Graph searching, elimination trees, and a generalization of bandwidth

Fedor V. Fomin, Pinar Heggernes, and Jan Arne Telle
{fomin, pinar, telle}@ii.uib.no

Department of Informatics, University of Bergen, N-5020 Bergen, Norway

Abstract. The bandwidth minimization problem has a long history and a number of practical applications. In this paper we introduce a generalization of bandwidth to partially ordered layouts. We consider this generalization from two main viewpoints: graph searching and tree decompositions. The three graph parameters pathwidth, profile and bandwidth related to linear layouts can be defined by variants of graph searching using a standard fugitive. Switching to an inert fugitive, the two former parameters are generalized to treewidth and fill-in, and our first viewpoint considers the analogous tree-like generalization that arises from the bandwidth variant. Bandwidth also has a definition in terms of ordered path decompositions, and our second viewpoint generalizes this in a natural way to ordered tree decompositions. In showing that both generalizations are equivalent we employ the third viewpoint of elimination trees, as used in the field of sparse matrix computations. We call the resulting parameter the treespan of a graph and prove some of its combinatorial and algorithmic properties.

1 Motivation through graph searching games

Different versions of graph searching has been attracting the attention of researchers from Discrete Mathematics and Computer Science for a variety of elegant and unexpected applications in different and seemingly unrelated fields. There is a strong resemblance of graph searching to certain pebble games [15] that model sequential computation. Other applications of graph searching can be found in VLSI theory since this game-theoretic approach to some important parameters of graph layouts such as the cutwidth [19], the topological bandwidth [18], the bandwidth [9], the profile [10], and the vertex separation number [8] is very useful for the design of efficient algorithms. There is also a connection between graph searching, pathwidth and treewidth, parameters that play an important role in the theory of graph minors developed by Robertson & Seymour [3, 7, 22]. Furthermore, some search problems have applications in problems of privacy in distributed environments with mobile eavesdroppers ('bugs') [11].

In the standard node-search version of searching, a single searcher is placed at a vertex of a graph G at every move, while from other vertices searchers are removed (see *e.g.* [15]). The purpose of searching is to capture an invisible fugitive moving fast along paths in G. The fugitive is not allowed to run through the vertices currently occupied by searchers. So the fugitive is caught when a searcher is placed on the vertex it occupies, and it has no possibility to leave the vertex because all the neighbors are occupied (guarded) by searchers. The goal of search games is to find a search strategy to *guarantee* the fugitive's capture while minimizing some resource usage.

Because the fugitive is invisible, the only information the searchers possess are the previous search moves that may give knowledge about subgraphs where the fugitive cannot possibly be present. This brings us to the interesting interpretation of the search problem [3] as the problem of fighting against damage spread in complex systems, *e.g.* the spread of a mobile computer virus in networks. Initially all vertices are viewed as contaminated (infected by a virus or damaged) and a contaminated vertex is cleared once it is occupied by a searcher (checked by an anti-virus program). A clear vertex v is recontaminated if there is a path without searchers leading from v to a contaminated vertex. In some applications it is required that recontamination should never occur, but for most of the search game variants considered it can be shown, by some very clever techniques, that the resource usage does not increase in spite of this constraint [15, 16, 4, 7]. The 'classical' goal of the search problem is to find the search program such that the maximum number of searchers in use at any move is minimized. The minimum number of searchers needed to clear the graph is related to the parameter called pathwidth. Dendris et al. [7] studied a variation of the node-search problem with *inert*, or lazy, fugitive. In this version of the game the fugitive is allowed to move only just before a searcher is placed on the vertex it occupies. The smallest number of searchers needed to find the fugitive in this version of searching is related to the parameter called treewidth [7].

Another criteria of optimality in node-searching, namely *search cost* was studied in [10]. Here the goal is to minimize the sum of the number of searchers in use over all moves of the search program. The search cost of a graph is equal to the interval completion number, or profile, which is the smallest number of edges in any interval supergraph of the given graph. Looking at the search cost version but now with an inert fugitive, it is easy to see that this parameter is equal to the smallest number of edges in the chordal supergraph of a given graph, so called *fill-in*. We thus have the following elegant relation: the parameters related to standard node searching (pathwidth, profile) expressible in terms of interval completion problems, correspond in inert fugitive searching to chordal completion problems (treewidth, fill-in).

In this paper we want to minimize the maximum length of time (number of intermediate moves) during which a searcher occupies a vertex. A similar problem for pebbling games (that can be transferred into search terms) was studied by Rosenberg & Sudborough [23]. It turned out that this parameter is related to the bandwidth of a graph G, which is the minimum over all linear layouts of vertices in G of the maximum distance between imagesof adjacent vertices. The following table summarizes the knowledge about known relations between graph searching and graph parameters.

	Number of Searchers	Cost of Searching	Occupation Time
Standard Search	pathwidth [15]	profile [10]	bandwidth [23]
Inert Search	treewidth [7]	fill-in	???

One of the main questions answered in this paper concerns the entry labeled ??? above: What kind of graph parameter $\gamma_m(G)$ corresponds to the minimum occupation time for inert fugitive search? In section 2 we introduce a generalization of bandwidth to tree-like layouts, called *treespan*, based on what we call ordered tree decompositions. In section 3 we give the formal definition of the parameter $\gamma_m(G)$, and then in section

4 we show that it is equivalent to a parameter arising from elimination trees, as used in the sparse matrix computation community. In section 5 we show the equivalence also between this elimination tree parameter and treespan, thereby providing evidence that the entry labeled ??? above indeed corresponds to a natural generalization of bandwidth to partially ordered (tree) layouts. Finally in section 6 we obtain some algorithmic and complexity results on the treespan parameter.

2 Motivation through tree decompositions

We assume simple, undirected, connected graphs $G = (V, E)$, where $|V| = n$. We let $N(v)$ denote the neighbors of vertex v, and $d(v) = |N(v)|$ is the *degree* of v. The maximum degree of any vertex in G is denoted by $\Delta(G)$. For a set of vertices $U \subseteq V$, $N(U) = \{v \notin U \mid uv \in E \text{ and } u \in U\}$. $H \subseteq G$ means that H is a subgraph of G. For a rooted tree T, and a vertex v in T, we let $T[v]$ denote the subtree of T with root in v.

A *chord* of a cycle C in a graph is an edge that connects two non-consecutive vertices of C. A graph G is *chordal* if there are no induced chordless cycles of length ≥ 4 in G. Given any graph $G = (V, E)$, a *triangulation* $G^+ = (V, E^+)$ of G is a chordal graph such that $E \subseteq E^+$.

A *tree decomposition* of a graph $G = (V, E)$ is a pair (X, T), where $T = (I, M)$ is a tree and $X = \{X_i : i \in I\}$ is a collection of subsets of V called *bags*, such that:

1. $\bigcup_{i \in I} X_i = V$
2. $uv \in E \Rightarrow \exists i \in I$ with $u, v \in X_i$
3. For all vertices $v \in V$, the set $\{i \in I \mid v \in X_i\}$ induces a connected subtree of T.

The *width* of a tree decomposition (X, T) is $tw(X, T) = \max_{i \in I} |X_i| - 1$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G. A *path decomposition* is a tree decomposition (X, T) such that T is a path. The *pathwidth* of a graph G is the minimum width over all path decompositions of G. We refer to Bodlaender's survey [5] for further information on treewidth.

For a chordal graph G, the treewidth is one less than the size of the largest clique in G. For a non-chordal graph G, the treewidth is the minimum treewidth over all triangulations of G. This is due to the fact that a tree decomposition (X, T) of G actually corresponds to a triangulation of the given graph G: Simply add edges to G such that each bag of X becomes a clique. The resulting graph, which we will call $tri(X, T)$ is a chordal graph of which G is a subgraph. In addition, any triangulation G^+ of G is equal to $tri(X, T)$ for some tree decomposition (X, T) of G.

Another reason why tree decompositions and chordal graphs are closely related is that chordal graphs are exactly the intersection graphs of subtrees of a tree [14]. Analogously, interval graphs are related to path decompositions, and they are the intersection graphs of subpaths of a path. A graph is *interval* if there is a mapping I of its vertices into sets of consecutive integers such that for each pair of vertices v, w the following is true: vw is an edge $\Leftrightarrow I(v) \cap I(w) \neq \emptyset$. Interval graphs form a subclass of chordal graphs. Similar to treewidth, the pathwidth of a graph G is one less than the smallest clique number over all triangulations of G into interval graphs.

The bandwidth of G, $bw(G)$, is defined as the minimum, over all linear orders of the vertices of G, maximum difference between labels of two adjacent vertices. Similar to pathwidth and treewidth, bandwidth can be defined in terms of triangulations as follows. A graph isomorphic to $K_{1,3}$ is referred to as a *claw*, and a graph that does not contain an induced claw is said to be *claw-free*. An interval graph G is a *proper interval graph* if it is claw-free [21]. As it was observed by Parra & Scheffler [20], the bandwidth of a graph G is one less than the smallest clique number over all triangulations of G into proper interval graphs. One can define bandwidth in terms of *ordered path decompositions*. In an ordered path decomposition, the bags are numbered $1, 2, ..., n$ from left to right. The first bag X_1 contains only one vertex of G, and for $1 \le i \le n-1$ we have $|X_{i+1} \setminus X_i| = 1$, meaning that exactly one new graph vertex is introduced in each new bag. The number of bags a vertex v belongs to is denoted by $l(v)$. It is easy to show that $bw(G)$ is the minimum, over all ordered path decompositions, $\max\{l(v) - 1 \mid v \in V\}$.

The natural question here is, what kind of parameter corresponds to bandwidth when, instead of path decompositions, we switch to tree decompositions? This brings us to the definition of ordered tree decomposition and treespan.

Definition 1. *An* ordered tree decomposition (X, T, r) *of a graph* $G = (V, E)$ *is a tree decomposition* (X, T) *of* G *where* $T = (I, M)$ *is a rooted tree with root* $r \in I$*, such that:*

- $|X_r| = 1$*, and if* i *is the parent of* j *in* T*, then* $|X_j \setminus X_i| = 1$.

Definition 2. *Given a graph* $G = (V, E)$ *and an ordered tree decomposition* (X, T, r) *of* G*,*

- $l(v) = |\{i \in I \mid v \in X_i\}|$ *(number of bags that contain* v*), for each* $v \in V$.
- $ts(X, T, r) = \max\{l(v) \mid v \in V\} - 1$.

The treespan *of a graph* G *is* $ts(G) = \min\{ts(X, T, r) \mid (X, T, r)$ *is an ordered tree decomposition of* $G\}$.

Since every ordered path decomposition is an ordered tree decomposition, it is clear that for every graph G, $ts(G) \le bw(G)$.

3 Search minimizing occupation time with inert fugitive

In this section we give a formal definition of minimum occupation time for inert fugitive searching. A *search program* Π on a graph $G = (V, E)$ is the sequence of pairs

$$(A_0, Z_0), (A_1, Z_1), \ldots, (A_m, Z_m)$$

such that

I. For $i \in \{0, \ldots, m\}$, $A_i \subseteq V$ and $Z_i \subseteq V$. We say that vertices A_i are *cleared*, vertices $V - A_i$ are *contaminated* and vertices Z_i are *occupied by searchers* at the ith step.
II. (*Initial state.*) $A_0 = \emptyset$ and $Z_0 = \emptyset$. All vertices are contaminated.
III. (*Final state.*) $A_0 = V$ and $Z_0 = \emptyset$. All vertices are cleared.

IV. (*Placing-removing searchers and clearing vertices.*) For $i \in \{1, \dots, m\}$ there exists $v \in V$ and $Y_i \subseteq A_{i-1}$ such that $A_i - A_{i-1} = v$ and $Z_i = Y_i \cup \{v\}$. Thus at every step one of the searchers is placed on a contaminated vertex v while the others are placed on cleared vertices Y_i. The searchers are removed from vertices $Z_{i-1} - Y_i$. Note that Y_i is not necessarily a subset of Z_{i-1}.

V. (*Possible recontamination.*) For $i \in \{1, \dots, m\}$ $A_i - \{v\}$ is the set of vertices $u \in A_{i-1}$ such that *every* uv-path has an internal vertex in Z_i. This means that the fugitive awakening in v can run to a cleared vertex u if there is a uv-path unguarded by searchers.

Dendris, Thilikos & Kirousis [7] initiated the study of inert search problem, where the problem is to find a search program Π with the smallest $\max_{i \in \{0,\dots,m\}} |Z_i|$ (this maximum can be treated as the maximum number of searchers used in one step). It turns out that this number is equal to the treewidth of a graph. We find an alternative measure of search to be interesting as well. For a search program $\Pi = (A_0, Z_0), (A_1, Z_1), \dots, (A_m, Z_m)$ on a graph $G = (V, E)$ and vertex $v \in V$ we define

$$\delta_i(v) := \begin{cases} 1, v \in Z_i \\ 0, v \notin Z_i \end{cases}$$

Then the number $\sum_{i=0}^{m} \delta_i(v)$ is the number of steps at which vertex v was occupied by searchers. For a program Π we define the *maximum vertex occupation time* to be

$$\gamma(\Pi, G) = \max_{v \in V} \sum_{i=0}^{m} \delta_i(v).$$

The *vertex occupation time* of a graph G, denoted by $\gamma(G)$, is the minimum maximum vertex occupation time over all search programs on G.

A search program $(A_0, Z_0), (A_1, Z_1), \dots, (A_m, Z_m)$ is *monotone* if for each $i \in \{1, \dots, m\}$, $A_{i-1} \subseteq A_i$. Note that recontamination does not occur when a searcher is placed on a contaminated vertex thus awaking the fugitive.

Finally, for a graph G we define $\gamma_m(G)$, as the minimum maximum vertex occupation time over all *monotone* search programs on G. We do not know whether $\gamma_m(G) = \gamma(G)$ for every graph G, and leave it as an interesting open question.

4 Searching and elimination trees

In this section we discuss a relation between $\gamma_m(G)$ and elimination trees of G. This relation is not only interesting in its own but also serves as a tool in further proofs.

For a graph $G = (V, E)$, an elimination order $\alpha : \{1, 2, ..., n\} \to V$ is a linear order of the vertices of G. For each given order α, a unique triangulation G_α^+ of G can be computed from the following procedure: Starting with vertex $\alpha(1)$, at each step i, make the higher numbered neighbors of vertex $\alpha(i)$ in the transitory graph into a clique by adding edges. The resulting graph, which is denoted G_α^+, is chordal [12], and the given elimination ordering decides the quality of this resulting triangulation. The following lemma follows from the definition of G_α^+.

Lemma 1. *uv is an edge of G^+_α $\Leftrightarrow$ uv is an edge of G or there is a path $u, x_1, x_2, ..., x_k, v$ in G with $k \geq 1$ such that all x_i are ordered before u and v by α (in other words, $\max\{\alpha^{-1}(x_i) \mid 1 \leq i \leq k\} < \min\{\alpha^{-1}(u), \alpha^{-1}(v)\}$).*

Definition 3. *For a vertex $v \in V$ we define $madj^+(v)$ to be the set of vertices $u \in V$ such that $\alpha(u) \geq \alpha(v)$ and uv is an edge of G^+_α. (The higher numbered neighbors of v in G^+_α.)*

Given a graph G, and an elimination order α on G, the corresponding *elimination tree* is a rooted tree $ET = (V, P)$, where the edges in P are defined by the following *parent* function: $parent(\alpha(i)) = \alpha(j)$ where $j = \min\{k \mid \alpha(k) \in madj^+(\alpha(i))\}$, for $i = 1, 2, ..., n$. Hence the elimination tree is a tree on the vertices of G, and vertex $\alpha(n)$ is always the root. The *height* of the elimination tree is the longest path from a leaf to the root. *Minimum elimination tree height* of a graph G, $mh(G)$ is the minimum height of an elimination tree corresponding to any triangulation of G. For a vertex $u \in V$ we denote by $ET[u]$ the subtree of ET rooted in u and containing all descendants (in ET) of u. It is important to note that, for two vertices u and v such that $ET[u]$ and $ET[v]$ are disjunct subtrees of ET, no vertex belonging to $ET[u]$ is adjacent to any vertex belonging to $ET[v]$ in G or G^+_α. In addition, $N(ET[v])$ is a clique in G^+_α, and a minimal vertex separator in both G^+_α and G when v is not the only child of its parent in ET.

Let α be an elimination order of the vertices of a graph $G = (V, E)$ and let ET be the corresponding elimination tree of G. Observe that the elimination tree ET gives enough information about the chordal completion G^+ of G that ET corresponds to. It is important to understand that *any post order α of the vertices of ET is an elimination order on G that results in the same chordal completion $G^+_\alpha = G^+$*. Thus given G and ET, we have all the information we need on the corresponding triangulation.

Definition 4. *Given an elimination tree ET of G, the* pruned subtree with root in x, *$ET_p[x]$, is the subtree obtained from $ET[x]$ by deleting all descendants of every vertex $y \in ET[x]$ such that $xy \in E(G)$ but no descendant of y is a neighbor of x in G.*

Thus, the leaves of $ET_p[x]$ are neighbors of x in G, and all lower numbered neighbors in G^+ of x are also included in $ET_p[x]$. In addition, there might clearly appear vertices in $ET_p[x]$ that are not neighbors of x in G. However, every neighbor of x in G^+ appears in $ET_p[x]$, as we prove in the following lemma.

Lemma 2. *Let α be an elimination order of graph $G = (V, E)$ and let ET be a corresponding elimination tree. Then for any $u, v \in V$, $u \in ET_p[v]$ if and only if $v \in madj^+(u)$.*

Proof. Let $u \in ET_p[v]$ and let w be a neighbor of v in G such that u is on a vw-path in ET. By the definition of pruned tree such a vertex w always exists. Because ET is an elimination tree, there is a uw-path P^+ in G^+_α such that for any vertex x of P^+, $\alpha^{-1}(x) \leq \alpha^{-1}(u)$. By Lemma 1, this implies that there is also an uw-path P in G such that for any vertex x of P, $\alpha^{-1}(x) \leq \alpha^{-1}(u)$. Since w is adjacent to v in G, we conclude that $v \in madj^+(u)$.

Let $v \in madj^+(u)$. Then there is an uv-path P in G (and hence in G^+_α) such that all inner vertices of the path are ordered before u in α. Let w be the vertex of P adjacent to v. Because ET is elimination tree, we have that u is on vw-path in ET. Thus $u \in ET_p[v]$.

We define a parameter called *elimination span*, es, as follows:

Definition 5. *Given an elimination tree ET of a graph $G = (V, E)$, for each vertex $v \in V$ we define $s(v) = |ET_p[v]|$ and $es(ET) = \max\{s(v) \mid v \in V\} - 1$. The* elimination span *of a graph G is $es(G) = \min\{es(ET) \mid ET$ is an elimination tree of $G\}$.*

Theorem 1. *For any graph $G = (V, E)$, $es(G) = \gamma_m(G) - 1$.*

Proof. Let us prove $es(G) \leq \gamma_m(G) - 1$ first. Let $\Pi = (A_0, Z_0), (A_1, Z_1), \ldots, (A_m, Z_n)$ be a monotone search program. At every step of the program exactly one new vertex $A_i - A_{i-1}$ is cleared. Thus we can define the vertex ordering α by putting for $1 \leq i \leq n$

$$\alpha(A_i - A_{i-1}) = n - i + 1.$$

At the ith step, when a searcher is placed at a vertex $u = A_i - A_{i-1}$ every vertex $v \in A_i$ such that there is a uv-path with no inner vertices in A_i should be occupied by a searcher (otherwise v would be recontaminated). Therefore, $v \in madj^+(u)$ and the number of steps when a vertex v is occupied by searchers, is $|\{u \mid v \in madj^+(u)\}|$. By Lemma 2, $|\{u \mid v \in madj^+(u)\}| = s(v)$ and we arrive at

$$es(ET) \leq \gamma_m(\Pi, G) - 1.$$

We now show that $es(G) \geq \gamma_m(G) - 1$. Let ET be an elimination tree and let α be a corresponding elimination vertex ordering. We consider a search program Π where at the ith step of the program, $1 \leq i \leq n$, the searchers occupy the set of vertices $madj^+(v)$, where v is a vertex with $\alpha(v) = n - i + 1$. Let us first prove that Π is recontamination free. Suppose, on the contrary, that a vertex u is recontaminated at the ith step after placing a searcher on a vertex v. Then there is a uv-path P such that no vertex of P except v contains a searcher at the ith step. On the other hand, vertex u is after v in ordering α. Thus P should contain a vertex $w \in madj^+(u)$, $w \neq u$, occupied by a searcher. This is a contradiction. Since every vertex was occupied at least once and no recontamination occurs, we conclude that at the end of Π all vertices are cleared. Every vertex v was occupied by searchers during $|\{u \mid v \in madj^+(u)\}|$ steps and using Lemma 2 we conclude that $es(ET) \geq \gamma_m(\Pi, G) - 1$.

5 Ordered tree decompositions and elimination trees

In this section we discuss a relation between the treespan $ts(G)$ and elimination trees of G, establishing that $ts(G) = \gamma_m(G)$. We first give a simplified view of ordered tree decompositions and then proceed to prove some of their properties.

There are exactly n bags in X of an ordered tree decomposition (X, T, r) of G. Thus, the index set I for $X_i, i \in I$ can be chosen so that $I = V$, with $r \in V$. Then T is a tree on the vertices of G. To identify the bags and to define the correspondence between I and V uniquely, name the bags so that X_r is the bag corresponding to the root r of T. Regarding the bags in a top down fashion according to T, name the bag in which vertex v appears for the first time X_v and the corresponding tree node v. Thus if y is the parent of v in T then $X_v \setminus X_y = \{v\}$. This explains how to rename the bags and the vertices of

T with elements from V given a tree decomposition based on I. However, if we replace i with v and I with V in Conditions 1 - 3 of the definition of a tree decomposition, and change condition in the definition of ordered tree decompositions to "$X_r = \{r\}$, and if y is the parent of v in T then $X_v \setminus X_y = \{v\}$", then this will automatically give a tree T on the vertices of G as we have explained above. For the remainder of this paper, when we mention an ordered tree decomposition (X, T, r), we will assume that T is a tree on the vertices of G as explained here. The following lemma will make the role of T even clearer.

Lemma 3. *Given a graph $G = (V, E)$ and a rooted tree $T = (V, P)$, there exists an ordered tree decomposition (X, T, r) of $G \Leftrightarrow$ for every edge $uv \in E$, u and v have an ancestor-descendant relationship in T.*

Proof. Assume that T corresponds to a valid ordered tree decomposition of G, but there is an edge uv in G such that $T[u]$ and $T[v]$ are disjunct subtrees of T. X_u is the first bag in which u appears and X_v is the first bag in which v appears, thus u and v do not appear in any bag X_w where w is on the path from u to the root or from v to the root in T. Thus if u and v appear together in any other bag X_y where y belongs to $T[u]$ or $T[v]$ or any other disjunct subtree in T, this would violate Condition 3 of a tree decomposition. Therefore, u and v cannot appear together in any bag, and there cannot exist a valid decomposition (X, T, r) of G.

For the reverse direction, assume that for every edge uv in G, u and v have an ancestor-descendant relationship in T. Assume without loss of generality that v is an ancestor of u. Then the bags can be defined so that 1) X_v contains v, 2) no bag X_y contains v where y is an ancestor of v, 3) for every vertex w on the path from v to u in T, X_w contains v (and w of course), and 4) X_u contains both u and v. We can see that all the conditions of an ordered tree decomposition are satisfied.

Lemma 4. *Let (X, T, r) be an ordered tree decomposition of a given graph. For every edge uv in $tri(X, T)$, u and v have an ancestor-descendant relationship in T.*

Proof. As we have seen in the proof of Lemma 3, if u and v belong to disjunct subtrees of T, then they cannot appear together in the same bag. Since only the bags are made into cliques, u and v cannot belong to the same clique in $tri(X, T)$, which means that the edge uv does not exist in $tri(X, T)$.

Lemma 5. *Let (X, T, r) be an ordered tree decomposition of a given graph. Let uv be an edge of $tri(X, T)$ such that v is an ancestor of u in T. Then v belongs to bag X_w for every w on the path from v to u including X_v and X_u.*

Proof. Vertex v appears for the first time in X_v on the path from the root, and u appears for the first time in X_u. For every vertex w on the path from v to u, exactly vertex w is introduced in X_w. Thus X_u is the first bag in which u and v both can belong to. In order for this to be possible, v must belong to bag X_w for every vertex w on the path from v to u in T.

Lemma 6. *For each graph G, there exists an ordered tree decomposition (X, T, r) of G of minimum treespan such that if u is a child of v in T then $v \in X_u$.*

Proof. Assume that u is a child of v in T and $v \notin X_u$. Clearly, uv is not an edge of G. Since v does not belong to any bag X_y for a descendant y of u, we can move u up to be a child of a node w in T where uw is an edge of G and where w is the first node on the path from v to the root that is a neighbor of u.

Lemma 7. *Let (X, T, r) be an ordered tree decomposition of G, and let $\alpha : \{1, ..., n\} \rightarrow V$ be a post order of T. Then $G_\alpha^+ \subseteq tri(X, T)$.*

Proof. Let uv be an edge of G_α^+, and assume without loss of generality that u has a lower number than v according to α. If uv is an edge of of G, then we are done. Otherwise, due to Lemma 1, there must exist a path $u, x_1, x_2, ..., x_k, v$ in G with $k \geq 1$ such that all x_i are ordered before u. Since α is a post order of T, none of the vertices $x_i, i = 1, ..., k$, can lie on the path from u to the root in T. Consequently and due to Lemma 3, since ux_1 is an edge of G, x_1 belongs to $T[u]$. With the same argument, since $x_1, x_2, ..., x_k$ is a path in G, all the vertices $x_1, x_2, ..., x_k$ must belong to $T[u]$. Now, since vx_k is an edge in G, v must be an ancestor of x_k and thus of u in T, where u lies on the path from v to x_k. By Lemma 5, vertex v must be present in all bags X_w where w lies on the path from v to x_k, and consequently also in bag X_u. Therefore, u and v are both present in bag X_u and are neighbors in $tri(X, T)$.

Lemma 8. *Let (X, T, r) be an ordered tree decomposition of G, and let α be a post order of T. Let ET be the elimination tree of G_α^+. Then for any vertex u, if v is the parent of u in ET, then v lies on the path from u to the root in T.*

Proof. Since v is the parent of u in ET, uv is an edge of G_α^+. By Lemma 7, uv is also an edge of $tri(X, T)$. By Lemma 4, u and v must have an ancestor-descendant relationship in T. Since α is a post order of T, and $\alpha^{-1}(u) < \alpha^{-1}(v)$, v must be an ancestor of u in T.

Theorem 2. *For any graph G, $ts(G) = es(G)$.*

Proof. First we prove that $ts(G) \leq es(G)$. Let $ET = (V, P)$ be an elimination tree of G such that $es(G) = es(ET)$, and let r be the root vertex of ET. We define an ordered tree decomposition $(X = \{X_v \mid v \in V\}, T = ET, r)$ of G in the following way. For each vertex v in ET, put v in exactly the bags X_u such that $u \in ET_p[v]$. Regarding ET top down, each vertex u will appear for the first time in bag X_u, and clearly $|X_u \setminus X_v| = 1$ whenever v is the parent of u. It remains to show that (X, ET) is a tree decomposition of G. Conditions 1 and 3 of a tree decomposition are trivially satisfied since $ET_p[v]$ is connected and includes u for every vertex v. For Condition 2, if uv is an edge of G, then the lower numbered of v and u is a descendant of the other in ET. Let us say u is a descendant of v, then $u \in ET_p[v]$, and v and u will both appear in bag X_u. Thus (X, ET) is an ordered tree decomposition of G, and clearly, $ts(X, ET) = es(G)$. Consequently, $ts(G) \leq es(G)$.

Now we show that $es(G) \leq ts(G)$. Let (X, T, r) be an ordered tree decomposition of G with $ts(X, T, r) = ts(G)$. Let α be a post order on T, and let ET be the elimination tree of G_α^+. For any two adjacent vertices u and v in G, u and v must have an ancestor-descendant relationship both in T and in ET. Moreover, due to Lemma 8, all vertices that are on the path between u and v in ET must also be present on the path between

u and v in T. Assume, without loss of generality, that u is numbered lower than v. By Lemma 5, v must belong to all the bags corresponding to the vertices on the path from v to u in T. Thus for each vertex v, $s(v)$ in ET is at most $l(v)$ in (X, T, r). Consequently, $es(G) \leq ts(G)$, and the proof is complete.

Theorems 1 and 2 imply the main combinatorial result of this paper.

Corollary 1. *For any graph G, $ts(G) = es(G) = \gamma_m(G)$.*

6 Treespan of some special graph classes

The *diameter* of a graph G, $diam(G)$, is the maximum length of a shortest path between any two vertices of G. The *density* of a graph G is defined as $dens(G) = (n-1)/diam(G)$. The following result is well known

Lemma 9. [6] For any graph G, $bw(G) \geq \max\{dens(H) \mid H \subseteq G\}$.

A *caterpillar* is a tree consisting of a main path of vertices of degree at least two with some leaves attached to this main path.

Theorem 3. *For any graph G, $ts(G) \geq \max\{dens(H) \mid H \subseteq G$ and H is a caterpillar$\}$.*

Proof. Let the caterpillar H be a subgraph of G consisting of the following main path: $c_1, c_2, ..., c_{diam(H)-1}$. We view the bags of an ordered tree decomposition as labeled by vertices of G in the natural manner (as described before Lemma 3). Let (X, T, r) be an ordered tree decomposition of G with (X', T', r') being the topologically induced ordered tree decomposition on H, *i.e.* containing only bags labeled by a vertex from H, where we contract edges of T going to vertices labeled by vertices not in H to get T'. Let X_{c_i} be the 'highest' bag in (X', T', r') labeled by a vertex from the main path, so that only the subtree of (X', T', r') rooted at X_{c_i} contains any vertices from the main path. Let there be $h+1$ bags on the path from X_{c_i} to the root $X_{r'}$ of (X', T', r'). Since vertex r' of H (a leaf unless $r' = c_i$) is adjacent to a vertex on the main path it appears in at least $h+1$ bags, giving $ts(G) \geq h$. Moreover, by applying Lemma 3 we get that T' between its root $X_{r'}$ and X_{c_i} consists simply of a path without further children, so that the subtree rooted at X_{c_i} has $|V(H)| - h$ bags. Each of these bags contain a vertex from the main path since every leaf of H is adjacent in H only to a vertex on the main path, and by the pigeonhole principle we thus have that some main path vertex lives in at least $\lceil (|V(H)| - h)/(diam(H) - 1) \rceil$ bags. If $(|V(H)| - h)/(diam(H) - 1)$ is not an integer, then immediately we have the bound $ts(G) \geq \lfloor (|V(H)| - h)/(diam(H) - 1) \rfloor$. If $(diam(H) - 1)$ on the other hand does divide $(|V(H)| - h)$ then we apply the fact that at least $diam(H) - 2$ bags must contain at least two vertices from the main path, to account for edges between them, and for $diam(H) \geq 3$ (which holds except for the trivial case H a star) this increases the span of at least one main path vertex and we again get $ts(G) \geq \lfloor (|V(H)| - h)/(diam(H) - 1) \rfloor$.

Thus $ts(G) \geq \max\{h, \lfloor (|V(H)| - h)/(diam(H) - 1) \rfloor\}$. If $h \leq dens(H)$ we have that $\lfloor (|V(H)| - h)/(diam(H) - 1) \rfloor \geq (|V(H)| - 1)/diam(H)$ and therefore $\lfloor (|V(H)| - h)/(diam(H) - 1) \rfloor \geq dens(H)$. We conclude that $ts(G) \geq dens(H)$ and the lemma follows.

With this theorem, in connection with the following result from [2], we can conclude that $bw(G) = ts(G)$ for a caterpillar graph G.

Lemma 10. [2] For a caterpillar graph G, $bw(G) \leq \max\{dens(H) \mid H \subseteq G\}$.

Lemma 11. *For a caterpillar graph G, $bw(G) = ts(G) = \max\{dens(H) \mid H \subseteq G\}$.*

Proof. Let G be a caterpillar graph. Then, $bw(G) \geq ts(G) \geq \max\{dens(H) \mid H \subseteq G\} \geq bw(G)$. The first inequality was mentioned in Section 5, the second inequality is due to Theorem 3, and the last inequality is due to Lemma 10 since G is a caterpillar. Thus all of the mentioned parameters on G are equal.

A set of three vertices x, y, z of a graph G is called an *asteroidal triple* (AT) if for any two of these vertices there exists a path joining them that avoids the (closed) neighborhood of the third. A graph G is called an *asteroidal triple-free* (AT-free) graph if G does not contain an asteroidal triple. This notion was introduced by Lekkerkerker an Boland [17] for the following characterization of interval graphs: G is an interval graph if and only if it is chordal and AT-free.

A graph G is said to be *cobipartite* if it is the complement of a bipartite graph. Notice that cobipartite graphs form a subclass of AT-free claw-free graphs. Another subclass of AT-free claw-free graphs are the proper interval graphs, which were mentioned earlier. Thus G is a proper interval graph if and only if it is chordal and AT-free claw-free. A *minimal* triangulation of G is a triangulation H such that no proper subgraph of H is a triangulation of G. The following result is due to Parra and Scheffler.

Theorem 4. [20] *Let G be an AT-free claw-free graph. Then every minimal triangulation of G is a proper interval graph, and hence, $bw(G) = pw(G) = tw(G)$.*

Theorem 5. *For an AT-free claw-free graph G, $ts(G) = bw(G) = pw(G) = tw(G)$.*

Proof. Let G be AT-free claw-free and let H be its minimal triangulation such that $ts(G) = ts(H)$. Such a graph H must exist, since for an optimal ordered tree decomposition (X, T, r), the graph $tri(X, T)$ is chordal and $ts(tri(X, T)) = ts(G)$. Thus any minimal graph from the set of chordal graphs 'sandwiched' between $tri(X, T)$ and G can be chosen as H. By Theorem 4, H is a proper interval graph. Thus $\omega(H) - 1 = bw(H) \geq bw(G)$. Since $ts(H) \geq \omega(H) - 1$, we have that $ts(G) = ts(H) \geq \omega(H) - 1 \geq bw(G) \geq ts(G)$.

By the celebrated result of Arnborg, Corneil & Proskurowski [1], tree-width (and hence path-width and bandwidth) is NP-hard even for cobipartite graphs. Thus Theorem 5 yields the following corollary.

Corollary 2. *Computing treespan is NP-hard for cobipartite graphs.*

We conclude with an open question. For any graph G, $ts(G) \geq \lceil \Delta(G)/2 \rceil$. For trees of maximum degree at most 3 it is easy to prove that $ts(G) \leq \lceil \Delta(G)/2 \rceil$. It is an interesting question whether treespan can be computed in polynomial time for trees of larger max degree. Notice that bandwidth remains NP-complete on trees of max degree 3 [13].

References

1. S. Arnborg, D. G. Corneil, and A. Proskurowski, *Complexity of finding embeddings in a k-tree*, SIAM J. Alg. Disc. Meth., 8 (1987), pp. 277–284.
2. S. F. Assman, G. W. Peck, M. M. Sysło, and J. Zak, *The bandwidth of caterpillars with hairs of length 1 and 2*, SIAM J. Alg. Disc. Meth., 2 (1981), pp. 387–392.
3. D. Bienstock, *Graph searching, path-width, tree-width and related problems (a survey)*, DIMACS Ser. in Discrete Mathematics and Theoretical Computer Science, 5 (1991), pp. 33–49.
4. D. Bienstock and P. Seymour, *Monotonicity in graph searching*, J. Algorithms, 12 (1991), pp. 239 – 245.
5. H. L. Bodlaender, *A partial k-arboretum of graphs with bounded treewidth*, Theor. Comp. Sc., 209 (1998), pp. 1–45.
6. P. Z. Chinn, J. Chvátalová, A. K. Dewdney, and N. E. Gibbs, *The bandwidth problem for graphs and matrices — a survey*, J. Graph Theory, 6 (1982), pp. 223–254.
7. N. D. Dendris, L. M. Kirousis, and D. M. Thilikos, *Fugitive-search games on graphs and related parameters*, Theor. Comp. Sc., 172 (1997), pp. 233–254.
8. J. A. Ellis, I. H. Sudborough, and J. Turner, *The vertex separation and search number of a graph*, Information and Computation, 113 (1994), pp. 50–79.
9. F. Fomin, *Helicopter search problems, bandwidth and pathwidth*, Discrete Appl. Math., 85 (1998), pp. 59–71.
10. F. V. Fomin and P. A. Golovach, *Graph searching and interval completion*, SIAM J. Discrete Math., 13 (2000), pp. 454–464 (electronic).
11. M. Franklin, Z. Galil, and M. Yung, *Eavesdropping games: A graph-theoretic approach to privacy in distributed systems*, J. ACM, 47 (2000), pp. 225 – 243.
12. D. Fulkerson and O. Gross, *Incidence matrices and interval graphs*, Pacific Journal of Math., 15 (1965), pp. 835–855.
13. M. R. Garey, R. L. Graham, D. S. Johnson, and D. E. Knuth, *Complexity results for bandwidth minimization*, SIAM J. Appl. Math., 34 (1978), pp. 477–495.
14. F. Gavril, *The intersection graphs of subtrees in trees are exactly the chordal graphs*, J. Combin. Theory Ser. B, 16 (1974), pp. 47–56.
15. L. M. Kirousis and C. H. Papadimitriou, *Searching and pebbling*, Theor. Comp. Sc., 47 (1986), pp. 205–218.
16. A. S. LaPaugh, *Recontamination does not help to search a graph*, J. ACM, 40 (1993), pp. 224–245.
17. C. G. Lekkerkerker and J. C. Boland, *Representation of a finite graph by a set of intervals on the real line*, Fund. Math, 51 (1962), pp. 45–64.
18. F. S. Makedon, C. H. Papadimitriou, and I. H. Sudborough, *Topological bandwidth*, SIAM J. Alg. Disc. Meth., 6 (1985), pp. 418–444.
19. F. S. Makedon and I. H. Sudborough, *On minimizing width in linear layouts*, Disc. Appl. Math., 23 (1989), pp. 201–298.
20. A. Parra and P. Scheffler, *Treewidth equals bandwidth for AT-free claw-free graphs*, Technical Report 436/1995, Technische Universität Berlin, Fachbereich Mathematik, Berlin, Germany, 1995.
21. F. S. Roberts, *Indifference graphs*, in Proof Techniques in Graph Theory, F. Harary, ed., Academic Press, 1969, pp. 139 – 146.
22. N. Robertson and P. D. Seymour, *Graph minors — a survey*, in Surveys in Combinatorics, I. Anderson, ed., Cambridge Univ. Press, 1985, pp. 153–171.
23. A. L. Rosenberg and I. H. Sudborough, *Bandwidth and pebbling*, Computing, 31 (1983), pp. 115–139.