On Strong Tree-Breadth

Arne Leitert(✉) and Feodor F. Dragan

Department of Computer Science, Kent State University, Kent, OH, USA
{aleitert,dragan}@cs.kent.edu

Abstract. In this paper, we introduce and investigate a new notion of *strong tree-breadth*. We say that a graph G has *strong tree-breadth* ρ if there is a tree-decomposition T for G such that each bag B of T is equal to the complete ρ-neighbourhood of some vertex v in G, i. e., $B = N_G^\rho[v]$. We show that

- it is NP-complete to determine if a given graph has strong tree-breadth ρ, even for $\rho = 1$;
- if a graph G has strong tree-breadth ρ, then we can find a tree-decomposition for G with tree-breadth ρ in $\mathcal{O}(n^2m)$ time;
- with some additional restrictions, a tree-decomposition with strong breadth ρ can be found in polynomial time;
- some graph classes including distance-hereditary graphs have strong tree-breadth 1.

1 Introduction

Decomposing a graph into a tree is an old concept. It was introduced already by Halin [14]. However, a more popular introduction was given by Robertson and Seymour [15,16]. The idea is to decompose a graph into multiple induced subgraphs, usually called *bags*, where each vertex can be in multiple bags. These bags are combined to a tree in such a way that the following requirements are fulfilled: Each vertex is in at least one bag, each edge is in at least one bag, and, for each vertex, the bags containing it induces a subtree. We will give formal definitions in the next section.

For a given graph, there can be up to exponentially many different tree-decompositions. The easiest is to have only one bag containing the whole graph. To make the concept more interesting, it is necessary to add additional restrictions. The most known is called *tree-width*. A decomposition has *width* ω if each bag contains at most $\omega + 1$ vertices. Then, a graph G has tree-width ω if there is a tree-decomposition for G which has width ω.

In the last years, a new perspective on tree-decompositions was invested. Instead of limiting the number of vertices in each bag, the distance between vertices inside a bag is limited [8,9]. In this paper, we are interested in a variant called *tree-breadth*. It was introduced by Dragan and Köhler in [9]. The *breadth* of a tree-decomposition is ρ, if, for each bag B, there is a vertex v such that each vertex in B has distance at most ρ to v. Accordingly, we say the *tree-breadth* of a graph G is ρ (written as $\mathrm{tb}(G) = \rho$) if there is a tree-decomposition

T-H.H. Chan et al. (Eds.): COCOA 2016, LNCS 10043, pp. 62–76, 2016.
DOI: 10.1007/978-3-319-48749-6_5

for G with breadth ρ and there is no tree-decomposition with smaller breadth. This new concept of tree-breadth played a crucial role in designing an efficient and best to the date approximation algorithm for the well-known tree t-spanner problem (see [9] for details). Recently, Ducoffe et al. [13] have shown that it is NP-complete to determine if a graph has tree-breadth ρ for all $\rho \geq 1$. On the other hand, for a given graph G, a tree-decomposition of breadth at most $3\,\mathrm{tb}(G)$ can be computed in linear time [1].

By definition, a tree-decomposition has breadth ρ if each bag B is the subset of the ρ-neighbourhood of some vertex v, i. e., the set of bags is the set of *subsets* of the ρ-neighbourhoods of *some* vertices. Tree-breadth 1 graphs contain the class of *dually chordal* graphs which can be defines as follows: A graph G is dually chordal if it admits a tree-decomposition T such that, for each vertex v in G, T contains a bag $B = N_G[v]$ [4]. That is, the set of bags in T is the set of *complete* neighbourhoods of *all* vertices.

In this paper, we investigate the case which lays between dually chordal graphs and general tree-breadth ρ graphs. In particular, tree-decompositions are considered where the set of bags are the *complete* ρ-neighbourhoods of *some* vertices. We call this *strong tree-breadth*. The *strong breadth* of a tree-decomposition is ρ, if, for each bag B, there is a vertex v such that $B = N_G^{\rho}[v]$. Accordingly, a graph G has strong tree-breadth smaller than or equal to ρ (written as $\mathrm{stb}(G) \leq \rho$) if there is a tree-decomposition for G with strong breadth at most ρ.

Dually chordal graphs and their powers are exactly the graphs admitting a tree-decomposition where the set of bags is equal to the set of *complete* neighbourhoods (complete ρ-neighbourhoods) of *all* vertices. It is a known fact that the dually chordal graphs (the powers of dually chordal graphs) can be recognised in linear time (respectively, polynomial time) [4]. General tree-breadth ρ graphs cannot be recognised in polynomial time unless P = NP [13]. It remained an interesting open question if the graphs with strong tree-breadth ρ can be recognised in polynomial time.

In this paper we show that it is NP-complete to determine if a given graph has strong tree-breadth ρ, even for $\rho = 1$. Furthermore, we demonstrate that: if a graph G has strong tree-breadth ρ, then we can find a tree-decomposition for G with tree-breadth ρ in $\mathcal{O}(n^2m)$ time; with some additional restrictions, a tree-decomposition with strong breadth ρ can be found in polynomial time; some graph classes including distance-hereditary graphs have strong tree-breadth 1. Our future research plans are to investigate algorithmic implications of the existence for a graph of a tree-decomposition with small strong tree-breadth. Can some algorithmic problems that remain NP-complete on general tree-breadth ρ graphs be solved/approximated efficiently on the graphs with strong tree-breadth ρ? Recall that, for example, greedy routing with aid of a spanning tree [12], (connected) r-domination [3], Steiner tree [3], and (weighted) efficient domination [5,6] can be efficiently solved on dually chordal graphs and their powers.

2 Preliminaries

All graphs occurring in this paper are (if not stated or constructed otherwise) connected, finite, unweighted, undirected, without loops, and without multiple edges. For a graph $G = (V, E)$, we use $n = |V|$ and $m = |E|$ to denote the cardinality of the vertex set and the edge set of G. The *length* of a path from a vertex v to a vertex u is the number of edges in the path. The *distance* $d_G(u, v)$ of two vertices u and v is the length of a shortest path connecting u and v. The distance between a vertex v and a set $S \subseteq V$ is defined as $d_G(v, S) = \min_{u \in S} d_G(u, v)$.

For a vertex v of G, $N_G(v) = \{ u \in V \mid uv \in E\}$ is called the *open neighborhood* of v. Similarly, for a set $S \subseteq V$, we define $N_G(S) = \{ u \in V \mid d_G(u, S) = 1\}$. The *r-neighbourhood* of a vertex v in G is $N_G^r[v] = \{ u \mid d_G(u, v) \leq r\}$; if r is not specified, then $r = 1$. Two vertices u and v are *true twins* if $N_G[u] = N_G[v]$ and are *false twins* if they are non-adjacent and $N_G(u) = N_G(v)$.

For a vertex set S, let $G[S]$ denote the subgraph of G induced by S. With $G - S$, we denote the graph $G[V \backslash S]$. A vertex set S is a *separator* for two vertices u and v in G if each path from u to v contains a vertex $s \in S$; in this case we say S *separates* u from v. If a separator S contains only one vertex s, i. e., $S = \{s\}$, then s is an *articulation point*. A *block* is a maximal subgraph without articulation points.

A *chord* in a cycle is an edge connecting two non-consecutive vertices of the cycle. A cycle is called *induced* if it has no chords. For each $k \geq 3$, an induced cycle of length k is called as C_k. A subgraph is called *clique* if all its vertices are pairwise adjacent. A *maximal clique* is a clique that cannot be extended by including any additional vertex.

A *tree-decomposition* of a graph $G = (V, E)$ is a tree T with the vertex set $\mathcal{B}$ where each vertex of T, called bag, is a subset of V such that: (i) $V = \bigcup_{B \in \mathcal{B}} B$, (ii) for each edge $uv \in E$, there is a bag $B \in \mathcal{B}$ with $u, v \in B$, and (iii) for each vertex $v \in V$, the bags containing v induce a subtree of T. A tree-decomposition T of G has *breadth* ρ if, for each bag B of T, there is a vertex v in G with $B \subseteq N_G^\rho[v]$. The *tree-breadth* of a graph G is ρ, written as $\mathrm{tb}(G) = \rho$, if ρ is the minimal breadth of all tree-decomposition for G. Similarly, a tree-decomposition T of G has *strong breadth* ρ if, for each bag B of T, there is a vertex v in G with $B = N_G^\rho[v]$. The *strong tree-breadth* of a graph G is the minimal ρ for which G admits a tree-decomposition with strong breadth ρ. This is written as $\mathrm{stb}(G) = \rho$.

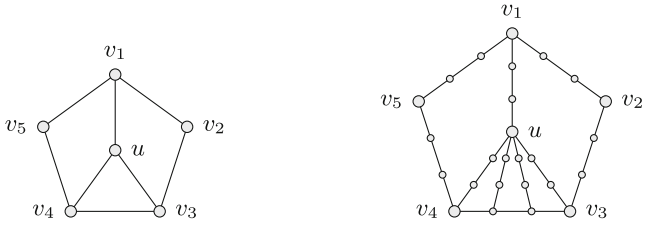
3 NP-Completeness

In this section, we will show that it is NP-complete to determine if a given graph has strong tree-breadth ρ even if $\rho = 1$. To do so, we will first show that, for some small graphs, the choice of possible centers is restricted. Then, we will use these small graphs to construct a reduction.

Lemma 1. *Let $C = \{v_1, v_2, v_3, v_4\}$ be an induced C_4 in a graph G with the edge set $\{v_1v_2, v_2v_3, v_3v_4, v_4v_1\}$. If there is no vertex $w \notin C$ with $N_G[w] \supseteq C$, then $N_G[v_1]$ and $N_G[v_2]$ cannot both be bags in the same tree-decomposition with strong breadth* 1.

Proof. Assume that there is a decomposition T with strong breadth 1 containing the bags $B_1 = N_G[v_1]$ and $B_2 = N_G[v_2]$. Because v_3 and v_4 are adjacent, there is a bag $B_3 \supseteq \{v_3, v_4\}$. Consider the subtrees T_1, T_2, T_3, and T_4 of T induced by v_1, v_2, v_3, and v_4, respectively. These subtrees pairwise intersect in the bags B_1, B_2, and B_3. Because pairwise intersecting subtrees of a tree have a common vertex, T contains a bag $N_G[w] \supseteq C$. Note that there is no $v_i \in C$ with $N_G[v_i] \supseteq C$. Thus, $w \notin C$. This contradicts with the condition that there is no vertex $w \notin C$ with $N_G[w] \supseteq C$. □

Let $C = \{v_1, \ldots, v_5\}$ be a C_5 with the edges $E_5 = \{v_1v_2, v_2v_3, \ldots, v_5v_1\}$. We call the graph $H = (C \cup \{u\}, E_5 \cup \{uv_1, uv_3, uv_4\})$, with $u \notin C$, an *extended C_5 of degree* 1 and refer to the vertices u, v_1, v_2, and v_5 as *middle*, *top*, *right*, and *left* vertex of H, respectively. Based on $H = (V_H, E_H)$, we construct an *extended C_5 of degree ρ* (with $\rho > 1$) as follows. First, replace each edge $xy \in E_H$ by a path of length ρ. Second, for each vertex w on the shortest path from v_3 to v_4, connect u with w using a path of length ρ. Figure 1 gives an illustration.



(a) Degree 1 (b) Degree 3

Fig. 1. Two *extended C_5* of degree 1 and degree 3. We refer to the vertices u, v_1, v_2, and v_5 as middle, top, right, and left vertex, respectively.

Lemma 2. *Let B be a bag of a tree-decomposition T for a graph G and let C be a connected component in $G - B$. Then, T contains a bag B_C with $B_C \supseteq N_G(C)$ and $B_C \cap C \neq \emptyset$.*

Proof. Let B_C be the bag in T for which $B_C \cap C \neq \emptyset$ and the distance between B and B_C in T is minimal. Additionally, let B' be the bag in T adjacent to B_C which is closest to B and let $S = B_C \cap B'$. Note that $S \cap C = \emptyset$ and, by properties of tree-decompositions, S separates C from all vertices in $B \backslash S$. Assume that there is a vertex $u \in N_G(C) \backslash S$. Because $u \in N_G(C)$, there is a vertex $v \in C$ which is adjacent to u. This contradicts with S being a separator for u and v. Therefore, $N_G(C) \subseteq S \subseteq B_C$. □

Lemma 3. *Let H be an extended C_5 of degree ρ in a graph G as defined above. Additionally, let H be a block of G and its top vertex v_1 be the only articulation point of G in H. Then, there is no vertex w in G with $d_G(w, v_1) < \rho$ which is the center of a bag in a tree-decomposition for G with strong breadth ρ.*

Proof. Let T be a tree-decomposition for G with strong breadth ρ. Assume that T contains a bag $B_w = N_G^\rho[w]$ with $d_G(w, v_1) < \rho$. Note that the distance from v_1 to any vertex on the shortest path from v_3 to v_4 is 2ρ. Hence, $G - B_w$ has a connected component C containing the vertices v_3 and v_4. Then, by Lemma 2, there has to be a vertex $w' \neq w$ in G and a bag $B'_w = N_G^\rho[w']$ in T such that (i) $B'_w \supseteq N_G(C)$ and (ii) $B'_w \cap C \neq \emptyset$. Thus, if we can show, for a given w, that there is no such w', then w cannot be center of a bag.

First, consider the case that w is in H. We will construct a set $X = \{x, y\} \subseteq N_G(C)$ such that there is a unique shortest path from x to y in G passing w. If $w = v_1$, let $x = v_2$ and $y = v_5$. If w is on the shortest path from v_1 to u, let x and y be on the shortest path from v_1 to v_2 and from v_4 to u, respectively. If w is on the shortest path from v_1 to v_2, let x and y be on the shortest path from v_1 to v_5 and from v_2 to v_3, respectively. In each case, there is a unique shortest path from x to y passing w. Note that, for all three cases, $d_G(v_1, y) \geq \rho$. Thus, each w' with $d_G(w', y) \leq \rho$ is in H. Therefore, w is the only vertex in G with $X \subseteq N_G^\rho[w]$, i. e., there is no vertex $w' \neq w$ satisfying condition (i). This implies that w cannot be center of a bag in T.

Next, consider the case that w is not in H. Without loss of generality, let w be a center for which $d_G(v_1, w)$ is minimal. As shown above, there is no vertex w' in H with $d_G(v_1, w') < \rho$ which is center of a bag. Hence, w' is not in H either. However, because v_1 is an articulation point, w' has to be closer to v_1 than w to satisfy condition (ii). This contradicts with $d_G(v_1, w)$ being minimal. Therefore, there is no vertex w' satisfying condition (ii) and w cannot be center of a bag in T. □

Theorem 1. *It is NP-complete to decide, for a given graph G, if* $\mathrm{stb}(G) = 1$.

Proof. Clearly, the problem is in NP: Select non-deterministically a set S of vertices such that their neighbourhoods cover each vertex and each edge. Then, check deterministically if the neighbourhoods of the vertices in S give a valid tree-decomposition. This can be done in linear time [18]. The algorithm in [18] also creates the corresponding tree.

To show that the problem is NP-hard, we will make a reduction from 1-in-3-SAT [17]. That is, you are given a boolean formula in CNF with at most three literals per clause; find a satisfying assignment such that, in each clause, only one literal becomes true.

Let $\mathcal{I}$ be an instance of 1-in-3-SAT with the literals $\mathcal{L} = \{p_1, \ldots, p_n\}$, the clauses $\mathcal{C} = \{c_1, \ldots, c_m\}$, and, for each $c \in \mathcal{C}$, $c \subseteq \mathcal{L}$. We create a graph $G = (V, E)$ as follows. Create a vertex for each literal $p \in \mathcal{L}$ and, for all literals p_i and p_j with $p_i \equiv \neg p_j$, create an induced $C_4 = \{p_i, p_j, q_i, q_j\}$ with the edges p_ip_j, q_iq_j, p_iq_i, and p_jq_j. For each clause $c \in \mathcal{C}$ with $c = \{p_i, p_j, p_k\}$, create an extended C_5 with c as top vertex, connect c with an edge to all literals it contains, and

make all literals in c pairwise adjacent, i. e., the vertex set $\{c, p_i, p_j, p_k\}$ induces a maximal clique in G. Additionally, create a vertex v and make v adjacent to all literals. Figure 2a gives an illustration for the construction so far.

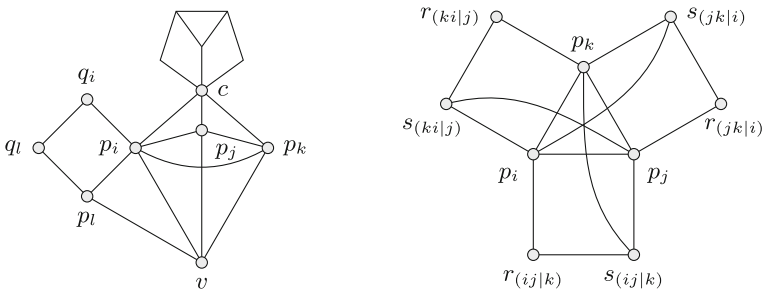


Fig. 2. Illustration to the proof of Theorem 1. The graphs shown are subgraphs of G as created by a clause $c = \{p_i, p_j, p_k\}$ and a literal p_l with $p_i \equiv \neg p_l$.

Next, for each clause $\{p_i, p_j, p_k\}$ and for each $(xy|z) \in \{(ij|k), (jk|i), (ki|j)\}$, create the vertices $r_{(xy|z)}$ and $s_{(xy|z)}$, make $r_{(xy|z)}$ adjacent to $s_{(xy|z)}$ and p_x, and make $s_{(xy|z)}$ adjacent to p_y and p_z. See Fig. 2b for an illustration. Note that $r_{(ij|k)}$ and $s_{(ij|k)}$ are specific for the clause $\{p_i, p_j, p_k\}$. Thus, if p_i and p_j are additionally in a clause with p_l, then we also create the vertices $r_{(ij|l)}$ and $s_{(ij|l)}$. For the case that a clause only contains two literals p_i and p_j, create the vertices $r_{(ij)}$ and $s_{(ij)}$, make $r_{(ij)}$ adjacent to p_i and $s_{(ij)}$, and make $s_{(ij)}$ adjacent to p_j, i. e., $\{p_i, p_j, r_{(ij)}, s_{(ij)}\}$ induces a C_4 in G.

For the reduction, first, consider the case that $\mathcal{I}$ is a *yes*-instance for 1-in-3-SAT. Let $f\colon \mathcal{P} \to \{T, F\}$ be a satisfying assignment such that each clause contains only one literal p_i with $f(p_i) = T$. Select the following vertices as centers of bags: v, the middle, left and right vertex of each extended C_5, p_i if $f(p_i) = T$, and q_j if $f(p_j) = F$. Additionally, for each clause $\{p_i, p_j, p_k\}$ with $f(p_i) = T$, select the vertices $s_{(ij|k)}$, $r_{(jk|i)}$, and $r_{(ki|j)}$. The neighbourhoods of the selected vertices give a valid tree-decomposition for G. Therefore, $\mathrm{stb}(G) = 1$.

Next, assume that $\mathrm{stb}(G) = 1$. Recall that, for a clause $c = \{p_i, p_j, p_k\}$, the vertex set $\{c, p_i, p_j, p_k\}$ induces a maximal clique in G. By Lemma 3, c cannot be center of a bag because it is top of an extended C_5. Therefore, at least one vertex in $\{p_i, p_j, p_k\}$ must be center of a bag. Without loss of generality, let p_i be a center of a bag. By construction, p_i is adjacent to all $p \in \{p_j, p_k, p_l\}$, where $p_l \equiv \neg p_i$. Additionally, p and p_i are vertices in an induced C_4, say C, and there is no vertex w in G with $N_G[w] \supseteq C$. Thus, by Lemma 1, at most one vertex in $\{p_i, p_j, p_k\}$ can be center of a bag. Therefore, the function $f\colon \mathcal{L} \to \{T, F\}$ defined as

$$f(p_i) = \begin{cases} T & \text{if } p_i \text{ is center of a bag,} \\ F & \text{else} \end{cases}$$

is a satisfying assignment for $\mathcal{I}$. □

In [13], Ducoffe et al. have shown how to construct a graph G'_ρ based on a given graph G such that $\mathrm{tb}(G'_\rho) = 1$ if and only if $\mathrm{tb}(G) \leq \rho$. We will slightly extend their construction to achieve a similar result for strong tree-breadth.

Consider a given graph $G = (V, E)$ with $\mathrm{stb}(G) = \rho$. We will construct G'_ρ as follows. Let $V = \{v_1, v_2, \ldots, v_n\}$. Add the vertices $U = \{u_1, u_2, \ldots, u_n\}$ and make them pairwise adjacent. Additionally, make each vertex u_i, with $1 \leq i \leq n$, adjacent to all vertices in $N^\rho_G[v_i]$. Last, for each $v_i \in V$, add an extended C_5 of degree 1 with v_i as top vertex.

Lemma 4. $\mathrm{stb}(G) \leq \rho$ *if and only if* $\mathrm{stb}(G'_\rho) = 1$.

Proof. First, consider a tree-decomposition T for G with strong breadth ρ. Let T'_ρ be a tree-decomposition for G'_ρ created from T by adding all vertices in U into each bag of T and by making the center, left, and right vertices of each extended C_5 centers of bags. Because the set U induces a clique in G'_ρ and $N^\rho_G[v_i] = N_{G'_\rho}[u_i] \cap V$, each bag of T'_ρ is the complete neighbourhood of some vertex.

Next, consider a tree-decomposition T'_ρ for G'_ρ with strong breadth 1. Note that each vertex v_i is top vertex of some extended C_5. Thus, v_i cannot be center of a bag. Therefore, each edge v_iv_j is in a bag $B_k = N_{G'_\rho}[u_k]$. By construction of G'_ρ, $B_k \cap V = N^\rho_G[v_k]$. Thus, we can construct a tree-decomposition T for G with strong breadth ρ by creating a bag $B_i = N^\rho_G[v_i]$ for each bag $N_{G'_\rho}[u_i]$ of T'_ρ. □

Next, consider a given graph $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_n\}$ and $\mathrm{stb}(G) = 1$. For a given $\rho > 1$, we obtain the graph G^+_ρ by doing the following for each $v_i \in V$:

- Add the vertices $u_{i,1}$, …, $u_{i,5}$, x_i, and y_i.
- Add an extended C_5 of degree ρ with the top vertex z_i.
- Connect
  - $u_{i,1}$ and x_i with a path of length $\lfloor \rho/2 \rfloor - 1$,
  - $u_{i,2}$ and y_i with a path of length $\lfloor \rho/2 \rfloor$,
  - $u_{i,3}$ and v_i with a path of length $\lceil \rho/2 \rceil - 1$,
  - $u_{i,4}$ and v_i with a path of length $\lfloor \rho/2 \rfloor$, and
  - $u_{i,4}$ and z_i with a path of length $\lceil \rho/2 \rceil - 1$.
- Add the edges $u_{i,1}u_{i,2}$, $u_{i,1}u_{i,3}$, $u_{i,2}u_{i,3}$, $u_{i,2}u_{i,4}$, and $u_{i,3}u_{i,4}$.

Note that, for small ρ, it can happen that $v_i = u_{i,4}$, $x_i = u_{i,1}$, $y_i = u_{i,2}$, or $z_i = u_{i,5}$. Figure 3 gives an illustration.

Lemma 5. $\mathrm{stb}(G) = 1$ *if and only if* $\mathrm{stb}(G^+_\rho) = \rho$.

Proof. First, assume that $\mathrm{stb}(G) = 1$. Then, there is a tree-decomposition T for G with strong breadth 1. We will construct for G^+_ρ a tree-decomposition T^+_ρ with strong breadth ρ. Make the middle, left, and right vertex of each extended C_5 center of a bag. For each $v_i \in V$, if v_i is center of a bag of T, make x_i a center of a bag of T^+_ρ. Otherwise, make y_i center of a bag of T^+_ρ. The distance in G^+_ρ

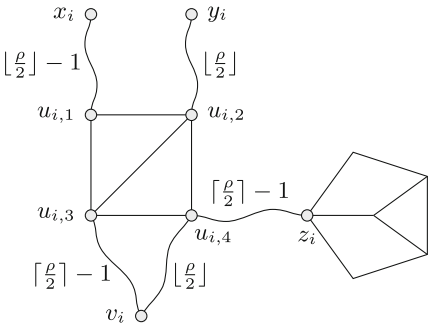


Fig. 3. Illustration for the graph G^+_ρ. The graph shown is a subgraph of G^+_ρ as constructed for each v_i in G.

from v_i to x_i is $\rho - 1$. The distances from v_i to y_i, from x_i to z_i, and from y_i to z_i are ρ. Thus, $N^\rho_{G^+_\rho}[x_i] \cap V = N_G[v_i]$, $N^\rho_{G^+_\rho}[y_i] \cap V = \{v_i\}$, and there is no conflict with Lemma 3. Therefore, the constructed T^+_ρ is a valid tree-decomposition with strong breadth ρ for G^+_ρ.

Next, assume that $\mathrm{stb}(G^+_\rho) = \rho$ and there is a tree-decomposition T^+_ρ with strong breadth ρ for G^+_ρ. By Lemma 3, no vertex in distance less than ρ to any z_i can be a center of a bag in T^+_ρ. Therefore, because the distance from v_i to z_i in G^+_ρ is $\rho - 1$, no $v_i \in V$ can be a center of a bag in T^+_ρ. The only vertices with a large enough distance to z_i to be a center of a bag are x_i and y_i. Therefore, either x_i or y_i is selected as center. To construct a tree-decomposition T with strong breadth 1 for G, select v_i as center if and only if x_i is a center of a bag in T^+_ρ. Because $N^\rho_{G^+_\rho}[x_i] \cap V = N_G[v_i]$ and $N^\rho_{G^+_\rho}[y_i] \cap V = \{v_i\}$, the constructed T is a valid tree-decomposition with strong breadth 1 for G. □

Constructing G'_ρ can be done in $\mathcal{O}(n^2)$ time and constructing G^+_ρ can be done in $\mathcal{O}(\rho \cdot n + m)$ time. Thus, combining Lemmas 4 and 5 allows us, for a given graph G, some given ρ, and some given ρ', to construct a graph H in $\mathcal{O}(\rho \cdot n^2)$ time such that $\mathrm{stb}(G) \leq \rho$ if and only if $\mathrm{stb}(H) \leq \rho'$. Additionally, by combining Theorem 1 and Lemma 3, we get:

Theorem 2. *It is NP-complete to decide, for a graph G and a given ρ, if* $\mathrm{stb}(G) = \rho$.

4 Polynomial Time Cases

In the previous section, we have shown that, in general, it is NP-complete to determine the strong tree-breadth of a graph. In this section, we will investigate cases for which a decomposition can be found in polynomial time.

4.1 General Graphs

Let G be a graph with strong tree-breadth ρ and let T be a corresponding tree-decomposition. For a given vertex u in G, we denote the set of connected components in $G - N_G^\rho[u]$ as $\mathcal{C}_G[u]$. We say that a vertex v is a *potential partner* of u for some $C \in \mathcal{C}_G[u]$ if $N_G^\rho[v] \supseteq N_G(C)$ and $N_G^\rho[v] \cap C \neq \emptyset$.

Lemma 6. *Let C be a connected component in $G - B_u$ for some $B_u \subseteq N_G^\rho[u]$. Also, let $C \in \mathcal{C}_G[u]$ and v be a potential partner of u for C. Then, for all connected components C_v in $G[C] - N_G^\rho[v]$, $C_v \in \mathcal{C}_G[v]$.*

Proof. Consider a connected component C_v in $G[C] - N_G^\rho[v]$. Clearly, $C_v \subseteq C$ and there is a connected component $C' \in \mathcal{C}[v]$ such that $C' \supseteq C_v$.

Let x be an arbitrary vertex in C'. Then, there is a path $P \subseteq C'$ from x to C_v. Because $N_G(C) \subseteq B_u$ and v is a potential partner of u for C, $N_G(C) \subseteq B_u \cap N_G^\rho[v]$. Also, $N_G(C)$ separates all vertices in C from all other vertices in G. Therefore, $x \in C$ and $C' \subseteq C$; otherwise, P would intersect $N_G^\rho[v]$. It follows that each vertex in P is in the same connected component of $G[C] - N_G^\rho[v]$ and, thus, $C_v = C'$. □

From Lemma 2, it directly follows:

Corollary 1. *If $N_G^\rho[u]$ is a bag in T, then T contains a bag $N_G^\rho[v]$ for each $C \in \mathcal{C}_G[u]$ such that v is a potential partner of u for C.*

Because of Corollary 1, there is a vertex set U such that each $u \in U$ has a potential partner $v \in U$ for each connected component $C \in \mathcal{C}_G[u]$. With such a set, we can construct a tree-decomposition for G with the following approach: Pick a vertex $u \in U$ and make it center of a bag B_u. For each connected component $C \in \mathcal{C}_G[u]$, u has a potential partner v. $N_G^\rho[v]$ splits C in more connected components and, because $v \in U$, v has a potential partner $w \in U$ for each of these components. Hence, create a bag $B_v = N_G^\rho[v] \cap (B_u \cup C)$ and continue this until the whole graph is covered. Algorithm 1 will determine such a set of vertices with their potential partners (represented as a graph H) and then construct a decomposition as described above.

Theorem 3. *Algorithm 1 constructs, for a given graph G with strong tree-breadth ρ, a tree-decomposition T with breadth ρ in $\mathcal{O}(n^2 m)$ time.*

Proof (Correctness). The Algorithm 1 works in two parts. First, it creates a graph H with potential centers (line 1 to line 6). Second, it uses H to create a tree-decomposition for G (line 9 to line 15). To show the correctness of the algorithm, we will, first, show that centers of a tree-decomposition for G are vertices in H and, then, show that a tree-decomposition created based on H is a valid tree-decomposition for G.

A vertex u is added to H (line 4) if, for *at least one* connected component $C \in \mathcal{C}_G[u]$, u has a potential partner v. Later, u is kept in H (line 5 and 6) if it has a potential partner v *for all* connected components in $C \in \mathcal{C}_G[u]$. By Corollary 1,

Algorithm 1. Constructs, for a given graph $G = (V, E)$ with strong tree-breadth ρ, a tree-decomposition T with breadth ρ.

1 Create an empty directed graph $H = (V_H, E_H)$. Let ϕ be a function that maps each edge $(u, v) \in E_H$ to a connected component $C \in \mathcal{C}_G[u]$.
2 **foreach** *$u, v \in V$ and all $C \in \mathcal{C}_G[u]$* **do**
3 **if** *v is a potential parter of u for C* **then**
4 Add the directed edge (u, v) to H and set $\phi(u, v) := C$. (Add u and v to H if necessary.)
5 **while** *there is a vertex $u \in V_H$ and some $C \in \mathcal{C}_G[u]$ such that there is no $(u, v) \in E_H$ with $\phi(u, v) = C$* **do**
6 Remove u from H.
7 **if** *H is empty* **then**
8 **Stop**. $\mathrm{stb}(G) > \rho$.
9 Create an empty tree-decomposition T.
10 Let $G - T$ be the subgraph of G that is not covered by T and let ψ be a function that maps each connected component in $G - T$ to a bag $B_u \subseteq N_G^\rho[u]$.
11 Pick an arbitrary vertex $u \in V_H$, add $B_u = N_G^\rho[u]$ as bag to T, and set $\psi(C) := B_u$ for each connect component C in $G - T$.
12 **while** *$G - T$ is non-empty* **do**
13 Pick a connected component C in $G - T$, determine the bag $B_v := \psi(C)$ and find an edge $(v, w) \in E_H$ with $\phi(v, w) = C$.
14 Add $B_w = N_G^\rho[w] \cap (B_v \cup C)$ to T, and make B_v and B_w adjacent in T.
15 For each new connected component C' in $G - T$ with $C' \subseteq C$, set $\psi(C_w) := B_w$.
16 Output T.

each center of a bag in a tree-decomposition T with strong breadth ρ satisfies these conditions. Therefore, after line 6, H contains all centers of bags in T, i. e., if G has strong tree-breadth ρ, H is non-empty.

Next, we show that T created in the second part of the algorithm (line 9 to line 15) is a valid tree-decomposition for G with breadth ρ. To do so, we will show the following invariant for the loop starting in line 12: (i) T is a valid tree-decomposition with breadth ρ for the subgraph covered by T and (ii) for each connected component C in $G - T$, the bag $B_v = \psi(C)$ is in T, $N_G(C) \subseteq B_v$, and $C \in \mathcal{C}_G[v]$. After line 11, the invariant clearly holds. Assume by induction that the invariant holds each time line 12 is checked. If T covers the whole graph, the check fails and the algorithm outputs T. If T does not cover G completely, there is a connected component C in $G - T$. By condition (ii), the bag $B_v = \psi(C)$ is in T, $N_G(C) \subseteq B_v$, and $C \in \mathcal{C}_G[v]$. Because of the way H is constructed and $C \in \mathcal{C}_G[v]$, there is an edge $(v, w) \in E_H$ with $\phi(v, w) = C$, i. e., w is a potential partner of v for C. Thus, line 13 is successful and the algorithm adds a new bag $B_w = N_G^\rho[w] \cap (B_v \cup C)$ (line 14). Because w is a potential partner of v for C, i. e., $N_G(C) \subseteq N_G^\rho[w]$, and $N_G(C) \subseteq B_v$, $B_w \supseteq N_G(C)$. Therefore, after adding B_w to T, T still satisfies condition (i). Additionally, B_w splits C in a

set $\mathcal{C}'$ of connected components such that, for each $C' \in \mathcal{C}'$, $N_G(C') \subseteq B_w$ and, by Lemma 6, $C' \in \mathcal{C}_G[w]$. Thus, condition (ii) is also satisfied. □

Proof (Complexity). First, determine the pairwise distance of all vertices. This can be done in $\mathcal{O}(nm)$ time and allows to check the distance between vertices in constant time.

For a vertex u, let $\mathcal{N}[u] = \{ N_G(C) \mid C \in \mathcal{C}_G[u]\}$. Note that, for some $C \in \mathcal{C}_G[u]$ and each vertex $x \in N_G(C)$, there is an edge xy with $y \in C$. Therefore, $|\mathcal{N}[u]| := \sum_{C \in \mathcal{C}_G[u]} |N_G(C)| \leq m$. To determine, for some vertex u, all its potential partners v, first, compute $\mathcal{N}[u]$. This can be done in $\mathcal{O}(m)$ time. Then, check, for each vertex v and each $N_G(C) \in \mathcal{N}[u]$, if $N_G(C) \subseteq N_G^\rho[v]$ and add the edge (u, v) to H if successful. For a single vertex v this requires $\mathcal{O}(m)$ time because $|\mathcal{N}[u]| \leq m$ and distances can be determined in constant time. Therefore, the total runtime for creating H (line 1 to line 4) is $\mathcal{O}(n(m + nm)) = \mathcal{O}(n^2 m)$.

Assume that, for each $\phi(u, v) = C$, C is represented buy two values: (i) a characteristic vertex $x \in C$ (for example the vertex with the lowest index) and (ii) the index of C in $\mathcal{C}_G[u]$. While creating H, count and store, for each vertex u and each connected component $C \in \mathcal{C}_G[u]$, the number of edges $(u, v) \in E_H$ with $\phi(u, v) = C$. Note that there is a different counter for each $C \in \mathcal{C}_G[u]$. With this information, we can implement line 5 and 6 as follows. First check, for every vertex v in H, if one of its counters is 0. In this case, remove v from H and update the counters for all vertices u with $(u, v) \in E_H$ using value (ii) of $\phi(u, v)$. If this sets a counter for u to 0, add u to a queue Q of vertices to process. Continue this until each vertex is checked. Then, for each vertex u in Q, remove u form H and add its neighbours into Q if necessary until Q is empty. This way, a vertex is processed at most twice. A single iteration runs in at most $\mathcal{O}(n)$ time. Therefore, line 5 and 6 can be implemented in $\mathcal{O}(n^2)$ time.

Assume that ψ uses the characteristic vertex x to represent a connected component, i. e., value (i) of ϕ. Then, finding an edge $(v, w) \in E_H$ (line 13) can be done in $\mathcal{O}(m)$ time. Creating B_w (line 14), splitting C into new connected components C', finding their characteristic vertex, and setting $\psi(C')$ (line 15) takes $\mathcal{O}(m)$ time, too. In each iteration, at least one more vertex of G is covered by T. Hence, there are at most n iterations and, thus, the loop starting in line 12 runs in $\mathcal{O}(mn)$ time.

Therefore, Algorithm 1 runs in total $\mathcal{O}(n^2 m)$ time. □

Algorithm 1 creates for each graph G with $\mathrm{stb}(G) \leq \rho$ a tree-decomposition T with breadth ρ. Next, we will invest a case where we can construct a tree-decomposition for G with strong breadth ρ.

We say that two vertices u and v are *perfect partners* if (i) u and v are potential partner of each other for some $C_u \in \mathcal{C}_G[u]$ and some $C_v \in \mathcal{C}_G[v]$, (ii) C_u is the only connected component in $\mathcal{C}_G[u]$ which is intersected by $N_G^\rho[v]$, and (iii) C_v is the only connected component in $\mathcal{C}_G[v]$ which is intersected by $N_G^\rho[u]$. Accordingly, we say that a tree-decomposition T has *perfect strong breadth* ρ if it has strong breadth ρ and, for each center u of some bag and each connected component $C \in \mathcal{C}_G[u]$, there is a center v such that v is a perfect partner of u for C.

Theorem 4. *A tree-decomposition with perfect strong breadth ρ can be constructed in polynomial time.*

Proof. To construct such a tree-decomposition, we can modify Algorithm 1. Instead of checking if u has a potential partner v (line 3), check if u and v are perfect partners.

Assume by induction that, for each bag B_v in T, $B_v = N_G^\rho[v]$. By definition of perfect partners v and w, $N_G^\rho[w]$ intersects only one $C \in \mathcal{C}_G[v]$, i. e., $N_G^\rho[w] \subseteq N_G^\rho[v] \cup C$. Thus, when creating the bag B_w (line 14), $B_w = N_G^\rho[w] \cap (B_v \cup C) = N_G^\rho[w] \cap (N_G^\rho[v] \cup C) = N_G^\rho[w]$. Therefore, the created tree-decomposition T has perfect strong tree-breadth ρ. □

We conjecture that there are weaker cases than perfect strong breadth which allow to construct a tree-decomposition with strong-breadth ρ. For example, if the centers of two adjacent bags are perfect partners, but a center u does not need to have a perfect partner for each $C \in \mathcal{C}_G[u]$. However, when using a similar approach as in Algorithm 1, this would require a more complex way of constructing H.

4.2 Special Graph Classes

A graph G is *distance-hereditary* if, in any connected induced subgraph, the distances are the same as in G.

Theorem 5. *Distance-hereditary graphs have strong tree-breadth 1. An according decomposition can be computed in linear time.*

Proof. Let $\sigma = \langle v_1, v_2, \ldots, v_n \rangle$ be an ordering for the vertices of a graph G, $V_i = \{v_1, v_2, \ldots, v_i\}$, and G_i denote the graph $G[V_i]$. An ordering σ is called a *pruning sequence* for G if, for $1 < i \leq n$, each v_i satisfies one of the following conditions in G_i:

(i) v_i is a pendant vertex,
(ii) v_i is a true twin of some vertex v_j, or
(iii) v_i is a false twin of some vertex v_j.

A graph G is distance-hereditary if and only if there is a pruning sequence for G [2].

Assume that we are given such a pruning sequence. Additionally, assume by induction over i that G_i has a tree-decomposition T_i with strong breadth 1. Then, there are three cases:

(i) *v_{i+1} is a pendant vertex in G_{i+1}.* If the neighbour u of v_{i+1} is a center of a bag B_u, add v_{i+1} to B_u. Thus, T_{i+1} is a valid decomposition for G_{i+1}. Otherwise, if u is not a center, make v_{i+1} center of a bag. Because u is an articulation point, $T_{i+1} = T_i + N_G[v]$ is a valid decomposition for G_{i+1}.
(ii) *v_{i+1} is a true twin of a vertex u in G_{i+1}.* Simply add v_{i+1} into any bag containing u. The resulting decomposition is a valid decomposition for G_{i+1}.

(iii) v_{i+1} *is a false twin of a vertex* u in G_{i+1}. If u is not center of a bag, add v_{i+1} into any bag u is in. Otherwise, make a new bag $B_{i+1} = N_G[v_{i+1}]$ and make it adjacent to the bag $N_G[u]$. Because no vertex in $N_G(u)$ is center of a bag, the resulting decomposition is a valid decomposition for G_{i+1}.

Therefore, distance-hereditary graphs have strong tree-breadth 1.

Next, we will show how to compute an according tree-decomposition in linear time. The argument above already gives an algorithmic approach. First, we compute a pruning sequence for G. This can be done in linear time with an algorithm by Damiand et al. [7]. Then, we determine which vertex becomes a center of a bag. Note that we can simplify the three cases above with the following rule: If v_i has no neighbour in G_i which is center of a bag, make v_i center of a bag. Otherwise, proceed with v_{i+1}. This can be easily implemented in linear time with a binary flag for each vertex. □

Algorithm 2 formalizes the method described in the proof of Theorem 5.

Algorithm 2. Computes, for a given distance-hereditary graph G, a tree-decomposition T with strong breadth 1.

```
1 Compute a pruning sequence ⟨v_1, v_2, ..., v_n⟩ (see [7]).
2 Create a set C := ∅.
3 for i := 1 to n do
4 |   if N_G[v_i] ∩ V_i ∩ C = ∅ then
5 |   |   Add v_i to C.
6 Create a tree-decomposition T with the vertices in C as centers of its bags.
```

A bipartite graph is *chordal bipartite* if each cycle of length at least 6 has a chord. In [11], it was shown that any chordal bipartite graph $G = (X, Y, E)$ admits a tree-decomposition with the set of bags $\mathcal{B} = \{B_1, B_2, \ldots, B_{|X|}\}$, where $B_i = N_G[x_i]$, $x_i \in X$. As far as we can tell, there is no linear time algorithm known to recognise chordal bipartite graphs. However, we can still compute a tree-decomposition in linear time with three steps. First, compute a 2-colouring. Second, select a colour and make the neighbourhood of all vertices with this colour bags. Third, use the algorithm in [18] to check if the selected bags give a valid tree-decomposition.

Theorem 6 [11]. *Each chordal bipartite graph has strong tree-breadth* 1. *An according tree-decomposition can be found in linear time.*

Consider two parallel lines (upper and lower) in the plane. Assume that each line contains n points, labelled 1 to n. Each two points with the same label define a segment with that label. The intersection graph of such a set of segments between two parallel lines is called a *permutation graph*. In [10], an algorithm was presented that finds, for a given permutation graph, a path-decomposition with strong breadth 1 in linear time.

Theorem 7 [10]. *Permutation graphs have strong tree-breadth* 1. *An according tree-decomposition can be found in linear time.*

5 Conclusion

We have shown that, in general, it is NP-complete to determine if a given graph G admits a tree-decomposition with strong breadth ρ for all $\rho \geq 1$. Consider the case that a vertex v is center of a bag. Part of the hardness of finding a decomposition, even for $\rho = 1$, lays in determining which connected component $C \in \mathcal{C}_G[v]$ will be covered by which neighbouring bag $N_G[u]$. If, for two vertices u and w, $N_G[u]$ and $N_G[w]$ intersect C and are bags in the same decompositions T, both cannot be separated in T by $N_G[v]$. Additionally, if u is adjacent to v, it might happen that $N_G[u]$ intersects multiple connected components. This leads to a potentially exponential number of combinations.

A *path-decomposition* of graph is a tree-decomposition with the restriction that the bags form a path instead of a tree with multiple branches. Accordingly, a graph has *(strong) path-breadth* ρ if it admits a path-decomposition with (strong) breadth ρ. In [10], it was shown that, for graphs with bounded path-breadth, a constant factor approximation for the bandwidth problem and the line-distortion problem can be found in polynomial time.

Now, consider the case that we want to compute if a given graph admits a path-decomposition P with strong breadth 1. In this case, there can be at most two bags adjacent to a bag $N_G[v]$ in P. Hence, for each v, there is at most a quadratic number of combinations. This leads to the following conjecture.

Conjecture. *The strong path-breadth of a graph can be computed in polynomial time.*

Another question is if a bounded strong tree-breadth leads to a lower bound for the tree-breadth of a graph. That is, is there a constant c such that, for any graph G, $\mathrm{stb}(G) \leq c \cdot \mathrm{tb}(G)$. Using Algorithm 1, a small constant might lead to a new approach for approximating the tree-breadth of a graph.

References

1. Abu-Ata, M., Dragan, F.F.: Metric tree-like structures in real-life networks: an empirical study. Networks **67**(1), 49–68 (2016)
2. Bandelt, H.-J., Mulder, H.M.: Distance-hereditary graphs. J. Comb. Theory Ser. B **41**, 182–208 (1986)
3. Brandstädt, A., Chepoi, V.D., Dragan, F.F.: The algorithmic use of hypertree structure and maximum neighborhood orderings. Discret. Appl. Math. **82**, 43–77 (1998)
4. Brandstädt, A., Dragan, F.F., Chepoi, V.D., Voloshin, V.: Dually chordal graphs. SIAM J. Discret. Math. **11**(3), 437–455 (1998)
5. Brandstädt, A., Fičur, P., Leitert, A., Milanič, M.: Polynomial-time algorithms for weighted efficient domination problems in AT-free graphs and dually chordal graphs. Inf. Process. Lett. **115**(2), 256–262 (2015)

6. Brandstädt, A., Leitert, A., Rautenbach, D.: Efficient dominating and edge dominating sets for graphs and hypergraphs. In: Chao, K.-M., Hsu, T., Lee, D.-T. (eds.) ISAAC 2012. LNCS, vol. 7676, pp. 267–277. Springer, Heidelberg (2012)
7. Damiand, G., Habib, M., Paul, C.: A simple paradigm for graph recognition: application to cographs and distance hereditary graphs. Theoret. Comput. Sci. **263**(1–2), 99–111 (2001)
8. Dourisboure, Y., Gavoille, C.: Tree-decompositions with bags of small diameter. Discret. Math. **307**(16), 2008–2029 (2007)
9. Dragan, F.F., Köhler, E.: An approximation algorithm for the tree t-spanner problem on unweighted graphs via generalized chordal graphs. Algorithmica **69**, 884–905 (2014)
10. Dragan, F.F., Köhler, E., Leitert, A.: Line-distortion, bandwidth and path-length of a graph. Algorithmica (in print)
11. Dragan, F.F., Lomonosov, I.: On compact and efficient routing in certain graph classes. Discret. Appl. Math. **155**, 1458–1470 (2007)
12. Dragan, F.F., Matamala, M.: Navigating in a graph by aid of its spanning tree. SIAM J. Discret. Math. **25**(1), 306–332 (2011)
13. Ducoffe, G., Legay, S., Nisse, N.: On computing tree and path decompositions with metric constraints on the bags. CoRR abs/1601.01958 (2016)
14. Halin, R.: S-functions for graphs. J. Geom. **8**(1–2), 171–186 (1976)
15. Robertson, N., Seymour, P.D.: Graph minors. I. Excluding a forest. J. Comb. Theory Ser. B **35**(1), 39–61 (1983)
16. Robertson, N., Seymour, P.D.: Graph minors. III. Planar tree-width. J. Comb. Theory Ser. B **36**(1), 49–64 (1984)
17. Schaefer, T.J.: The complexity of satisfiability problems. In: Proceedings of the Tenth Annual ACM Symposium on Theory of Computing (STOC 1978), pp. 216–226 (1978)
18. Tarjan, R.E., Yannakakis, M.: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. SIAM J. Comput. **13**(3), 566–579 (1984)