Bandwidth on AT-free graphs*

Petr Golovach[1] Pinar Heggernes[1] Dieter Kratsch[2]
Daniel Lokshtanov[1] Daniel Meister[1] Saket Saurabh[1]

[1] Department of Informatics, University of Bergen, N-5020 Bergen, Norway.
{petr.golovach, pinar.heggernes, daniel.lokshtanov, daniel.meister, saket.saurabh}@ii.uib.no
[2] Laboratoire d'Informatique Théorique et Appliquée, Université Paul Verlaine - Metz, 57045 Metz Cedex 01, France. kratsch@univ-metz.fr

Abstract. We study the classical Bandwidth problem from the viewpoint of parameterized algorithms. In the Bandwidth problem we are given a graph $G = (V, E)$ together with a positive integer k, and asked whether there is an bijective function $\beta : \{1, \ldots, n\} \to V$ such that for every edge $uv \in E$, $|\beta^{-1}(u) - \beta^{-1}(v)| \leq k$. The problem is notoriously hard, and it is known to be NP-complete even on very restricted subclasses of trees. The best known algorithm for Bandwidth for small values of k is the celebrated algorithm by Saxe [*SIAM Journal on Algebraic and Discrete Methods, 1980*], which runs in time $2^{\mathcal{O}(k)}n^{k+1}$. In a seminal paper, Bodlaender, Fellows and Hallet [*STOC 1994*] ruled out the existence of an algorithm with running time of the form $f(k)n^{\mathcal{O}(1)}$ for any function f even for trees, unless the entire W-hierarchy collapses.

We initiate the search for classes of graphs where Bandwidth is fixed parameter tractable (FPT), that is, solvable in time $f(k)n^{\mathcal{O}(1)}$ for some function f. In this paper we present an algorithm with running time $2^{\mathcal{O}(k \log k)}n^2$ for Bandwidth on AT-free graphs, a well-studied graph class that contains interval, permutation, and cocomparability graphs. Our result is the first non-trivial FPT algorithm for Bandwidth on a graph class where the problem remains NP-complete.

1 Introduction

The *bandwidth* of a graph G is the smallest integer b such that there is an bijective function $\beta : \{1, \ldots, n\} \to V$, also called a layout for G, such that for every edge $uv \in E$, $|\beta^{-1}(u) - \beta^{-1}(v)| \leq b$. Given a graph G and an integer k, Bandwidth asks whether the bandwidth of G is at most k. The problem arises in sparse matrix computations, where given an $n \times n$ matrix A and an integer k, the goal is to decide whether there is a permutation matrix P such that PAP^T is a matrix whose all non-zero entries lie within the k diagonals on either side of the main diagonal. Standard matrix operations like inversion and multiplication as well as Gaussian elimination can be sped up considerably if the input matrix A can be transformed into a matrix PAP^T of small bandwidth [10].

Bandwidth is one of the most well-known and extensively studied graph layout problems [9]. The Bandwidth problem is NP-complete [21] and remains NP-complete even on very restricted subclasses of trees, like caterpillars of hair length at most 3 [18]. Furthermore, the bandwidth of a graph is NP-hard to approximate within a constant factor for trees [2]. Polynomial-time algorithms for the exact computation of bandwidth are known

* This work is supported by the Research Council of Norway.

for a few graph classes including caterpillars with hair length at most 2 [1], cographs [23], interval graphs [14] and bipartite permutation graphs [12]. A classical algorithm by Saxe [22] solves Bandwidth in time $2^{\mathcal{O}(k)}n^{k+1}$, which is polynomial when k is a constant. However, as the value of k grows, the exponent of the polynomial grows with it. A natural question is whether Bandwidth can be solved in time $f(k)n^c$ where c is a constant independent of k. This amounts to asking whether Bandwidth is *fixed parameter tractable*.

Parameterized complexity is a two-dimensional generalization of "P vs. NP" where, in addition to the overall input size n, one studies how a secondary measurement that captures additional relevant information affects the computational complexity of the problem in question. Parameterized decision problems are defined by specifying the input, the parameter and the question to be answered. The two-dimensional analogue of P is solvability within a time bound of $f(k)n^c$, where n is the total input size, k is the parameter, f is some computable function, and c is a constant that does not depend on k or n. A parameterized problem that can be solved in such time is termed *fixed-parameter tractable* (FPT). There is a hierarchy of intractable parameterized problem classes above FPT, the main ones being:

$$\mathrm{FPT} \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \cdots \subseteq \mathrm{W}[P] \subseteq \mathrm{XP}\,.$$

The principal analogue of the classical intractability class NP is W[1], which is a strong analogue, because a fundamental problem complete for W[1] is the k-Step Halting Problem for Nondeterministic Turing Machines with unlimited nondeterminism and alphabet size — this completeness result provides an analogue of Cook's Theorem in classical complexity. Thus a parameterized problem that is hard for W[1] is unlikely to be fixed parameter tractable. For general background on the theory see the textbooks by Downey and Fellows [7], Flum and Grohe [11] and Niedermeier [19].

In a seminal paper, Bodlaender, Fellows and Hallet showed that Bandwidth is hard for W[t] for every $t \geq 1$, even for trees [3]. This rules out the existence of an FPT algorithm for Bandwidth unless the entire W-hierarchy collapses. The hardness result in [3] indicates that the tractable cases for Bandwidth seem to be few and far between. Here, we initiate the search for classes of graphs where Bandwidth is fixed parameter tractable.

For the graph classes for which polynomial time algorithms are known, it has been proved that Bandwidth becomes NP-complete (or its complexity remains unknown) on slightly larger graph classes. Therefore it is natural to investigate the *parameterized* complexity of Bandwidth on these larger classes of graphs. In this paper we present an algorithm with running time $2^{\mathcal{O}(k \log k)}n^2$ for Bandwidth on AT-free graphs. A graph is *AT-free* if for every triple of pairwise non-adjacent vertices, the neighborhood of one of them separates the two others. The class of AT-free graphs contains various well-known graph classes, like interval, permutation, trapezoid, and cocomparability graphs [4]. While Bandwidth is polynomial-time solvable on interval graphs [14] and well-studied subclasses of permutation graphs [23, 12] it is NP-complete on cocomparability graphs and hence on AT-free graphs [16]. For permutation graphs, the complexity of Bandwidth is a well-known open problem. Most natural superclasses of AT-free graphs contain trees, and thus the hardness result in [3] rules out an FPT algorithm for Bandwidth on these

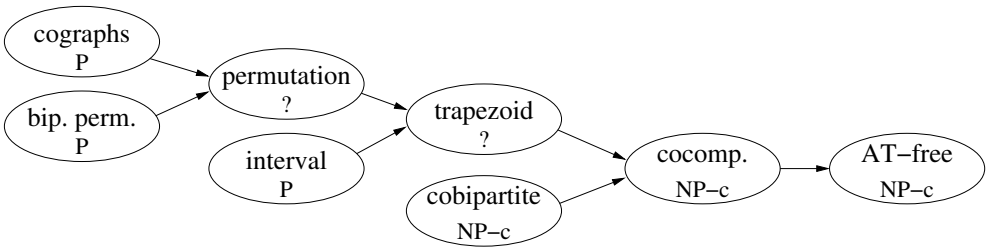

Fig. 1. A graph class inclusion diagram with the classical complexity of BANDWIDTH on these graph classes.

classes. Thus our FPT algorithm on AT-free graphs essentially settles the parameterized complexity of BANDWIDTH on the chain of natural graph classes above the polynomial cases (Figure 1).

Our algorithm is based on structural properties of AT-free graphs and their relation to interval graphs. The principal idea is to combine layouts for small subgraphs of the input graph to a layout for the whole graph. This approach can be described as sweeping a small window over a layout. For arbitrary graphs, there is no information about the relationship between the vertices in the window. For AT-free graphs, however, we are able to show that we can restrict to windows of vertices that have small distance to a common vertex in the input AT-free graph. This enables us to restrict the number of considered windows and to establish the claimed FPT running time of our algorithm.

2 Definitions and notation

In this paper, we mainly consider simple finite undirected graphs. However, as an auxiliary structure, we also define a directed graph. By "graph", we always mean undirected graph, and by "digraph", we mean a directed graph.

For a graph $G = (V, E)$, we denote the vertex and edge set of G by respectively $V = V(G)$ and $E = E(G)$, with $n = |V|$. Edges of a graph are denoted as uv, and if uv is an edge of G, we call u and v *adjacent*. The *neighborhood* of a vertex u is the set of vertices that are adjacent to u and is denoted as $N_G(u)$. For two vertices $u, v \in V$, a *u, v-path of G of length r* is a sequence $(u_0, \ldots, u_r)$ of distinct vertices where $u_0 = u$, $u_r = v$ and $u_i u_{i+1} \in E$ for $0 \leq i < r$. The *distance of u and v in G*, denoted by $\mathrm{d}_G(u, v)$, is the smallest length of a u, v-path in G. For a vertex u of G and an integer $\ell \geq 1$, the *ball around u of radius ℓ*, $B_G(u, \ell)$, is the set of vertices different from u that are at distance at most ℓ to u in G. Formally, $B_G(u, \ell) = \{x \neq u : \mathrm{d}_G(u, x) \leq \ell\}$. A graph G is *connected* if there is a u, v-path in G for every vertex pair u, v. A set of vertices is a *clique* if its vertices are pairwise adjacent. The *square* of G is $G^2 = (V, \{uv : 1 \leq \mathrm{d}_G(u, v) \leq 2\})$.

For a set $S \subseteq V$, the subgraph of G *induced* by S, denoted by $G[S]$, has vertex set S and all edges of G that have both their endpoints in S. By $G \setminus S$, we denote $G[V \setminus S]$. A graph G is a *subgraph* (not necessarily induced) of a graph H if $V(G) \subseteq V(H)$ and $E(G) \subseteq E(H)$. In this case, we write $G \subseteq H$. A *connected component* of G is a maximal connected induced subgraph of G.

For a graph G, a *layout* β (or *vertex ordering*) is a bijective function from $\{1,\ldots,n\}$ to V. We also write β as $\langle\beta(1),\ldots,\beta(n)\rangle$. For a vertex pair u,v of G, the *distance* between u and v in β is $\mathrm{d}_\beta(u,v)=|\beta^{-1}(u)-\beta^{-1}(v)|$. We write $u\preccurlyeq_\beta v$ if $\beta^{-1}(u)\leq\beta^{-1}(v)$ and $u\prec_\beta v$ if $\beta^{-1}(u)<\beta^{-1}(v)$. The *leftmost* and *rightmost* vertex in β are respectively $\beta(1)$ and $\beta(n)$. For an integer $k\geq 1$, we call β a *k-layout* for G if for every edge uv of G, $d_\beta(u,v)\leq k$.

The *bandwidth* of G, $\mathrm{bw}(G)$, is the smallest integer b such that G has a b-layout. A *minimum bandwidth layout* for G is a k-layout for G with $k=\mathrm{bw}(G)$. Observe that for any two graphs G and H with $G\subseteq H$, $\mathrm{bw}(G)\leq\mathrm{bw}(H)$.

If β is a layout for G and ℓ is between 1 and n, then every i between 1 and $n-\ell+1$ defines an *ℓ-window of β*: $\langle\beta(i),\ldots,\beta(i+\ell-1)\rangle$. Informally, an ℓ-window is a portion of β containing exactly ℓ consecutive vertices. Vertices $\beta(i)$ and $\beta(i+\ell-1)$ are called respectively *left* and *right vertex* of the ℓ-window.

Layouts of sets of vertices are defined analogously to layouts for graphs. Let $U,U'\subseteq V$ and let β and β' be layouts of respectively U and U'. Let $U\cap U'\neq\emptyset$ and let $1\leq t\leq|U|$ be smallest with $\beta(t)\in U'$. If $\beta(t+i)=\beta'(i+1)$ for all $0\leq i\leq|U|-t$ then $\beta\bullet\beta'$ is the layout $\langle\beta(1),\ldots,\beta(|U|),\beta'(|U'|-t+2),\ldots,\beta'(|U'|)\rangle$; otherwise, if the condition is violated, $\beta\bullet\beta'$ is not defined. Informally, $\beta\bullet\beta'$ is the concatenation of β and β' by overlapping in the common part. Note that the $\bullet$ operator satisfies the associativity law.

In this paper, we study AT-free graphs and subclasses of AT-free graphs. A set $\{u,v,w\}$ of three pairwise non-adjacent vertices of a graph is called *asteroidal triple*, *AT* for short, if for any pair of the vertices there is a path between these two vertices avoiding the neighborhood of the third vertex. A graph that has no asteroidal triple is called *AT-free*. For more information on structural properties of AT-free graphs we refer to [4, 5].

Finally, a *cycle* in a digraph G is a sequence $(u_1,\ldots,u_r)$ of distinct vertices where (u_r,u_1) and $(u_1,u_2),\ldots,(u_{r-1},u_r)$ are arcs of G. A digraph without cycles is called *acyclic*.

3 Preliminary and auxiliary results on bandwidth

The following observation will be important for the running time of our algorithm. The result is easy to establish from the fact that a graph of bandwidth at most k cannot have a vertex of degree more than $2k$.

Lemma 1. *For an arbitrary graph $G=(V,E)$, $|E|\leq\mathrm{bw}(G)\cdot|V|$.*

A graph H is an *interval graph* if its vertices can be assigned closed intervals of the real line such that two vertices of H are adjacent if and only if the assigned intervals have a non-empty intersection. For an arbitrary graph G, an *interval completion* of G is an interval graph H on vertex set $V(G)$ with $E(G)\subseteq E(H)$. If there is no interval completion H' of G with $E(H')\subset E(H)$ then H is a *minimal* interval completion of G.

An interval graph is a *proper interval graph* if there is an interval representation of it where no interval completely contains another interval. For an arbitrary graph G, a *proper interval completion* of G is a proper interval graph H on vertex set $V(G)$ with $E(G)\subseteq E(H)$.

Lemma 2 ([13]). *For any integer $k \geq 0$, a graph G has a proper interval completion H of maximum clique size at most $k+1$ if and only if $bw(G) \leq k$.*

From Lemma 2 it follows that any minimum bandwidth layout β for G defines a proper interval completion of it in the following way. For every edge uv of G, make the set of vertices between (including) u and v in β into a clique by adding the necessary missing edges.

Lemma 3. *For every graph G, there is a minimal interval completion H of G with* $\mathrm{bw}(G) = \mathrm{bw}(H)$.

Proof. Let β be a minimum bandwidth layout for G. Then, β defines a proper interval completion H' of G, where $\mathrm{bw}(G) = \mathrm{bw}(H')$. Since H' is an interval graph, there is a minimal interval completion H of G with $G \subseteq H \subseteq H'$ and $\mathrm{bw}(H) = \mathrm{bw}(G)$. ∎

An alternative characterization of interval graphs is that a graph G is an interval graph if and only if G has a vertex ordering σ such that for all vertex triples u, v, w of G with $u \prec_\sigma v \prec_\sigma w$, $uw \in E(G)$ implies $vw \in E(G)$ [20]. Such a vertex ordering is called an *interval ordering*.

Theorem 1 ([8]). *Let H be an interval graph with interval ordering σ. There is a minimum bandwidth layout β for H such that for every pair u, v of non-adjacent vertices of H, $u \prec_\sigma v$ implies $u \prec_\beta v$.*

Lemma 4. *Let H be an interval graph. There is a minimum bandwidth layout β for H with the property: for every vertex pair u, v of H, $\mathrm{d}_\beta(u, v) \geq \mathrm{d}_H(u, v) - 2$.*

Proof. Let σ be an interval ordering for H. Let β be a minimum bandwidth layout for H with the property of Theorem 1 with respect to σ. We show that β satisfies the lemma. Let u, v be a vertex pair of H. If $\mathrm{d}_H(u, v) \leq 3$, the lemma trivially holds. Let $\mathrm{d}_H(u, v) \geq 4$. Without loss of generality, we can assume $u \prec_\sigma v$. Let $(x_0, \ldots, x_r)$ be a shortest u, v-path of H, where $x_0 = u$ and $x_r = v$. If there is $1 \leq i \leq r-2$ with $x_{i-1} \prec_\sigma v \prec_\sigma x_i$ then v is adjacent to x_i by the properties of interval orderings, contradicting the choice of the shortest path. If there is $1 \leq i \leq r-1$ with $x_i \prec_\sigma u \prec_\sigma x_{i+1}$ then u and x_{i+1} are adjacent, which again gives a contradiction. Thus, $u \prec_\sigma x_i \prec_\sigma v$ for all $1 \leq i \leq r-2$. Since $x_2, \ldots, x_{r-2}$ are non-adjacent to u and v, Theorem 1 for β implies that $u \prec_\beta x_i \prec_\beta v$ for all $2 \leq i \leq r-2$. Hence, $\mathrm{d}_\beta(u, v) \geq r - 2 = \mathrm{d}_H(u, v) - 2$. ∎

Lemma 5. *Let G be a connected graph and β a k-layout for G, where $k \geq 1$. Let U be the vertices of a $(k+2)$-window of β with a and b the respectively left and right vertex. Then, $G \setminus U$ has at most $2k$ connected components, and for every connected component C of $G \setminus U$, the vertices of C are either all to the left of a or all to the right of b in β.*

Proof. Since there is no edge uv of G with $u \prec_\beta a \prec_\beta b \prec_\beta v$ by β being a k-layout and $\mathrm{d}_\beta(a, b) = k+1$, no connected component of $G \setminus U$ has vertices to the left of a and to the right of b in β. Since G is a connected graph, every connected component of $G \setminus U$ has a vertex with a neighbor in U. Thus, every connected component of $G \setminus U$ has a vertex to the left of a at distance (with respect to β) at most k to a or to the right of b at distance at most k to b in β. Hence there can be at most $2k$ connected components in $G \setminus U$. ∎

Lemma 6. *Let G be a graph and let β be a k-layout for G, where $k \geq 1$. For every vertex a of G and every integer $\ell \geq 1$, $|B_G(a,\ell)| \leq 2\ell \cdot k$.*

Proof. Let c and d be the respectively leftmost and rightmost vertex from $B_G(a,\ell)$ in β. Since $\mathrm{d}_\beta(c,a) \leq \ell \cdot k$ and $\mathrm{d}_\beta(a,d) \leq \ell \cdot k$ it follows that $\mathrm{d}_\beta(c,d) \leq 2\ell \cdot k$. Since all vertices from $B_G(a,\ell)$ are between c and d in β, the bound of the lemma follows. Note that by definition $a \notin B_G(a,\ell)$. ∎

4 Bandwidth of AT-free graphs

Combining the results of [16] and [17], we obtain the following:

Theorem 2 ([16, 17]). *Let G be an AT-free graph. For every minimal interval completion H of G, $H \subseteq G^2$.*

We use this fact to restrict the number of $(k+2)$-windows to be considered by our algorithm.

Lemma 7. *For a connected AT-free graph G with $\mathrm{bw}(G) \leq k$, there is a k-layout β that satisfies the following. Let U be the vertices of a $(k+2)$-window of β with left vertex a. For every vertex $x \in U$, $\mathrm{d}_G(a,x) \leq 2k+6$.*

Proof. Let H be a minimal interval completion of G with $\mathrm{bw}(H) = \mathrm{bw}(G)$; H exists due to Lemma 3. By Theorem 2 $\mathrm{d}_H(u,v) \geq \frac{1}{2} \cdot \mathrm{d}_G(u,v)$ for every vertex pair u, v of G. Let β be a minimum bandwidth layout for H with the property of Lemma 4. Then, for every vertex pair u, v of G, $\mathrm{d}_\beta(u,v) + 2 \geq \mathrm{d}_H(u,v) \geq \frac{1}{2}\mathrm{d}_G(u,v)$, i.e., $\mathrm{d}_\beta(u,v) \geq \frac{1}{2}\mathrm{d}_G(u,v) - 2$. Let U be the vertices of a $(k+2)$-window of β with left vertex a. Since $\mathrm{d}_\beta(a,x) \leq k+1$ for every $x \in U$, it follows that $\mathrm{d}_G(a,x) \leq 2k+6$. ∎

We construct a digraph to encode all feasible k-layouts. Let $k \geq 1$ and let $G = (V,E)$ be a connected AT-free graph on $n \geq k+3$ vertices. Note that every graph on at most $k+1$ vertices has bandwidth at most k, and every graph on $k+2$ vertices has bandwidth at most k if and only if the graph has a pair of non-adjacent vertices. For a vertex u of G, we say that *u has few close neighbors* if $|B_G(u, 2k+6)| \leq 2k \cdot (2k+6)$. Let a be a vertex of G that has few close neighbors. An *a-bag* is a tuple $(\mathcal{C}_1, \gamma, \mathcal{C}_2)$ where $\mathcal{C}_1, \mathcal{C}_2 \subseteq V$ such that $U = V \setminus (\mathcal{C}_1 \cup \mathcal{C}_2)$ and $\mathcal{C}_1, \mathcal{C}_2, \gamma$ have the following properties:

- $\{a\} \subseteq U \subseteq B_G(a, 2k+6) \cup \{a\}$ and $|U| = k+2$;
- γ is a layout of U where a is the leftmost vertex;
- let b be the rightmost vertex in γ, a has no neighbor in $\mathcal{C}_2 \cup \{b\}$, and b has no neighbor in $\mathcal{C}_1 \cup \{a\}$;
- $G \setminus U$ has at most $2k$ connected components, and
  for every connected component C of $G \setminus U$, either $V(C) \subseteq \mathcal{C}_1$ or $V(C) \subseteq \mathcal{C}_2$.

Lemma 8. *For every vertex a of G where a has few close neighbors, the number of a-bags is at most $\binom{2k\cdot(2k+6)}{k+1} \cdot (k+1)! \cdot 2^{2k} \leq 2^{\mathcal{O}(k \log k)}$.*

Proof. Let a be a vertex of G with few close neighbors. Since $|B_G(a, 2k+6)| \leq 2k \cdot (2k+6)$, there are at most $\binom{2k\cdot(2k+6)}{k+1}$ subsets of $B_G(a, 2k+6)$ of size $k+1$. Let U be such a subset. There are $(k+1)!$ possible layouts of U. And since $G \setminus (U \cup \{a\})$ has at most $2k$ connected components according to the definition of an a-bag, there are at most 2^{2k} different partitions of the connected components of $G \setminus (U \cup \{a\})$ into a left and a right set. ∎

The auxiliary digraph that we will define has a vertex for every bag and arcs between bags representing the fact that one bag can follow another bag in a minimum bandwidth layout. We now describe the construction of the auxiliary digraph $\mathrm{aux}(G, k)$. The digraph $\mathrm{aux}(G, k)$ has a vertex for every a-bag where $a \in V$ has few close neighbors, a source vertex S and a sink vertex T. The vertices of $\mathrm{aux}(G, k)$ corresponding to a-bags are labeled with these a-bags. For two vertices u and u' of $\mathrm{aux}(G, k)$ such that $u \neq u'$ and $u, u' \notin \{S, T\}$ with u labeled with an a-bag $(\mathcal{C}_1, \gamma, \mathcal{C}_2)$ and u' labeled with an a'-bag $(\mathcal{C}'_1, \gamma', \mathcal{C}'_2)$, with $U = V \setminus (\mathcal{C}_1 \cup \mathcal{C}_2)$ and $U' = V \setminus (\mathcal{C}'_1 \cup \mathcal{C}'_2)$, the digraph contains the arc (u, u') if and only if the bags satisfy one of the following two conditions:

1) $\mathcal{C}_1 \subset \mathcal{C}'_1 \subset (\mathcal{C}_1 \cup U)$ and $\mathcal{C}'_2 = \emptyset$ and $\gamma \bullet \gamma'$ is k-layout for $G[U \cup U']$
2) $|U \cap U'| = 1$ and $\mathcal{C}'_1 = \mathcal{C}_1 \cup (U \setminus U')$ and $\gamma \bullet \gamma'$ is k-layout for $G[U \cup U']$.

Note that, by the definition of the $\bullet$ operator for layouts, the end of layout γ is equal to the beginning of layout γ'. For condition 2, this means that the rightmost vertex in γ is equal to the leftmost vertex in γ'. To complete the definition of $\mathrm{aux}(G, k)$, add all arcs (S, u) with u labeled with a bag of the type $(\emptyset, \gamma, \mathcal{C}_2)$ and all arcs (u', T) with u' labeled with a bag of the type $(\mathcal{C}'_1, \gamma', \emptyset)$.

Lemma 9. *The auxiliary digraph* $\mathrm{aux}(G, k)$ *is acyclic.*

Proof. Suppose that $\mathrm{aux}(G, k)$ contains a cycle $(u^{(1)}, \ldots, u^{(r)})$. Let $(\mathcal{C}_1^{(i)}, \gamma^{(i)}, \mathcal{C}_2^{(i)})$ be the bag that $u^{(i)}$ is labeled with, for $1 \leq i \leq r$. According to the definition of $\mathrm{aux}(G, k)$, it holds that $\mathcal{C}_1^{(1)} \subset \cdots \subset \mathcal{C}_1^{(r)} \subset \mathcal{C}_1^{(1)}$. This yields a contradiction. ∎

Lemma 10. *The auxiliary digraph* $\mathrm{aux}(G, k)$ *has an* S, T*-path if and only if* $\mathrm{bw}(G) \leq k$*. Furthermore, every* S, T*-path of* $\mathrm{aux}(G, k)$ *defines a* k*-layout for* G*.*

Proof. We show the two implications of the statement separately. We first show that every S, T-path of $\mathrm{aux}(G, k)$ defines a k-layout for G. Let $(u^{(0)}, u^{(1)}, \ldots, u^{(r+1)})$ be an S, T-path of $\mathrm{aux}(G, k)$. Note that $u^{(0)} = S$ and $u^{(r+1)} = T$. For every $1 \leq i \leq r$, let $(\mathcal{C}_1^{(i)}, \gamma^{(i)}, \mathcal{C}_2^{(i)})$ be the bag that $u^{(i)}$ is labeled with. We show that $\beta = \gamma^{(1)} \bullet \cdots \bullet \gamma^{(r)}$ is a k-layout for G. First, we show that β is a layout for G, i.e., every vertex of G appears exactly once in β. For every $1 \leq i \leq r$, let $U^{(i)} = V \setminus (\mathcal{C}_1^{(i)} \cup \mathcal{C}_2^{(i)})$. According to the definition of $\mathrm{aux}(G, k)$, $\mathcal{C}_1^{(1)} = \mathcal{C}_2^{(r)} = \emptyset$ and $\mathcal{C}_1^{(i+1)} \subseteq \mathcal{C}_1^{(i)} \cup U^{(i)}$ for all $1 \leq i \leq r-1$. It follows that $V \setminus (U^{(1)} \cup \cdots \cup U^{(r-2)}) \subseteq U^{(r-1)} \cup U^{(r)}$. Thus, for every vertex x of G, there is $1 \leq i \leq r$ with $x \in U^{(i)}$ and $\gamma^{(i)}$ places x. If there are $x \in V$ and $1 \leq i < j \leq r$ such that $x \in U^{(i)} \cap U^{(j)}$ then the definition of arcs of $\mathrm{aux}(G, k)$ ensures that $j = i + 1$. Then, $\gamma^{(i)} \bullet \gamma^{(i+1)}$ being a layout for $G[U^{(i)} \cup U^{(i+1)}]$ guarantees that x appears exactly once.

Hence, β is a layout for G. Now, let uv be an edge of G. If there is $1 \leq i \leq r$ such that $u, v \in U^{(i)}$ then $\mathrm{d}_\beta(u,v) = \mathrm{d}_{\gamma^{(i)}}(u,v) \leq k$ due to the definition of a bag (leftmost and rightmost vertex in $\gamma^{(i)}$ are non-adjacent). Now, assume that there is no such U_i. Then, without loss of generality, there are $1 \leq i < j \leq r$ such that $u \in U^{(i)}$ and $v \in U^{(j)}$. With regard to overlapping of consecutive partial layouts, we choose i largest possible and j smallest possible. Suppose that $j > i+1$. It follows that $u \in \mathcal{C}_1^{(i+1)}$ and $v \in \mathcal{C}_2^{(i+1)}$. Since u and v are adjacent, there is a connected component of $G \setminus U^{(i+1)}$ with vertices in $\mathcal{C}_1^{(i+1)}$ and $\mathcal{C}_2^{(i+1)}$, thus contradicting the definition of a bag. Hence, $j = i+1$, and uv is an edge of $G[U^{(i)} \cup U^{(i+1)}]$. Since $\gamma^{(i)} \bullet \gamma^{(i+1)}$ is a k-layout for $G[U^{(i)} \cup U^{(i+1)}]$ according to the definition of $\mathrm{aux}(G,k)$ and the existence of arc $(u^{(i)}, u^{(i+1)})$, $\mathrm{d}_\beta(u,v) = \mathrm{d}_{\gamma^{(i)} \bullet \gamma^{(i+1)}}(u,v) \leq k$. Thus, β is a k-layout for G. In particular, if $\mathrm{aux}(G,k)$ has an S,T-path then $\mathrm{bw}(G) \leq k$.

For the converse, let $\mathrm{bw}(G) \leq k$. Let β be a minimum bandwidth layout for G with the property of Lemma 7. Then, β is also a k-layout for G. We partition β into sublayouts. Let $r = \lceil \frac{n-1}{k+1} \rceil$. Let $a^{(i)} = \beta((i-1)(k+1)+1)$ for all $1 \leq i \leq r$ and let $a^{(r+1)} = \beta(n-k-1)$ and $a^{(r+2)} = \beta(n)$. Let $U^{(i)}$ be the set of vertices x of G with $a^{(i)} \preccurlyeq_\beta x \preccurlyeq_\beta a^{(i+1)}$ for all $1 \leq i \leq r+1$ where $i \neq r$. Finally, for all $1 \leq i \leq r+1$ where $i \neq r$, let $\gamma^{(i)}$ be the $(k+2)$-window of β with left vertex $a^{(i)}$. Note that $\gamma^{(i)}$ is a layout of $U^{(i)}$. It holds that $U^{(i)} \cap U^{(i+1)} = \{a^{(i+1)}\}$ for $1 \leq i \leq r-2$ and $U^{(r-1)} \cap U^{(r+1)} \neq \emptyset$. Furthermore, $\beta = \gamma^{(1)} \bullet \cdots \bullet \gamma^{(r-1)} \bullet \gamma^{(r+1)}$. Since β is a k-layout for G and by the definition of $a^{(1)}, \ldots, a^{(r+2)}$, $a^{(i)}a^{(i+1)} \notin E$ for all $1 \leq i \leq r+1$ where $i \neq r$. We show that $C^{(i)} = (\{x : x \prec_\beta a^{(i)}\}, \gamma^{(i)}, \{x : a^{(i+1)} \prec_\beta x\})$ is an $a^{(i)}$-bag for $1 \leq i \leq r+1$ where $i \neq r$. By the definition, $|U^{(i)}| = k+2$ and $\gamma^{(i)}$ is a layout of $U^{(i)}$ with $a^{(i)}$ the leftmost vertex. And with β being a k-layout for G, $a^{(i)}$ has no neighbor in $\{x : a^{(i+1)} \prec_\beta x\}$) and $a^{(i+1)}$ has no neighbor in $\{x : x \prec_\beta a^{(i)}\}$. The connected components condition directly follows from Lemma 5. It remains to verify that $a^{(i)}$ has few close neighbors and that $U^{(i)} \subseteq B_G(a^{(i)}, 2k+6) \cup \{a^{(i)}\}$. By Lemma 7, the latter condition holds, and by Lemma 6, $a^{(i)}$ has indeed few close neighbors. Hence, $C^{(i)}$ is an $a^{(i)}$-bag, and there is a vertex $u^{(i)}$ of $\mathrm{aux}(G,k)$ that is labeled with $C^{(i)}$. With our definitions and the fact that $\gamma^{(i)} \bullet \gamma^{(i+1)}$ for every $1 \leq i \leq r+1$ where $i \neq r$ is a k-layout for $G[U^{(i)} \cup U^{(i+1)}]$, it follows that $(u^{(i)}, u^{(i+1)})$ is an arc of $\mathrm{aux}(G,k)$. Since $(S, u^{(1)})$ and $(u^{(r+1)}, T)$ are also arcs of $\mathrm{aux}(G,k)$, we conclude that $(S, u^{(1)}, \ldots, u^{(r-1)}, u^{(r+1)}, T)$ is an S,T-path of $\mathrm{aux}(G,k)$. Hence, $\mathrm{aux}(S,T)$ contains an S,T-path, and this completes the proof of the lemma. ∎

For the algorithm in the proof of the following theorem, we assume the input graph to be given in adjacency list representation.

Theorem 3. *For given an AT-free graph G and an integer $k \geq 1$, it can be decided in $2^{\mathcal{O}(k \log k)} n^2$ time whether* $\mathrm{bw}(G) \leq k$.

Proof. Let $k \geq 1$. Let $G = (V,E)$ be an AT-free graph. Since $\mathrm{bw}(G) \leq k$ if and only if $\mathrm{bw}(C) \leq k$ for every connected component C of G, the algorithm to be described below is to run on every connected component of G. By these considerations, we can assume that the input graph G to our algorithm is connected. The algorithm is: if $|E| > k \cdot |V|$ or if a vertex has more than $2k$ neighbors then reject, otherwise, compute $\mathrm{aux}(G,k)$, with

the distinguished source and sink vertices S and T, and accept if and only if $\mathrm{aux}(G,k)$ contains an S,T-path. Correctness follows from Lemmata 1, 6, and 10.

The running time of the algorithm is mainly determined by the generation of $\mathrm{aux}(G,k)$. Let a be a vertex of G. Since every vertex of G has at most $2k$ neighbors, it can be checked in time $\mathcal{O}(k^3)$ whether vertex a has few neighbors and, if so, to compute $B_G(a,2k+6)$. In time $2^{\mathcal{O}(k\log k)}n$, all a-bags can be listed due to Lemma 8. Determining the left side and right side connected components of a single bag takes time $\mathcal{O}(kn)$, and writing down a single a-bag takes time $\mathcal{O}(n)$. In total, the vertices of $\mathrm{aux}(G,k)$ together with the labels can be listed in $2^{\mathcal{O}(k\log k)}n^2$ time. Note that $\mathrm{aux}(G,k)$ has $2^{\mathcal{O}(k\log k)}n$ vertices. Let u be a vertex of $\mathrm{aux}(G,k)$ labeled with an a-bag $(\mathcal{C}_1,\gamma,\mathcal{C}_2)$. If $\mathcal{C}_2\neq\emptyset$, then all arcs from u go to vertices x that are labeled with a b-bag, where b is the rightmost vertex in γ in case $|\mathcal{C}_2|\geq k+1$ or is uniquely determined in γ from the size of $\mathcal{C}_2$. Hence, for every vertex of $\mathrm{aux}(G,k)$ there are at most $2^{\mathcal{O}(k\log k)}$ vertices to check, and each check takes $\mathcal{O}(n)$ time because of the left and right side connected components. Hence, in overall $2^{\mathcal{O}(k\log k)}n^2$ time, $\mathrm{aux}(G,k)$ can be generated, and it has $2^{\mathcal{O}(k\log k)}n$ vertices and arcs. Since verifying the existence of an S,T-path in $\mathrm{aux}(G,k)$ takes $2^{\mathcal{O}(k\log k)}n$ time, this concludes the running time analysis for our algorithm. ∎

5 Concluding remarks

Our algorithm guesses layouts of small sets of vertices and concatenates them to create a layout for the whole graph. The running time of the algorithm relies on the fact that the considered sets of vertices are of special type, namely that it suffices to consider vertices that are at small distance to a common vertex. Correctness of this restriction follows from the result that distances in minimal interval completions of AT-free graphs provide a constant-factor approximation of the distances in the graph. Thus, we can generalize our algorithm to graph classes with the same property. Which other graph class $\mathcal{C}$ has this property: there is constant c such that for every graph G from $\mathcal{C}$ and every minimal interval completion H of G, $\mathrm{d}_G(u,v)\leq c\cdot\mathrm{d}_H(u,v)$ for all vertex pairs u,v of G?

Classes of graphs of bounded diameter certainly have this property. These are classes of dense graphs. For such classes of dense graphs, the problem becomes trivial, as we show in the following. For a graph G, the *diameter* of G, denoted by $\mathrm{diam}(G)$, is the maximum distance between a vertex pair of G.

Lemma 11. *For an arbitrary connected graph $G=(V,E)$, $|V|\leq 1+\mathrm{diam}(G)\cdot\mathrm{bw}(G)$.*

Proof. Let β be a minimum bandwidth layout for G. Let a and b be the respectively leftmost and rightmost vertex in β. Then, $|V|-1=\mathrm{d}_\beta(a,b)\leq\mathrm{diam}(G)\cdot\mathrm{bw}(G)$. ∎

In other words, for a class of graphs of bounded diameter, there is only a finite number of graphs of bounded bandwidth. Thus, for such graph classes, deciding whether $\mathrm{bw}(G)\leq k$ for given graph G is trivial when k is fixed. If k is part of the input, we can apply the currently best known exact algorithm for computing bandwidth by Cygan and Pilipczuk, with running time $\mathcal{O}(4.473^n)$ [6], and obtain a $\mathcal{O}(4.473^{dk})$-time algorithm for deciding whether $\mathrm{bw}(G)\leq k$ for a given (connected) graph G with $\mathrm{diam}(G)\leq d$ and integer k.

Examples of such graphs are P_{d+1}-free graphs, in particular split graphs and cobipartite graphs, which are well-studied classes of P_5-free graphs. The bandwidth problem is NP-complete when restricted to split and cobipartite graphs [15, 16].

We conclude the paper with a few concrete open problems.

- Does there exist an FPT algorithm for Bandwidth on AT-free graphs running in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$? An algorithm with this running time would be interesting even for cocomparability graphs.
- Can bandwidth be FPT-approximated on trees? That is, is there an algorithm that given a tree T and integer k, runs in time $f(k)n^{\mathcal{O}(1)}$ and either correctly answers that $\mathrm{bw}(T) > k$ or outputs a $g(k)$-layout for T for some function g.
- What is the parameterized complexity of Bandwidth on caterpillars with hairlength c, for fixed constant $c \geq 3$?

Acknowledgements

We would like to thank Fedor Fomin for insightful discussions around bandwidth.

References

1. S. F. Assmann, 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.*, 2:387–393, 1981.
2. G. Blache, M. Karpinski, and J. Wirtgen. On approximation intractability of the bandwidth problem. Technical report TR98-014, University of Bonn, 1997.
3. H. L. Bodlaender, M. R. Fellows, and M. T. Hallet. Beyond NP-completeness for problems of bounded width (extended abstract): hardness for the W hierarchy. *Proceedings of STOC 1994*, pages 449–458, ACM, 1994.
4. A. Brandstädt, V. B. Le, and J. Spinrad. *Graph Classes: A Survey*. SIAM, Philadelphia, 1999.
5. D. G. Corneil, S. Olariu, and L. Stewart. Asteroidal Triple-Free Graphs. *SIAM J. Disc. Math.*, 10:399–430, 1997.
6. M. Cygan and M. Pilipczuk. Exact and Approximate Bandwidth. *Proceedings of ICALP 2009*, LNCS, 5555:304–315, 2009.
7. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, New York, 1999.
8. P. Fishburn, P. Tanenbaum, and A. Trenk. Linear discrepancy and bandwidth. *Order*, 18:237–245, 2001.
9. M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
10. J. A. George and J. W. H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, New Jersey, 1981.
11. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006.
12. P. Heggernes, D. Kratsch, and D. Meister. Bandwidth of bipartite permutation graphs in polynomial time. *Journal of Disc. Alg.*, to appear.
13. H. Kaplan and R. Shamir. Pathwidth, Bandwidth, and Completion Problems to Proper Interval Graphs with Small Cliques. *SIAM J. Computing*, 25:540–561, 1996.
14. D. J. Kleitman and R. V. Vohra. Computing the bandwidth of interval graphs. *SIAM J. Disc. Math.*, 3:373–375, 1990.
15. T. Kloks, D. Kratsch, Y. Le Borgne, and H. Müller. Bandwidth of Split and Circular Permutation Graphs. *Proceedings of WG 2000*, LNCS, 1928:243–254, 2000.
16. T. Kloks, D. Kratsch, and H. Müller. Approximating the bandwidth for AT-free graphs. *J. Alg.*, 32:41–57, 1999.

17. R. Möhring. Triangulating Graphs Without Asteroidal Triples. *Discrete Applied Mathematics*, 64:281–287, 1996.
18. B. Monien. The Bandwidth-Minimization Problem for Caterpillars with Hair Length 3 is NP-Complete. *SIAM J. Alg. Disc. Meth.*, 7:505–512, 1986.
19. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*, Oxford University Press, 2006.
20. S. Olariu. An optimal greedy heuristic to color interval graphs. *Information Processing Letters*, 37:21–25, 1991.
21. C. Papadimitriou. The NP-completeness of the bandwidth minimization problem. *Computing*, 16:263–270, 1976.
22. J. B. Saxe. Dynamic-Programming Algorithms for Recognizing Small-Bandwidth Graphs in Polynomial Time. *SIAM Journal on Algebraic and Discrete Methods*, 1:363–369, 1980.
23. J. H. Yan. The bandwidth problem in cographs. *Tamsui Oxf. J. Math. Sci.*, 13:31–36, 1997.