

Graph Search Trees and Their Leaves

Robert Scheffler

Institute of Mathematics, Brandenburg University of Technology,
Cottbus, Germany

`robert.scheffler@b-tu.de`

Graph searches and their respective search trees are widely used in algorithmic graph theory. The problem whether a given spanning tree can be a graph search tree has been considered for different searches, graph classes and search tree paradigms. Similarly, the question whether a particular vertex can be visited last by some search has been studied extensively in recent years. We combine these two problems by considering the question whether a vertex can be a leaf of a graph search tree. We show that for particular search trees, including DFS trees, this problem is easy if we allow the leaf to be the first vertex of the search ordering. We contrast this result by showing that the problem becomes hard for many searches, including DFS and BFS, if we forbid the leaf to be the first vertex. Additionally, we present several structural and algorithmic results for search tree leaves of chordal graphs.

1 Introduction

Graph searches are an extensively used concept in algorithmic graph theory. The searches BFS and DFS belong to the most basic algorithms and are used in a wide range of applications as subroutines. The same holds for more sophisticated searches as LBFS, LDFS, and MCS (see, e.g., [4, 9, 15]).

An important structure closely related to a graph search is the corresponding search tree. Such a tree contains all the vertices of the graph and for every vertex different from the start vertex exactly one edge to a vertex preceding it in the search ordering. Those trees can be of particular interest as for instance the tree obtained by a BFS contains the shortest paths from the root to all other vertices in the graph and DFS trees are used for fast planarity testing [20]. Furthermore, trees generated by LBFS were used to design a linear-time implementation of the search LDFS for chordal graphs [3].

The problem of deciding whether a given spanning tree of a graph can be obtained by a particular search was introduced by Hagerup [18] in 1985, who presented a linear-time algorithm that recognizes DFS trees. In the same year, Hagerup and Nowak [19] gave

a similar result for the BFS tree recognition. In 2021, Beisegel et al. [2] presented a more general framework for the search tree recognition problem. They introduced the term \mathcal{F} -tree for search trees where a vertex is connected to its first visited neighbor, i.e., BFS-like trees, and \mathcal{L} -trees for search trees where a vertex is connected to its most recently visited neighbor, i.e., DFS-like trees. They showed, among other things, that \mathcal{F} -tree recognition is NP-hard for LBFS, LDFS, and MCS on weakly chordal graphs, while the problem can be solved in polynomial time for all three searches on chordal graphs. These results are complemented in [30], where it is shown that the recognition of \mathcal{F} -trees of DFS and \mathcal{L} -trees of BFS is NP-hard, a strong contrast to the polynomial results for \mathcal{F} -trees of BFS and \mathcal{L} -trees of DFS.

Another feature of a graph search that was used several times within algorithms are its end-vertices, i.e., the vertices that can be visited last by the search. Some of these end-vertices have nice properties. One example are the end-vertices of LBFS on chordal graphs. These vertices are simplicial, a fact that was used by Rose et al. [28] to design a linear-time recognition algorithm for chordal graphs. Furthermore, the end-vertices of LBFS are strongly related to dominating pairs of AT-free graphs [14] and transitive orientations of comparability graphs [17]. Thus, it is well motivated to consider the end-vertex problem, i.e., the question whether a given vertex of a graph is an end-vertex of a particular search. Introduced in 2010 by Corneil et al. [12], the problem has gained much attention by several researchers, leading to a wide range of hardness results and algorithms for different searches on different graph classes (see, e.g., [1, 7, 16, 23, 26, 34]).

If we compare the known complexity results for the end-vertex problem and the recognition problem of \mathcal{F} -trees, we notice strong similarities between these two problems. Motivated by that fact, a generalization of both problems, called *Partial Search Order Problem*, was introduced in [29]. This problem asks whether a given partial order on a graph's vertex set can be linearly extended by a search ordering. Another way to combine the end-vertex problem with the search tree recognition problems is motivated by the following observation: If a vertex is the end-vertex of some search ordering, then it is a leaf in the respective search tree, no matter whether we consider the \mathcal{F} -tree or the \mathcal{L} -tree. Therefore, we ask whether a given vertex can be a leaf of a search tree constructed by a particular search. Note that this problem was first suggested in 2020 by Michel Habib. Here, we study its complexity for \mathcal{F} -trees and \mathcal{L} -trees of several searches, including BFS, DFS, LBFS, LDFS, and MCS.

Our Contribution. We consider two different types of leaves of search trees. A leaf is a *root leaf* of a search tree if it is the start vertex of the respective search ordering. All other leaves of a search tree are called *branch leaves*. We show that it is easy for all the searches considered here to identify the possible root leaves both for \mathcal{F} -trees and for \mathcal{L} -trees. For some searches, including DFS, these results imply directly that the general problem of recognizing leaves of \mathcal{L} -trees is easy. This is contrasted by the result that, at least for DFS, the recognition of branch leaves of \mathcal{L} -trees is NP-hard. We show that the same holds for \mathcal{F} -tree branch leaves of several searches, including DFS and BFS. In contrast, the leaves of \mathcal{L} -trees of BFS can be recognized in polynomial time for bipartite

graphs. This is quite surprising since the \mathcal{L} -tree recognition problem of BFS is NP-hard on bipartite graphs [30] while \mathcal{F} -trees of BFS can be recognized efficiently on general graphs [19]. In the final section we consider chordal graphs and show that on this graph class the branch leaves of almost all considered searches can be recognized in linear time.

2 Preliminaries

2.1 General Notation

The graphs considered in this paper are finite, undirected, simple and connected. Given a graph G , we denote by $V(G)$ the *set of vertices* and by $E(G)$ the *set of edges*. The terms $n(G)$ and $m(G)$ describe the number of vertices and edges of G , respectively, i.e., $n(G) = |V(G)|$ and $m(G) = |E(G)|$. For a vertex $v \in V(G)$, we denote by $N_G(v)$ the (*open*) *neighborhood* of v in G , i.e., the set $N_G(v) = \{u \in V \mid uv \in E\}$ where uv denotes an edge between u and v . The *closed neighborhood* of a vertex v is the union of the open neighborhood of v with the set $\{v\}$ and is denoted by $N_G[v]$. Given a set $S \subseteq V(G)$, the term $G[S]$ describes the subgraph of G that is induced by S .

The *distance* $dist_G(v, w)$ of two vertices v and w in G is the length (i.e., the number of edges) of the shortest v - w -path in G . The *eccentricity* $ecc_G(v)$ of a vertex v in G is the largest distance of v to any other vertex in G . The *diameter* $diam(G)$ of G is the largest eccentricity of a vertex in G and the *radius* $rad(G)$ of G is the smallest eccentricity of a vertex in G . A vertex v with $ecc_G(v) = rad(G)$ is called *central vertex* of G . The set $N_G^\ell(v)$ contains all vertices whose distance to the vertex v in G is equal to ℓ .

A *vertex ordering* of G is a bijection $\sigma : \{1, 2, \dots, |V(G)|\} \rightarrow V(G)$. We denote by $\sigma^{-1}(v)$ the position of vertex $v \in V(G)$. Given two vertices u and v in G we say that u is *to the left* (resp. *to the right*) of v if $\sigma^{-1}(u) < \sigma^{-1}(v)$ (resp. $\sigma^{-1}(u) > \sigma^{-1}(v)$) and we denote this by $u \prec_\sigma v$ (resp. $u \succ_\sigma v$). Given two orderings σ of X and ρ of Y with $X \cap Y = \emptyset$, the ordering $\tau = \sigma ++ \rho$ is the *concatenation* of σ and ρ , i.e., $\tau(i) = \sigma(i)$ if $1 \leq i \leq |X|$ and $\tau(i) = \rho(i - |X|)$ if $|X| < i \leq |X \cup Y|$. If $v \notin X$, then $v ++ \sigma$ (or $\sigma ++ v$) denotes the concatenation of σ with the linear ordering of the set $\{v\}$.

A *clique* in a graph G is a set of pairwise adjacent vertices and an *independent set* in G is a set of pairwise nonadjacent vertices. A clique C is *dominating* if any vertex of G is either in C or has a neighbor in C . A vertex v is *simplicial* if its neighborhood induces a clique. A vertex v of a connected graph G is a *cut vertex* if $G - v$ is not connected. Two vertices u and w form a *two-pair* if any induced path between u and w has length two.

A graph is *bipartite* if its vertex set can be partitioned into two independent sets X and Y . A graph is *weakly chordal* if G contains neither an induced cycle of the length ≥ 5 nor the complement of such an induced cycle. A graph is *chordal* if it does not contain an induced cycle of length ≥ 4 . A vertex ordering σ of a graph G is a *perfect elimination ordering* if any vertex v is simplicial in the graph $G[S(v)]$ with $S(v) := \{w \mid w \prec_\sigma v\}$. A graph G has a PEO if and only if G is chordal [27]. A *split graph* G is a graph whose

Algorithm 1: Label Search($\prec_{\mathcal{A}}$)

Input: A graph G
Output: A search ordering σ of G

```
1 begin
2   foreach  $v \in V(G)$  do  $label(v) \leftarrow \emptyset$ ;
3   for  $i \leftarrow 1$  to  $n(G)$  do
4      $Eligible \leftarrow \{x \in V(G) \mid x \text{ unnumbered and } \nexists \text{ unnumbered } y \in V(G)$ 
5        $\text{ such that } label(x) \prec_{\mathcal{A}} label(y)\}$ ;
6     let  $v$  be an arbitrary vertex in  $Eligible$ ;
7      $\sigma(i) \leftarrow v$ ; /* assigns to  $v$  the number  $i$  */
8     foreach unnumbered vertex  $w \in N(v)$  do  $label(w) \leftarrow label(w) \cup \{i\}$ ;
```

vertex set can be partitioned into sets C and I , such that C is a clique in G and I is an independent set in G . It is easy to see that any split graph is chordal.

A *tree* is an acyclic connected graph. A *spanning tree* of a graph G is an acyclic connected subgraph of G which contains all vertices of G . A tree together with a distinguished *root vertex* r is said to be *rooted*. In such a rooted tree T , a vertex v is an *ancestor* of vertex w if v is an element of the unique path from w to the root r . A vertex w is called the *descendant* of v if v is the ancestor of w . Vertex v is the *parent* of vertex w if v is an ancestor of w and is adjacent to w in T . Vertex w is called the *child* of v if v is the parent of w .

2.2 Searches, Search Trees and Their Leaves

In the most general sense, a *graph search* \mathcal{A} is a function that maps every graph G to a set $\mathcal{A}(G)$ of vertex orderings of G . The elements of the set $\mathcal{A}(G)$ are the \mathcal{A} -orderings of G . The graph searches considered in this paper can be formalized adapting a framework introduced by Corneil et al. [11] (a similar framework is given in [24]). This framework uses subsets of \mathbb{N}^+ as vertex labels. Whenever a vertex is numbered, its index in the search ordering is added to the labels of its unnumbered neighbors. The search \mathcal{A} is defined via a strict partial order $\prec_{\mathcal{A}}$ on the elements of $\mathcal{P}(\mathbb{N}^+)$ (see Algorithm 1). The respective \mathcal{A} -orderings are exactly those vertex orderings that can be found by this framework using the partial label order $\prec_{\mathcal{A}}$.

In the following, we define the searches considered in this paper by presenting suitable partial orders $\prec_{\mathcal{A}}$ (see [11]). The *Generic Search* (GS) is equal to the Label Search(\prec_{GS}) where $A \prec_{GS} B$ if and only if $A = \emptyset$ and $B \neq \emptyset$. Thus, any vertex with a numbered neighbor can be numbered next.

The partial label order \prec_{BFS} for *Breadth First Search* (BFS) is defined as follows: $A \prec_{BFS} B$ if and only if $A = \emptyset$ and $B \neq \emptyset$ or $\min(A) > \min(B)$. For the *Lexicographic Breadth First Search* (LBFS) [28] we consider the partial order \prec_{LBFS} with $A \prec_{LBFS} B$ if and only if $A \subsetneq B$ or $\min(A \setminus B) > \min(B \setminus A)$. Both BFS and LBFS are *layered*, i.e., the sets $N_G^\ell(r)$ are consecutive within orderings starting in r . We sometimes use the term *layer* if we refer to a set $N_G^\ell(r)$.

The partial label order \prec_{DFS} for *Depth First Search* (DFS) is defined as follows: $A \prec_{DFS} B$ if and only if $A = \emptyset$ and $B \neq \emptyset$ or $\max(A) < \max(B)$. For the *Lexicographic Depth First Search* [13] we use the strict partial order \prec_{LDFS} where $A \prec_{LDFS} B$ if and only if $A \subsetneq B$ or $\max(A \setminus B) < \max(B \setminus A)$.

The *Maximum Cardinality Search* (MCS) [33] uses the partial order \prec_{MCS} with $A \prec_{MCS} B$ if and only if $|A| < |B|$. The *Maximal Neighborhood Search* (MNS) [13] is defined using \prec_{MNS} with $A \prec_{MNS} B$ if and only if $A \subsetneq B$. It follows directly from these partial label orders, that any LBFS, LDFS, and MCS ordering is also an MNS ordering. Furthermore, the orderings of all presented searches are GS orderings.

In general, there can occur ties during an application of all the searches considered in this paper. This problem can be solved by considering so-called *+searches*. Let \mathcal{A} be a graph search. Given an arbitrary vertex ordering ρ of a graph G , an \mathcal{A} -ordering $\sigma = (v_1, \dots, v_n)$ of G is the $\mathcal{A}^+(\rho)$ -ordering of G if σ fulfills the following condition: for any $i \in \{0, \dots, n-1\}$, the vertex v_{i+1} is the *leftmost* vertex in ρ such that $(v_1, \dots, v_i, v_{i+1})$ is a prefix of an \mathcal{A} -ordering of G .¹

Searches as BFS and DFS are often used to compute corresponding graph search trees. Beisegel et al. [2] formalized the different concepts of search trees as follows.

Definition 1 (Beisegel et al. [2]). *Let σ be a GS ordering of a connected graph G . The \mathcal{F} -tree of σ is the spanning tree of G containing the edge from each vertex v with $\sigma^{-1}(v) > 1$ to its leftmost neighbor in σ .*

The \mathcal{L} -tree of σ is the spanning tree containing the edge from each vertex v with $\sigma^{-1}(v) > 1$ to its rightmost neighbor w in σ with $w \prec_{\sigma} v$.

We will need the following two lemmas about DFS \mathcal{L} -trees.

Lemma 2 (Tarjan [32]). *Let G be a graph and let T be a spanning tree of G . Then T is a DFS \mathcal{L} -tree of G if and only if for each edge $uv \in E(G)$ vertex u is either an ancestor or a descendant of v in T .*

Lemma 3 (Beisegel et al. [3]). *Let T be an \mathcal{L} -tree of some DFS ordering of G rooted in s and let σ be a DFS ordering of T starting with s . Then σ is a DFS ordering of G with \mathcal{L} -tree T .*

In this paper, we consider the leaves of search trees. For both \mathcal{F} -trees and \mathcal{L} -trees, we distinguish two different types of leaves.

Definition 4. *Let σ be a GS ordering of a connected graph G . A vertex $v \in V(G)$ is an \mathcal{F} -leaf (\mathcal{L} -leaf) of σ if v is a leaf in the \mathcal{F} -tree (\mathcal{L} -tree) of σ . If v is the first vertex of σ , then it is the \mathcal{F} -root leaf (\mathcal{L} -root leaf) of σ , otherwise it is an \mathcal{F} -branch leaf (\mathcal{L} -branch leaf) of σ .*

As the graph with exactly one vertex has no leaf in its spanning tree, we will consider only graphs with at least two vertices.

¹Note that other authors define the vertex v_{i+1} to be the *rightmost* vertex in ρ .

3 Root Leaves

We start this section with the simple observation that \mathcal{F} -root leaves of GS orderings of a graph G are quite boring as they are exactly the leaves of G .

Observation 5. *Let G be a connected graph with $n(G) \geq 2$. The following conditions are equivalent for a vertex $v \in V(G)$.*

- (i) *Vertex v is the \mathcal{F} -root leaf of some GS ordering of G .*
- (ii) *Vertex v is the \mathcal{F} -root leaf of every GS ordering of G starting in v .*
- (iii) *Vertex v is a leaf of G .*

Next we consider the \mathcal{L} -root leaves of GS, DFS, and MCS. They are exactly those vertices of the graph that are not cut vertices. The same even holds for \mathcal{F} -branch leaves and \mathcal{L} -branch leaves of GS.

Theorem 6. *Let G be a connected graph with $n(G) \geq 2$. The following conditions are equivalent for a vertex $v \in V(G)$.*

- (i) *Vertex v is the \mathcal{L} -root leaf of some DFS ordering of G starting in v .*
- (ii) *Vertex v is the \mathcal{L} -root leaf of every DFS ordering of G starting in v .*
- (iii) *Vertex v is the \mathcal{L} -root leaf of some MCS ordering of G .*
- (iv) *Vertex v is the \mathcal{L} -root leaf of some GS ordering of G .*
- (v) *Vertex v is an \mathcal{L} -branch leaf of some GS ordering of G .*
- (vi) *Vertex v is an \mathcal{F} -branch leaf of some GS ordering of G .*
- (vii) *Vertex v is the end-vertex of some GS ordering of G .*
- (viii) *Vertex v is not a cut vertex of G .*

Proof. First we show the equivalence of all statements concerning GS and statement (viii). Charbit et al. [7] showed that a vertex v is an end-vertex of a GS ordering if and only if it is not a cut vertex of G . Thus, (vii) \Leftrightarrow (viii). Clearly, the end-vertex of a GS ordering is also a branch leaf of its \mathcal{F} -tree and its \mathcal{L} -tree. Hence, (vii) \Rightarrow (v), (vi).

Now assume that vertex v is not a cut vertex of G . Then let σ be a GS ordering of $G - v$ that starts in a neighbor w of v . Obviously, $\sigma' = v \dashv\vdash \sigma$ is a GS ordering of G . As $G - v$ is connected, every vertex of $G - v$ except from w has a neighbor to its left in σ . Thus, the only neighbor of v in the \mathcal{L} -tree of σ' is w and v is a GS \mathcal{L} -root leaf of G . Hence, (viii) \Rightarrow (iv).

Assume that v is a cut vertex of G and let A and B be two components of $G - v$. If v is the start vertex of the GS ordering σ of G , then the leftmost vertices of A and B in

σ are neighbors of v in the \mathcal{L} -tree of σ . Therefore, v is not the \mathcal{L} -root leaf of σ and (iv) \Rightarrow (viii).

If the GS ordering σ does not start with the cut vertex v , then w.l.o.g. we may assume that σ starts with a vertex of A . Then the parent of v in the \mathcal{F} -tree and in the \mathcal{L} -tree of σ is an element of A . Furthermore, the first vertex of B in σ is a child of v in the \mathcal{F} -tree and in the \mathcal{L} -tree of σ . Therefore, v is neither an \mathcal{F} -branch leaf nor an \mathcal{L} -branch leaf of a GS ordering of G . Hence, (v) \Rightarrow (viii) and (vi) \Rightarrow (viii).

As DFS orderings and MCS orderings are also GS orderings, the statements (i) and (iii) directly imply (iv) and, thus, they also imply (viii).

To show that (viii) implies (iii), assume that v is not a cut vertex. Then let ρ be a vertex ordering of G that starts with v and has all neighbors of v to the right of all non-neighbors of v . Let σ be the $\text{MCS}^+(\rho)$ ordering of G . Assume for contradiction that v has more than one child in the \mathcal{L} -tree T of σ . Let x be the leftmost child of v in σ and let y be another child of v in T . Since T is an \mathcal{L} -tree and v is the first vertex of σ , both x and y have only one neighbor to its left in σ , namely v . As v is not a cut vertex, there is an x - y -path P in $G - v$. Let z be the vertex nearest to x on P that is to the right of y in σ . Vertex z exists since v is the only neighbor of y to the left of y in σ and, thus, the neighbor of y in P is not to the left of y in σ . Vertex z can only have one neighbor to the left of y since, otherwise, σ would not be an MCS ordering. By the choice of z , this neighbor lies on P . Thus, vertex z is not a neighbor of v and z is to the left of y in ρ . This is a contradiction as y and z have both exactly one neighbor to the left of y and, thus, $\text{MCS}^+(\rho)$ would have numbered z instead of y . Hence, v is an \mathcal{L} -root leaf of the MCS ordering σ .

Statement (ii) trivially implies (i). It remains to show that (viii) implies (ii). We show the contraposition. Let σ be a DFS ordering starting with v and let T be the \mathcal{L} -tree of σ . Assume v has two children in T . Any path in G between these two children runs through v , due to Lemma 2. Thus, v is a cut vertex of G . \square

Note that DFS differs from GS and MCS in this result. While for the latter three searches it is possible that a vertex is not the \mathcal{L} -root leaf of a search ordering starting with that vertex, this is not possible for DFS.

Since DFS, LDFS, MCS, and MNS orderings are also GS orderings, Theorem 6 directly implies that we can characterize the \mathcal{L} -leaves of these orderings.

Theorem 7. *For any search $\mathcal{A} \in \{GS, DFS, LDFS, MCS, MNS\}$ and any vertex v of a connected graph G with $n(G) \geq 2$, the following statements are equivalent.*

- (i) *Vertex v is the \mathcal{L} -root leaf of some \mathcal{A} -ordering of G .*
- (ii) *Vertex v is an \mathcal{L} -leaf of some \mathcal{A} -ordering of G .*
- (iii) *Vertex v is not a cut vertex of G .*

As we can check in linear time whether a vertex is a cut vertex, we can also recognize \mathcal{L} -leaves of GS, DFS, LDFS, MCS, and MNS within this time bound. However, we will see in Corollary 10 that at least for DFS the recognition of \mathcal{L} -branch leaves is NP-complete.

The characterization of \mathcal{L} -root leaves given in Theorem 6 does not work for BFS as the following theorem shows.

Theorem 8. *Let G be a connected graph with $n(G) \geq 2$. A vertex $v \in V(G)$ is the \mathcal{L} -root leaf of some BFS ordering of G if and only if $G[N_G(v)]$ is connected.*

Proof. First assume that $G[N_G(v)]$ is connected. Let σ be a GS ordering of $G[N_G(v)]$. The ordering $v \dashv\vdash \sigma$ is a prefix of a BFS ordering of G and all vertices of $N_G(v)$ except from the first in σ have a neighbor to its left in σ . Thus, v has only one child in the \mathcal{L} -tree of every BFS ordering starting with $v \dashv\vdash \sigma$.

Now assume $G[N_G(v)]$ is not connected. Let A and B be two components of $G[N_G(v)]$ and let σ be an arbitrary BFS ordering of G starting with v . The leftmost vertices of A and B in σ are neighbors of v and, thus, they are children of v in the \mathcal{L} -tree of σ . Therefore, v is not an \mathcal{L} -root leaf of σ . \square

4 NP-Hardness of Branch Leaf Recognition

4.1 Branch Leaves of DFS

DFS \mathcal{L} -trees can be recognized in linear time [18, 21]. As we have seen in Theorem 7, this also holds for DFS \mathcal{L} -leaves. In contrast, recognizing DFS \mathcal{L} -branch leaves of a graph is as hard as the recognition of DFS end-vertices since the two concepts are equivalent.

Theorem 9. *A vertex $v \in V(G)$ of a graph G is an \mathcal{L} -branch leaf of some DFS ordering of G if and only if v is the end-vertex of some DFS ordering of G .*

Proof. If v is the end-vertex of some DFS ordering σ of G , then it is also a branch leaf of the \mathcal{L} -tree of σ .

For the other direction, assume that v is a branch leaf of the \mathcal{L} -tree T of the DFS ordering σ of G starting with the vertex r . Consider a DFS ordering τ of T starting with r with the following constraint. Whenever a vertex has more than one unnumbered child, then the children that are not ancestors of v are to the left of v in σ . This means that the DFS always numbers non-ancestors of v before ancestors of v if this is possible. Due to Lemma 3, σ is also a DFS ordering of G .

We claim that v is the end-vertex of σ . Assume for contradiction that $w \neq v$ is the end-vertex of σ . Then, w is a branch leaf of T . Therefore, w is not an ancestor of v . Let x be the common ancestor of v and w that has the largest distance to the root vertex r in T . Let v' be the child of x that is an ancestor of v and let w' be the child of x that is an ancestor of w (v' could be equal to v and w' could be equal to w). By construction of σ , the vertex w' is to the left of v' in σ . Since the descendants of w' appear consecutively directly after w' in σ , w is to the left of v in σ ; a contradiction. \square

Charbit et al. [7] gave sufficient conditions on a graph class \mathcal{G} such that the end-vertex problem of DFS is NP-complete on \mathcal{G} . Due to Theorem 9, we can replace the term end-vertex in their result by the term \mathcal{L} -branch leaf.

Corollary 10. *Let \mathcal{G} be a graph class that is closed under the insertion of universal vertices. If the Hamiltonian path problem is NP-complete on \mathcal{G} , then the problem of deciding whether a vertex of a graph $G \in \mathcal{G}$ is an \mathcal{L} -branch leaf of some DFS ordering of G is NP-complete. In particular, the problem is NP-complete on split graphs.*

A similar result can be given for \mathcal{F} -branch leaves of DFS. By adapting the proof given in [30] that \mathcal{F} -trees of DFS are hard to recognize, we can show that the same holds for \mathcal{F} -branch leaves of DFS.

Theorem 11. *Let \mathcal{G} be a graph class that is closed under the insertion of universal vertices and leaves. If the Hamiltonian path problem is NP-complete on \mathcal{G} , then the problem of deciding whether a vertex of a graph $G \in \mathcal{G}$ is an \mathcal{F} -branch leaf of some DFS ordering of G is NP-complete. In particular, the problem is NP-complete on chordal graphs.*

Proof. Let G be a graph in \mathcal{G} with the vertex set $\{v_1, \dots, v_n\}$. W.l.o.g. we may assume that $n \geq 2$. We construct the graph G' as follows: First we add the vertex v'_i for all $i \in \{1, \dots, n\}$ and connect it to v_i . Then we add the vertex y which is adjacent to all other vertices. Due to the conditions on the graph class \mathcal{G} , the graph G' is in \mathcal{G} . We claim that there is a DFS ordering of G' with \mathcal{F} -branch leaf y if and only if there is a Hamiltonian path in G .

If there is an Hamiltonian path in G , then there is a DFS ordering of G' that starts with this path. In the \mathcal{F} -tree of such an ordering, vertex y is a branch leaf.

Now assume that y is a branch leaf of the \mathcal{F} -tree T of some DFS ordering σ of G . Assume for contradiction that $v \in V(G)$ is to the right of y in σ . As v' is not a child of y in T , vertex v' is to the left of y in σ . Thus, v' is to the left of both its neighbors in σ and, thus, v' must be the start vertex of σ with $s \neq x_i$ for all $i \in \{1, \dots, k+3\}$. However, this implies that y is the second vertex of σ and, since $n \geq 2$, vertex y has at least one child in T ; a contradiction. It follows that all vertices $v \in V(G)$ are to the left of y in σ . Now let T' be the \mathcal{L} -tree of σ . Due to Lemma 2, all the vertices of G are ancestors of y in T' and, thus, they all lie on a path P from the root of T' to y . If some vertex v' lies on this path, then either v' is the first vertex of P or the successor of v' in σ is vertex y . Thus, only the first and the last vertex of the path P could be one of these vertices. Therefore, P contains a Hamiltonian path of G .

The Hamiltonian path problem is NP-complete on chordal graphs [5]. Furthermore, chordal graphs are closed under the addition of leaves and universal vertices as neither universal vertices nor leaves can be an element of an induced cycle of length ≥ 4 . \square

If we compare Corollary 10 and Theorem 11, then we see that for \mathcal{L} -branch leaves it is sufficient that the graph class \mathcal{G} is closed under the addition of universal vertices while for \mathcal{F} -branch leaves we have the additional condition that \mathcal{G} is closed under the addition of leaves. We cannot omit this constraint (unless $P = NP$) as the \mathcal{F} -branch leaf recognition problem of DFS can be solved in polynomial time on split graphs (see Corollary 28).

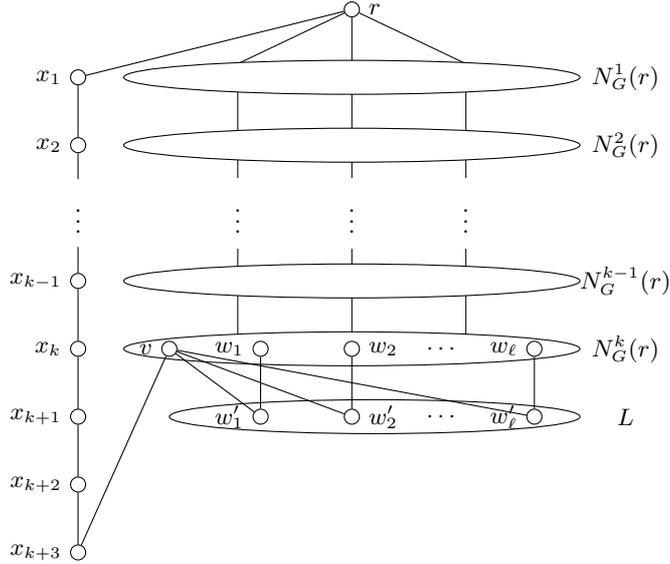


Figure 1: A graphical representation of the graph G' constructed in the proof of Theorem 12.

4.2 Branch Leaves of BFS

The end-vertex problem of BFS is NP-complete, even if the graph is bipartite and the start vertex of the BFS ordering is fixed [7]. This fact can be used to show that recognizing BFS \mathcal{F} -branch leaves is also NP-complete.

Theorem 12. *It is NP-complete to decide whether a vertex of a bipartite graph G is an \mathcal{F} -branch leaf of some BFS ordering of G .*

Proof. We reduce the beginning-end-vertex problem of BFS on bipartite graphs to the respective \mathcal{F} -branch leaf recognition problem. Given a connected bipartite graph G and a vertex $r \in V(G)$, the problem asks whether a vertex $v \in V(G) \setminus \{r\}$ can be the end-vertex of some BFS ordering of G starting with r . The problem is NP-complete [7].

Let G and $r, v \in V(G)$ be an input of this problem. Let k be the eccentricity of r . W.l.o.g. we may assume that v is in $N_G^k(r)$ since, otherwise, v is obviously not the end-vertex of any BFS ordering of G starting with r . We construct the graph G' from G as follows (see Fig. 1). We add a path $P = (r, x_1, \dots, x_{k+3})$ of length $k + 3$ to G as well as the edge vx_{k+3} . Furthermore, for every vertex $w \in N_G^k(r) \setminus \{v\}$ we add a vertex w' and the edges vw' and $w w'$ to G . We collect these vertices w' in the set L . Note that G' is also bipartite.

First assume that v is the end-vertex of some BFS ordering σ of G starting with r . Let $\rho = (x_1, x_2, \dots, x_{k+3}) ++ \sigma$ and let τ be the $\text{BFS}^+(\rho)$ ordering of G' . By the choice of ρ , the ordering τ starts with x_1 . Thus, every vertex x_i is in layer $i - 1$ and every vertex $v \in N_G^k(r)$ is in layer $j + 1$. The choice of ρ also implies that every vertex x_i is the first vertex of its layer in τ . In particular, vertex x_{k+2} is to the left of v in τ and, thus,

x_{k+2} is the parent of x_{k+3} in the \mathcal{F} -tree of τ . Furthermore, x_{k+3} is to the right of v in τ since $\text{dist}_{G'}(x_1, x_{k+3}) = k + 2$ and $\text{dist}(x_1, v) = k + 1$. For the same reason, the vertices $w' \in L$ are also to the right of any vertex of $V(G)$ in τ . Therefore, the restriction of τ to the vertices of the graph G is exactly σ and, thus, every $w \in N_G^k(r) \setminus \{v\}$ is to the left of v in τ . Hence, v is an \mathcal{F} -branch leaf of τ .

Now assume that v is an \mathcal{F} -branch leaf of some BFS ordering σ of G' . First we show that σ starts with a vertex x_i . Assume for contradiction that this is not the case and let s be the start vertex of σ . If every shortest path from s to x_{k+3} runs through v , then v is the parent of x_{k+3} in the \mathcal{F} -tree of σ ; a contradiction. Thus, we may assume that $\text{dist}_{G'}(s, x_{k+2}) < \text{dist}_{G'}(s, x_{k+3})$. This implies that $\text{dist}_{G'}(s, x_{k+2}) = \text{dist}_G(s, r) + k + 2$. However, $\text{dist}_{G'}(s, v) \leq \text{dist}_G(s, r) + \text{dist}_G(r, v) = \text{dist}_G(s, r) + k$. Hence, v is to the left of x_{k+2} in σ and, therefore, v is the parent of x_{k+3} in the \mathcal{F} -tree of σ ; a contradiction.

Thus, we may assume that σ starts with some vertex x_i . For any vertex $w' \in L$, it holds $\text{dist}_{G'}(x_i, w') > \text{dist}_{G'}(x_i, v)$. Thus, all the elements of L are to the right of v in σ . Therefore, all the vertices in $N_G^k(r)$ are to the left of v in σ since v is an \mathcal{F} -branch leaf of σ . If there is any vertex $y \in V(G) \setminus N_G^k(r)$ to the right of v in σ , then $\text{dist}_{G'}(x_i, y) \geq \text{dist}_{G'}(x_i, v)$. Thus, the shortest path from x_i to v runs through x_{k+3} which implies that $\text{dist}_{G'}(x_i, x_{k+3}) < \text{dist}(x_i, z)$ for all $z \in N_G^{k-1}(r)$. However, as x_{k+3} is adjacent to v in G but not adjacent to any other element of $N_G^k(r)$, vertex v is the leftmost vertex of $N_G^k(r)$ in σ ; a contradiction to the observation above.

Hence, v is the rightmost vertex of $V(G)$ in σ . Since σ starts with some x_i , vertex r is the leftmost vertex of $V(G)$ in σ . Let σ^* be the restriction of σ to the vertices of G . The ordering σ^* starts with r and ends with v . None of the vertices in $V(G') \setminus V(G)$ had an influence on the order of the vertices of G in σ . Thus, σ^* is a BFS ordering of G starting with r and ending with v . \square

In contrast to this result, there is a simple characterization of BFS \mathcal{L} -branch leaves of bipartite graphs.

Theorem 13. *Let G be a connected bipartite graph with $n(G) \geq 2$. A vertex $v \in V(G)$ is an \mathcal{L} -branch leaf of some BFS ordering of G if and only if there is an $r \in V(G) \setminus \{v\}$ such that $\text{dist}_G(r, w) = \text{dist}_{G-v}(r, w)$ for all $w \in V(G) \setminus \{v\}$.*

Proof. Assume that there is a vertex $r \in V(G) \setminus \{v\}$ such that $\text{dist}_G(r, w) = \text{dist}_{G-v}(r, w)$ for all $w \in V(G) \setminus \{v\}$. Let $(r = w_0, \dots, w_k = v)$ be a shortest path from r to v , i.e., v has distance k to r . It is easy to see that there is a BFS ordering σ of G in which every vertex w_i is the first vertex of the i -th layer. Let T be the \mathcal{L} -tree of σ and let x be a vertex in the $(k + 1)$ -th layer. Due to the condition on r , there is a shortest path from r to x in G that does not use vertex v . Therefore, x has a neighbor y in the k -th layer that is not v . Since $v \prec_\sigma y \prec_\sigma x$, vertex v is not the parent of x in T . Since G is bipartite, the layers of σ are independent sets and, thus, v is neither the parent of any vertex in the k -th layer. Hence, v is a leaf of T .

Now assume that v is a branch leaf of the \mathcal{L} -tree T of the BFS ordering σ . Let r be the start vertex of σ . Let w be a vertex different from v and r . Consider the r - w -path P in T . Since G is bipartite, the edges of G and, thus, the edges of T only connect vertices

of consecutive layers. Furthermore, every vertex has a neighbor in its preceding layer. Thus, P has $\text{dist}_G(r, w)$ edges. Since v is a leaf of T , P does not contain v . Therefore, P is also contained in $G - v$ and $\text{dist}_{G-v}(r, w) = \text{dist}_G(r, w)$. \square

To check whether the condition of Theorem 13 is fulfilled, we simply make two all-pair-shortest paths computations and compare the results. This can be done in $\mathcal{O}(n(G) \cdot m(G))$ by using $\mathcal{O}(n(G))$ many BFS computations.

Corollary 14. *Given a connected bipartite graph G and a vertex $v \in V(G)$, we can decide in time $\mathcal{O}(n(G) \cdot m(G))$ whether v is the \mathcal{L} -branch leaf of some BFS ordering of G .*

The results of Theorem 12 and Corollary 14 are quite surprising since the \mathcal{L} -tree recognition problem of BFS is NP-hard on bipartite graphs [30] while the \mathcal{F} -tree recognition problem of BFS can be solved in linear time [19, 25].

4.3 Branch Leaves of MNS-like Searches

For several subsearches of MNS, the recognition problem of \mathcal{F} -branch leaves is NP-complete on weakly chordal graphs.

Theorem 15. *Let \mathcal{A} be one of the following searches: LBFS, LDFS, MCS, MNS. It is NP-complete to decide whether a vertex of a weakly chordal graph G is an \mathcal{F} -branch leaf of some \mathcal{A} -ordering.*

Proof. The proof of the theorem is inspired by the NP-completeness proof of the \mathcal{F} -tree recognition problem of MNS given by Beisegel et al. [2]. We construct a polynomial-time reduction from 3-SAT. Let \mathcal{I} be an instance of 3-SAT. W.l.o.g. we may assume that \mathcal{I} contains at least two clauses. We construct the corresponding graph $G(\mathcal{I})$ as follows. Let $X = \{x_1, \dots, x_k, \bar{x}_1, \dots, \bar{x}_k\}$ be the set of vertices representing the literals of \mathcal{I} . The graph $G(\mathcal{I})[X]$ forms the complement of the matching in which x_i is matched to \bar{x}_i for every $i \in \{1, \dots, k\}$. Let $C = \{c_1, \dots, c_\ell\}$ be the set of vertices representing the clauses of \mathcal{I} . The set C forms an independent set in $G(\mathcal{I})$ and every clause vertex c_i is adjacent to each vertex of X whose corresponding literal is contained in the clause associated with c_i . Additionally, we add a universal vertex t .

Assume $G(\mathcal{I})$ has a fulfilling assignment \mathcal{B} . Then we create the following \mathcal{A} -ordering σ . We first number all literal vertices of literals that are set to true in \mathcal{B} and then we number t . Since these vertices form a clique, this ordering is a prefix of an \mathcal{A} -ordering. We number the remaining vertices following an arbitrary \mathcal{A} -ordering. As \mathcal{B} is fulfilling, all clause vertices and all literal vertices have a neighbor that is to the left of t in σ . Thus, t is an \mathcal{F} -branch leaf of σ .

Now assume that t is an \mathcal{F} -branch leaf of the \mathcal{A} -ordering σ of $G(\mathcal{I})$. Let S be the set of literal vertices that are to the left of t in σ . Since t is universal and the edges $x_i \bar{x}_i$ are missing, the set S contains at most one literal vertex for every variable. Thus, we can define an assignment \mathcal{B} by giving all literals whose vertices are contained in S the value true. If some variable value is not fixed, then we choose an arbitrary value for the

variable. If a clause vertex has a parent in the \mathcal{F} -tree T of σ , then this parent is an element of S since t is a leaf in T . If the clause vertex c_i does not have a parent in T , then c_i is the first vertex of σ . Since there are at least two clause vertices, the second vertex of σ is not t but a literal vertex adjacent to c_i . Therefore, every clause vertex has a neighbor in S and, thus, \mathcal{B} is a fulfilling assignment.

To see that $G(\mathcal{I})$ is weakly chordal, we first observe that every pair (x_i, \bar{x}_i) forms a two-pair in $G(\mathcal{I})$. Spinrad and Sritharan [31] showed that the graph that results from the addition of an edge between a two-pair is weakly chordal if and only if the initial graph is weakly chordal. If we add all the edges $x_i\bar{x}_i$, then the resulting graph is a split graphs and, thus, $G(\mathcal{I})$ is weakly chordal. \square

5 Branch Leaves and Chordal Graphs

5.1 Branch Leaves of MNS-like Searches

MNS and all of its subsearches compute PEOs of chordal graphs [13, 33]. Thus, any \mathcal{F} -tree or \mathcal{L} -tree of an MNS ordering is also an \mathcal{F} -tree or \mathcal{L} -tree of some PEO. Beisegel et al. [2] showed that this also holds the other way around for a large family of graph searches including LBFS, LDFS, MCS, and MNS, i.e., the rooted \mathcal{F} -trees and rooted \mathcal{L} -trees of these searches on chordal graphs are exactly the rooted \mathcal{F} -trees and rooted \mathcal{L} -trees of PEOs, respectively. Therefore, we will only characterize \mathcal{F} -branch leaves and \mathcal{L} -branch leaves of PEOs.

We start by showing that the \mathcal{L} -branch leaves of PEOs of a chordal graph are exactly the graph's simplicial vertices.

Theorem 16. *Let G be a connected chordal graph with $n(G) \geq 2$. A vertex $v \in V(G)$ is an \mathcal{L} -branch leaf of some PEO of G if and only if v is simplicial.*

Proof. If v is a simplicial vertex, then there is a PEO σ that ends with v . Vertex v is an \mathcal{L} -branch leaf of σ .

For the other direction, let σ be a PEO and let v be a non-simplicial vertex of G . Hence, not all neighbors of v are to the left of v in σ . Let w be the leftmost neighbor of v in σ that is to the right of v in σ . Let x be the parent of w in the \mathcal{L} -tree T of σ . If x is not equal v , then it holds $v \prec_\sigma x \prec_\sigma w$. As σ is a PEO and $vw, xw \in E(G)$, the edge vx is also in $E(G)$; a contradiction to the choice of w . Hence, v is the parent of w in T and v is not an \mathcal{L} -branch leaf of σ . \square

Since we can decide in linear time whether a vertex is simplicial [1], we can recognize \mathcal{L} -branch leaves of PEOs in linear time.

Corollary 17. *Given a connected chordal graph G and a vertex $v \in V(G)$, we can decide in time $\mathcal{O}(n(G) + m(G))$ whether v is the \mathcal{L} -branch leaf of some PEO of G . Therefore, we can also decide in time $\mathcal{O}(n(G) + m(G))$ whether v is the \mathcal{L} -branch leaf of some LBFS, LDFS, MCS, or MNS ordering.*

Obviously, simplicial vertices are also \mathcal{F} -branch leaves of PEOs. However, there are further \mathcal{F} -branch leaves.

Theorem 18. *Let G be a connected chordal graph with $n(G) \geq 2$. A vertex $v \in V(G)$ is an \mathcal{F} -branch leaf of some PEO of G if and only if the graph $G[N_G(v)]$ has a dominating clique.*

Proof. First assume that $G[N_G(v)]$ has a dominating clique C . It is obvious that there is an LBFS ordering σ of G that starts with the vertices of C . The ordering σ is a PEO. Since all neighbors of v have a neighbor in C or are elements of C , v is an \mathcal{F} -branch leaf of σ .

Now let v be an \mathcal{F} -branch leaf of the PEO σ . Let S be the set of neighbors of v that are to the left of v in σ . The set S induces a clique of G . Thus, if $S = N_G(v)$, then $G[N_G(v)]$ is a clique and we are done. Hence, we may assume that there is a vertex $w \in N_G(v) \setminus S$. As w is not a child of v in the \mathcal{F} -tree of σ , there is a vertex $x \in N_G(w)$ with $x \prec_\sigma v$. Since σ is a PEO, vertex x is a neighbor of v and, thus, $x \in S$. Thus, any neighbor of v that is not in S has a neighbor in S and, hence, S induces a dominating clique of $G[N_G(v)]$. \square

To decide the complexity of the \mathcal{F} -branch leaf recognition problem of PEOs, we examine the complexity of deciding the existence of a dominating clique in a chordal graph. Kratsch et al. [22] showed that such a clique exists if and only if the diameter of the graph is at most three.

Theorem 19 (Kratsch et al. [22]). *A chordal graph G has a dominating clique if and only if the diameter of G is at most three.*

As the diameter of a graph can be determined by computing $n(G)$ many BFS orderings, we can decide the existence of a dominating clique in a chordal graph in polynomial time. Although it is unlikely that the diameter of a chordal graph can be computed in linear time,² we can improve our algorithm to decide the existence of a dominating clique in linear time. To this end, we can use the following result of Corneil et al. [10].

Theorem 20 (Corneil et al. [10]). *Let G be a chordal graph and let $v \in V(G)$ be the end-vertex of some LBFS ordering of G . If $\text{ecc}(v) < \text{diam}(G)$, then $\text{ecc}(v)$ is even and $\text{ecc}(v) = \text{diam}(G) - 1$.*

Combining Theorems 18 to 20, we can give a linear-time recognition algorithm for \mathcal{F} -branch leaves of PEOs.

Corollary 21. *Given a chordal graph G and a vertex $v \in V(G)$, we can decide in time $\mathcal{O}(n(G) + m(G))$ whether v is an \mathcal{F} -branch leaf of some PEO of G . Therefore, we can also decide in time $\mathcal{O}(n(G) + m(G))$, whether v is the \mathcal{F} -branch leaf of some LBFS, LDFS, MCS, or MNS ordering.*

²Even on split graphs, the diameter cannot be computed in subquadratic time unless the Strong Exponential Time Hypothesis fails [6].

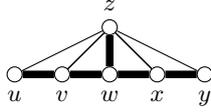


Figure 2: The given graph G is chordal. There is no dominating clique in the graph $G[N_G(z)]$. However, the given spanning tree is the \mathcal{F} -tree of the BFS ordering (w, v, x, z, u, y) and, thus, z is a \mathcal{F} -branch leaf of BFS.

Proof. Due to Theorems 18 and 19, it is sufficient to check whether $G' = G[N_G(v)]$ has diameter 3. We compute an LBFS ordering σ of G' in linear time. Let v be the end-vertex of σ . We compute the eccentricity of v in G' in linear time by starting a BFS in v . If $\text{ecc}_{G'}(v) > 3$, then the diameter of G' is larger than 3. If $\text{ecc}_{G'}(v) = 3$, then, by Theorem 20, $\text{diam}(G') = 3$. If $\text{ecc}_{G'}(v) < 3$, then $\text{diam}(G') \leq \text{ecc}_{G'}(v) + 1 \leq 3$, due to Theorem 20. \square

5.2 Branch Leaves of BFS

The condition given in Theorem 18 is also sufficient for a vertex to be a BFS \mathcal{F} -branch leaf since every LBFS ordering is also a BFS ordering. However, it is not necessary as can be seen in Fig. 2. To characterize BFS \mathcal{F} -branch leaves of chordal graphs, we start with the following two lemmas.

Lemma 22. *Let G be a chordal graph and let r be a vertex in $V(G)$. Let x and y be two vertices in $N_G^i(r)$. If there is a vertex $z \in N_G^{i+1}(r)$ which is adjacent to both x and y , then $xy \in E(G)$.*

Proof. Let σ be an LBFS ordering of G starting with r . Vertex z is to the right of both x and y in σ . Since σ is a PEO, the vertices x and y are adjacent. \square

Lemma 23. *Let G be a chordal graph and let r be a vertex in $V(G)$. Let x and y be two vertices in $N_G^i(r)$. If $xy \in E(G)$, then $N_G(x) \cap N_G^{i-1}(r) \subseteq N_G(y)$ or $N_G(y) \cap N_G^{i-1}(r) \subseteq N_G(x)$.*

Proof. Let σ be an LBFS ordering starting with r . W.l.o.g. we may assume that $x \prec_\sigma y$. All the vertices of $N_G^{i-1}(r)$ are to the left of y in σ . Since σ is a PEO, all the neighbors of y in $N_G^{i-1}(r)$ are adjacent to x and, thus, $N_G(y) \cap N_G^{i-1}(r) \subseteq N_G(x)$. \square

The next lemma makes a statement about the distances of neighbors of a vertex in a chordal graph.

Lemma 24. *Let G be a connected chordal graph and let $v \in V(G)$. For any $x, y \in N_G(v)$, the distance between x and y in $G-v$ is equal to the distance between x and y in $G[N_G(v)]$.*

Proof. Let P be a shortest x - y -path in $G - v$. If P is a subgraph of $G[N_G(v)]$, then we are done. Otherwise, let u be the first vertex of P (starting from x) whose successor on P is not in $N_G(v)$ and let w be the first element of P after u that is an element of



Figure 3: Two cases of the proof of Theorem 25. The vertical arrangement of the vertices represent their layers. Thick edges are edges of the \mathcal{F} -tree. Dotted edges are not present. Dashed edges are implied by either Lemma 22 or Lemma 23.

$N_G(v)$. Note that u could be equal to x and w could be equal to y . The subpath of P between u and w contains at least three vertices. Therefore, this subpath and the edges uv and vw form an induced cycle of length at least four in G ; a contradiction. \square

Using Lemmas 22 to 24, we characterize BFS \mathcal{F} -branch leaves of chordal graphs.

Theorem 25. *Let G be a connected chordal graph with $n(G) \geq 2$. A vertex $v \in V(G)$ is an \mathcal{F} -branch leaf of some BFS ordering of G if and only if the radius of $G[N_G(v)]$ is at most two.*

Proof. First assume that $G[N_G(v)]$ has radius two and let w be a central vertex of $G[N_G(v)]$. There is a BFS ordering σ that starts with w followed by all neighbors of w that are not v . Vertex v is an \mathcal{F} -branch leaf of σ since all neighbors of v have some neighbor in $N_G[w] \setminus \{v\}$ or are equal to w .

Now assume that $v \in N_G^i(r)$ is an \mathcal{F} -branch leaf of some BFS ordering σ of G starting with r . Let T be the \mathcal{F} -tree of σ rooted in r and let v' be the parent of v in T . Since T is an BFS \mathcal{F} -tree rooted in r , it holds that $v' \in N_G^{i-1}(r)$.

We claim that in $G - v$ vertex v' has a distance of at most two to every element of $N_G(v)$. Let $w \in N_G(v) \setminus \{v'\}$. If $v'w \in E(G)$, then v' and w have distance one in $G - v$. Therefore, we may assume in the following that $v'w \notin E(G)$. Then Lemma 22 implies that $w \notin N_G^{i-1}(r)$. Furthermore, the parent of w in T , say w' , is different from v and v' . If $v'w' \in E(G)$, then v' and w have distance two in $G - v$ via the path (v', w', w) . Thus, we may also assume that $v'w' \notin E(G)$.

First assume that $w \in N_G^i(r)$ (see left part of Fig. 3). Then $w' \in N_G^{i-1}(r)$. Lemma 23 implies that w' is adjacent to v because vw, vv' , and $ww' \in E(G)$ but $v'w \notin E(G)$. Now the non-existence of $v'w'$ contradicts Lemma 22.

Now assume that $w \in N_G^{i+1}(r)$ (see right part of Fig. 3). Then, due to Lemma 22, $vw' \in E(G)$. Since $v'w' \notin E(G)$, the parent of w' , say w'' , is different from v' . Since w' is the parent of w , it holds that $w' \prec_\sigma v$. This implies that $w'' \prec_\sigma v'$. Therefore, w'' is not adjacent to v since, otherwise, v' would not be the parent of v . The non-existence of both $v'w'$ and vw'' contradicts Lemma 23.

Summarizing, v' has distance at most two in $G - v$ to any neighbor of v . By Lemma 24, v' has distance at most two in $G[N_G(v)]$ to any neighbor of v in G and, thus, $G[N_G(v)]$ has radius at most two. \square

Chepoi and Dragan [8] presented a linear-time algorithm that computes a central vertex of a chordal graph. As the eccentricity of such a vertex can be computed in linear time using BFS, we can compute the radius of a chordal graph and, in particular, of $G[N_G(v)]$ in linear time. Thus, Theorem 25 implies a linear-time algorithm for the BFS \mathcal{F} -branch leaf recognition on chordal graphs.

Corollary 26. *Given a connected chordal graph G and a vertex $v \in V(G)$, we can decide in time $\mathcal{O}(n(G) + m(G))$ whether v is an \mathcal{F} -branch leaf of some BFS ordering of G .*

5.3 Branch Leaves of DFS

As we have seen in Theorem 11, the \mathcal{F} -branch leaf recognition problem of DFS is NP-complete on chordal graphs. However, there is a simple characterization of DFS \mathcal{F} -branch leaves of split graphs.

Theorem 27. *Let G be a connected split graph with $n(G) \geq 2$. A vertex $v \in V(G)$ is an \mathcal{F} -branch leaf of some DFS ordering if and only if v is not a cut vertex of G .*

Proof. Due to Theorem 6, no cut vertex of a graph G is an \mathcal{F} -branch leaf of any GS ordering. It remains to show that every vertex v of a connected split graph G for which $G - v$ is connected is an \mathcal{F} -branch leaf of some DFS ordering of G . For the connected split graph with two vertices, this is straightforwardly true. Thus, we may assume in the following that $n(G) \geq 3$. Let (C, I) be a the partition of $V(G)$ into a clique C and the independent set I . First note that C must contain at least one vertex that is not v since, otherwise, v would be a cut vertex of G . Obviously, there is a DFS ordering σ of G starting with the vertices of $C \setminus \{v\}$. As v is not a cut vertex, every vertex of I has at least one neighbor in $C \setminus \{v\}$. Thus, v is an \mathcal{F} -branch leaf of σ . \square

As cut vertices can be identified in linear time, Theorem 27 leads directly to a linear-time algorithm for the DFS \mathcal{F} -branch leaf recognition on split graphs.

Corollary 28. *Given a connected split graph G and a vertex $v \in V(G)$, we can decide in time $\mathcal{O}(n(G) + m(G))$ whether v is an \mathcal{F} -branch leaf of some DFS ordering of G .*

In contrast to this result, it is NP-hard to decide whether a vertex of a split graph is a DFS \mathcal{L} -branch leaf (see Corollary 10). Thus, the \mathcal{L} -branch leaf recognition of DFS seems to be harder than the \mathcal{F} -branch leaf recognition of DFS, a surprising contrast to the hardness of the DFS \mathcal{F} -tree recognition problem [30] and the easiness of the DFS \mathcal{L} -tree recognition problem [18, 21]. Recall that we have made a similar observation for the complexity of the branch leaf and tree recognition of BFS (see Theorem 12 and Corollary 14).

References

- [1] Jesse Beisegel, Carolin Denkert, Ekkehard Köhler, Matjaž Krnc, Nevena Pivač, Robert Scheffler, and Martin Strehler. On the end-vertex problem of graph searches. *Discrete Mathematics and Theoretical Computer Science*, 21(1), 2019. doi:10.23638/DMTCS-21-1-13.
- [2] Jesse Beisegel, Carolin Denkert, Ekkehard Köhler, Matjaž Krnc, Nevena Pivač, Robert Scheffler, and Martin Strehler. The recognition problem of graph search trees. *SIAM Journal on Discrete Mathematics*, 35(2):1418–1446, 2021. doi:10.1137/20M1313301.
- [3] Jesse Beisegel, Ekkehard Köhler, Robert Scheffler, and Martin Strehler. Linear time LexDFS on chordal graphs. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms (ESA 2020)*, volume 173 of *LIPICs*, pages 13:1–13:13, Dagstuhl, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPICs.ESA.2020.13.
- [4] Anne Berry, Jean R.S. Blair, Pinar Heggernes, and Barry W. Peyton. Maximum cardinality search for computing minimal triangulations of graphs. *Algorithmica*, 39(4):287–298, 2004. doi:10.1007/s00453-004-1084-3.
- [5] Alan A. Bertossi and Maurizio A. Bonuccelli. Hamiltonian circuits in interval graph generalizations. *Information Processing Letters*, 23(4):195–200, 1986. doi:10.1016/0020-0190(86)90135-3.
- [6] Michele Borassi, Pierluigi Crescenzi, and Michel Habib. Into the square: On the complexity of some quadratic-time solvable problems. In Pierluigi Crescenzi and Michele Loreti, editors, *Proceedings of ICTCS 2015, the 16th Italian Conference on Theoretical Computer Science*, volume 322 of *ENTCS*, pages 51–67, 2016. doi:10.1016/j.entcs.2016.03.005.
- [7] Pierre Charbit, Michel Habib, and Antoine Mamcarz. Influence of the tie-break rule on the end-vertex problem. *Discrete Mathematics and Theoretical Computer Science*, 16(2):57, 2014. doi:10.46298/dmtcs.2081.
- [8] Victor Chepoi and Feodor Dragan. A linear-time algorithm for finding a central vertex of a chordal graph. In Jan van Leeuwen, editor, *Algorithms — ESA '94*, volume 855 of *LNCS*, pages 159–170, Berlin, Heidelberg, 1994. Springer. doi:10.1007/BFb0049406.
- [9] Derek G. Corneil, Barnaby Dalton, and Michel Habib. LDFS-based certifying algorithm for the minimum path cover problem on cocomparability graphs. *SIAM Journal on Computing*, 42(3):792–807, 2013. doi:10.1137/11083856X.
- [10] Derek G. Corneil, Feodor F. Dragan, Michel Habib, and Christophe Paul. Diameter determination on restricted graph families. *Discrete Applied Mathematics*, 113(2):143–166, 2001. doi:10.1016/S0166-218X(00)00281-X.

- [11] Derek G. Corneil, Jérémie Dusart, Michel Habib, Antoine Mamcarz, and Fabien De Montgolfier. A tie-break model for graph search. *Discrete Applied Mathematics*, 199:89–100, 2016. doi:10.1016/j.dam.2015.06.011.
- [12] Derek G. Corneil, Ekkehard Köhler, and Jean-Marc Lanlignel. On end-vertices of lexicographic breadth first searches. *Discrete Applied Mathematics*, 158(5):434–443, 2010. doi:10.1016/j.dam.2009.10.001.
- [13] Derek G. Corneil and Richard M. Krueger. A unified view of graph searching. *SIAM Journal on Discrete Mathematics*, 22(4):1259–1276, 2008. doi:10.1137/050623498.
- [14] Derek G. Corneil, Stephan Olariu, and Lorna Stewart. Linear time algorithms for dominating pairs in asteroidal triple-free graphs. *SIAM Journal on Computing*, 28(4):1284–1297, 1999. doi:10.1137/S0097539795282377.
- [15] Derek G. Corneil, Stephan Olariu, and Lorna Stewart. The LBFS structure and recognition of interval graphs. *SIAM Journal on Discrete Mathematics*, 23(4):1905–1953, 2009. doi:10.1137/S0895480100373455.
- [16] Jan Gorzny and Jing Huang. End-vertices of LBFS of (AT-free) bigraphs. *Discrete Applied Mathematics*, 225:87–94, 2017. doi:10.1016/j.dam.2017.02.027.
- [17] Michel Habib, Ross McConnell, Christophe Paul, and Laurent Viennot. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234(1–2):59–84, 2000. doi:10.1016/S0304-3975(97)00241-7.
- [18] Torben Hagerup. Biconnected graph assembly and recognition of DFS trees. Technical Report A 85/03, Universität des Saarlandes, 1985. doi:10.22028/D291-26437.
- [19] Torben Hagerup and Manfred Nowak. Recognition of spanning trees defined by graph searches. Technical Report A 85/08, Universität des Saarlandes, 1985.
- [20] John Hopcroft and Robert E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974. doi:10.1145/321850.321852.
- [21] Ephraim Korach and Zvi Ostfeld. DFS tree construction: Algorithms and characterizations. In Jan van Leeuwen, editor, *Graph-Theoretic Concepts in Computer Science*, volume 344 of *LNCS*, pages 87–106, Berlin, Heidelberg, 1989. Springer. doi:10.1007/3-540-50728-0_37.
- [22] Dieter Kratsch, Peter Damaschke, and Anna Lubiw. Dominating cliques in chordal graphs. *Discrete Mathematics*, 128(1):269–275, 1994. doi:10.1016/0012-365X(94)90118-X.
- [23] Dieter Kratsch, Mathieu Liedloff, and Daniel Meister. End-vertices of graph search algorithms. In Peter Widmayer Vangelis Th. Paschos, editor, *Algorithms*

- and Complexity*, volume 9079 of *LNCS*, pages 300–312, Cham, 2015. Springer. doi:10.1007/978-3-319-18173-8_22.
- [24] Richard Krueger, Geneviève Simonet, and Anne Berry. A general label search to investigate classical graph search algorithms. *Discrete Applied Mathematics*, 159(2-3):128–142, 2011. doi:10.1016/j.dam.2010.02.011.
- [25] Udi Manber. Recognizing breadth-first search trees in linear time. *Information Processing Letters*, 34(4):167–171, 1990. doi:10.1016/0020-0190(90)90155-Q.
- [26] Guozhen Rong, Yixin Cao, Jianxin Wang, and Zhifeng Wang. Graph searches and their end vertices. *Algorithmica*, 84:2642–2666, 2022. doi:10.1007/s00453-022-00981-5.
- [27] Donald J. Rose. Triangulated graphs and the elimination process. *Journal of Mathematical Analysis and Applications*, 32(3):597–609, 1970. doi:10.1016/0022-247X(70)90282-9.
- [28] Donald J. Rose, R. Endre Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976. doi:10.1137/0205021.
- [29] Robert Scheffler. Linearizing partial search orders. In Michael A. Bekos and Michael Kaufmann, editors, *Graph-Theoretic Concepts in Computer Science*, volume 13453 of *LNCS*, pages 425–438, Cham, 2022. Springer. doi:10.1007/978-3-031-15914-5_31.
- [30] Robert Scheffler. On the recognition of search trees generated by BFS and DFS. *Theoretical Computer Science*, 936:116–128, 2022. doi:10.1016/j.tcs.2022.09.018.
- [31] Jeremy Spinrad and R. Sritharan. Algorithms for weakly triangulated graphs. *Discrete Applied Mathematics*, 59(2):181–191, 1995. doi:10.1016/0166-218X(93)E0161-Q.
- [32] Robert E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972. doi:10.1137/0201010.
- [33] Robert E. Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13(3):566–579, 1984. doi:10.1137/0213035.
- [34] Meibiao Zou, Zhifeng Wang, Jianxin Wang, and Yixin Cao. End vertices of graph searches on bipartite graphs. *Information Processing Letters*, 173, 2022. doi:10.1016/j.ip1.2021.106176.