 99-116, (1989).

[24] D.Peleg and J.D.Ullman, An optimal synchronizer for the hypercube, *Proceedings of the 6th ACM Symposium on Principles of Distributed Computing*, Vancouver (1987), 77-85.

[25] D.Peleg and E.Upfal, A tradeoff between space and efficiency for routing tables, *Proceedings of the 20th ACM Symposium on Theory of Computing*, Chicago (1988), 43-52.

[26] G.Ramalingam and C.Pandu Rangan, A Unified Approach to Domination problems on Interval graphs, *Info. Proc. Letters*, **27** (1988), 271-274.

[27] J.A. Makowsky and U. Rotics. Spanners in interval and chordal graphs. Technical Report, Department of Computer Science, Technion–Israel Institute of Technology, April 1996.

[28] A.P.Sprague, An $O(n \log n)$ algorithm for bandwidth of interval graphs, *SIAM Journal on Discrete Mathematics*, **7**, 213-220, (1994).

[29] M.Yannakakis and F.Gavril, Edge Dominating sets in graphs, *SIAM Jl. of Appl. Math.*, **38** (1980), 364-372.

[30] M.Yannakakis, Edge-deletion problems, *SIAM J. Computing*, **10** (1981), 297-309.