ISSN 1855-3966 (printed edn.), ISSN 1855-3974 (electronic edn.)
ARS MATHEMATICA CONTEMPORANEA 26 (2026) #P1.08
https://doi.org/10.26493/1855-3974.3238.2a8
(Also available at http://amc-journal.eu)

On the leaves of graph search trees*

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

Received 29 September 2023, accepted 6 December 2024, published online 30 November 2025

Abstract

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.

Keywords: Graph search, graph search trees, leaves, chordal graphs.

Math. Subj. Class. (2020): 05C05, 05C85, 68Q25, 68R10

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., [6, 12, 19]).

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 each vertex different from the start vertex, exactly one edge to one of its neighbors in the graph 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

*An extended abstract of this paper appeared as *Graph Search Trees and Their Leaves* in the Proceedings of WG 2023 [36].

E-mail address: robert.scheffler@b-tu.de (Robert Scheffler)

This work is licensed under https://creativecommons.org/licenses/by/4.0/

trees are used for fast planarity testing [25]. Furthermore, trees generated by LBFS were used to design a linear-time implementation of the search LDFS for chordal graphs [5]. This is of particular interest as for this search no general linear-time implementation is known so far.

The problem of deciding whether a given spanning tree of a graph can be obtained by a particular search was introduced by Hagerup [23] in 1985, who presented a linear-time algorithm that recognizes DFS trees. In the same year, Hagerup and Nowak [24] gave a similar result for the BFS tree recognition. In 2021, Beisegel et al. [3] 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 last visited neighbor that was visited before it, 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 [35], 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. Most recently, NP-hardness proofs have been given for the recognition of $\mathcal{L}$-trees of LBFS [38] and for the recognition of $\mathcal{L}$-trees of Generic Search [4], a graph search that generalizes BFS, LBFS, DFS, LDFS, and MCS.

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. [33] 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 [18] and transitive orientations of comparability graphs [22]. 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. [15], 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., [2, 9, 21, 28, 31, 43]). A summary of these results can be found in [37, Table 1].

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 [34]. 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. This problem was first suggested in 2020 by Michel Habib during the defense of Jesse Beisegel's Ph.D. thesis [1]. 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 in Section 3 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. We contrast this by showing in Section 4 that the recognition of branch leaves of $\mathcal{L}$-trees (and of $\mathcal{F}$-trees) is NP-hard for almost all searches considered here. Besides this, we present some polynomial-time algorithms for (L)BFS branch leaves of bipartite graphs. In Section 5 we consider chordal graphs. We characterize the branch leaves of almost all considered searches on this graph class and use these characterizations to design linear-time algorithms for the leaf recognition. An overview of the results is given in Table 1.

Table 1: Complexity of the $\mathcal{F}$- and $\mathcal{L}$-branch leaf problems. NPC stands for NP-complete, P stands for a polynomial-time algorithm and L stands for a linear-time algorithm. The dashes before the graph classes represent their inclusion relation, i.e., every graph class is included in all classes above with less dashes.

$\mathcal{F}/\mathcal{L}$-branch	GS	BFS	LBFS	DFS	LDFS	MCS & MNS
All Graphs	L / L	NPC / NPC	NPC / NPC	NPC / NPC	NPC / ?	NPC / NPC
- Weakly Chordal	L / L	? / ?	NPC / ?	NPC / NPC	NPC / ?	NPC / NPC
- - Chordal	L / L	L / ?	L / L	NPC / NPC	L / L	L / L
- - - Split	L / L	L / L	L / L	L / NPC	L / L	L / L
- - - Interval	L / L	L / ?	L / L	L / L	L / L	L / L
- Bipartite	L / L	NPC / P	NPC / ?	? / NPC	? / ?	? / ?
- - Chordal Bipartite	L / L	? / P	P / P	? / NPC	? / ?	? / ?

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 $V(G)$ its *vertex set* and by $E(G)$ its *edge set*. 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. We write $G - S$ if we refer to the graph $G[V(G) \setminus S]$. We simplify this notation to $G - v$ if $S = \{v\}$.

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 $N_G^i(v)$ contains all vertices whose distance to the vertex v in G is equal to i. In particular, $N_G^0(v) = \{v\}$ and $N_G^1(v) = N_G(v)$.

A *(linear) ordering* of a set X is a bijection $\sigma\colon \{1,2,\dots,|X|\} \to X$. A *vertex ordering* of a graph G is an ordering of $V(G)$. We will often denote an ordering by a tuple $(x_1,\dots,x_n)$ which means that $\sigma(i) = x_i$. A *prefix* of an ordering $\sigma = (x_1,\dots,x_n)$ is an ordering $(x_1,\dots,x_k)$ with $1 \le k \le n$. We write $\sigma^{-1}(v)$ for the position of element v in σ. Given two elements u and v of a set X, we say that u is *to the left* (resp. *to the right*) of v in an ordering σ of X 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 a set X and ρ of a set 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 $1 \le i \le |X|$ and $\tau(i) = \rho(i - |X|)$ if $|X| < i \le |X \cup Y|$. 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 set of vertices S is *dominating* if any vertex of G is either in S or has a neighbor in S. 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 *weakly chordal* if G contains neither an induced cycle of length ≥ 5 nor the complement of such an induced cycle. The following two lemmas present some operations that do not change the status of a graph of being weakly chordal or not.

Lemma 2.1 (Spinrad and Sritharan [40])**.** *Let G be a graph with a two pair $\{u,w\}$. Then G is weakly chordal if and only if the graph that is constructed by adding the edge uw to G is weakly chordal.*

Lemma 2.2 (Beisegel et al. [3])**.** *Let G be a graph and $v \in V(G)$ such that v is simplicial or adjacent to at least $n(G) - 2$ vertices of G. Then G is weakly chordal if and only if $G - v$ is weakly chordal.*

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 $S(v) := \{w \mid w \prec_\sigma v\}$.[1] A graph G has a PEO if and only if G is chordal [32]. A *split graph* G is a graph whose vertex set can be partitioned into a clique C and an independent set I. An *interval graph* is an intersection graph of intervals on the real line, i.e., each vertex corresponds to an interval and two vertices are adjacent if and only if the two corresponding intervals have a non-empty intersection. A graph is *bipartite* if its vertex set can be partitioned into two independent sets X and Y. A bipartite graph is *chordal bipartite* if it does not contain an induced cycle of length ≥ 6. It is easy to see that any split graph and any interval graph is chordal and any chordal graph is weakly chordal. Furthermore, any chordal bipartite graph is weakly chordal. Note that all these graph classes are *hereditary*, i.e., the induced subgraphs of any graph in one of these classes is also in the class.

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 *descendant* of v if v is an 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.

[1] Note that some authors define PEOs in a reversed way, i.e., the first vertex of a PEO has to be simplicial instead of the last one as in our definition.

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

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

```
1 begin
2     foreach v ∈ V(G) do label(v) ← ∅;
3     for i ← 1 to n(G) do
4         Eligible ← {x ∈ V(G) | x unvisited and ∄ unvisited y ∈ V(G)
5                                   such that label(x) ≺_A label(y)};
6         let v be an arbitrary vertex in Eligible;
7         σ(i) ← v;        /* visits v and assigns the number i to it */
8         foreach unvisited vertex w ∈ N(v) do label(w) ← label(w) ∪ {i};
```

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*. An ordering is called *$\mathcal{A}$-prefix* of G if it is the prefix of some $\mathcal{A}$-ordering of G. We call a search $\mathcal{A}'$ a *subsearch* of another search $\mathcal{A}$ if $\mathcal{A}'(G) \subseteq \mathcal{A}(G)$ holds for all graphs G. Most of the graph searches considered in this paper can be formalized adapting a framework introduced by Corneil et al. [14] (a similar framework is given in [29]). This framework uses subsets of $\mathbb{N}^+$ as vertex labels. Whenever a vertex is visited, its index in the search ordering is added to the labels of its unvisited neighbors. The search $\mathcal{A}$ is defined via a strict partial order $\prec_{\mathcal{A}}$ on the powerset of $\mathbb{N}^+$ (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 [14]). In the definition of any of those partial orders $\prec_{\mathcal{A}}$, the sets A and B will be arbitrary subsets of $\mathbb{N}^+$. For the ease of notation, we define $\min(\emptyset) = \infty$ and $\max(\emptyset) = 0$. 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 visited neighbor can be visited next. We will also use the term *connected search ordering* for orderings that can be produced by GS.

The partial label order $\prec_{BFS}$ for *Breadth First Search* (BFS) is defined as follows: $A \prec_{BFS} B$ if and only if $\min(A) > \min(B)$. For the *Lexicographic Breadth First Search* (LBFS) [33] we consider the partial order $\prec_{LBFS}$ with $A \prec_{LBFS} B$ if and only if $\min(A \setminus B) > \min(B \setminus A)$. 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 $\max(A) < \max(B)$. For the *Lexicographic Depth First Search* [16] we use the strict partial order $\prec_{LDFS}$ where $A \prec_{LDFS} B$ if and only if $\max(A \setminus B) < \max(B \setminus A)$.

The *Maximum Cardinality Search* (MCS) [42] uses the partial order $\prec_{MCS}$ with $A \prec_{MCS} B$ if and only if $|A| < |B|$. The *Maximal Neighborhood Search* (MNS) [16] 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.

Some of the results presented in this paper are not only fulfilled by searches defined above but by a whole family of graph searches, called *strictly comp-monotone graph searches*.

Definition 2.3. A vertex ordering $\sigma = (v_1, \ldots, v_n)$ of a graph G has the *comp-inclusion property* if for any pair $i, j \in \{1, \ldots, n\}$ with $i < j$ such that v_i and v_j are part of the same component of $G[v_i, \ldots, v_n]$ it holds that $N_G(v_i) \cap \{v_1, \ldots, v_{i-1}\}$ is not a strict subset of $N_G(v_j) \cap \{v_1, \ldots, v_{i-1}\}$. A graph search $\mathcal{A}$ has the *comp-inclusion property* if every $\mathcal{A}$-ordering of every graph G has the comp-inclusion property.

A search $\mathcal{A}$ has the *comp-tie property* if for any graph G, any (possibly empty) $\mathcal{A}$-prefix $(v_1, \ldots, v_k)$ of G and any two vertices x, y which are in same component of $G - \{v_1, \ldots, v_k\}$ the following holds: If $N_G(x) \cap \{v_1, \ldots, v_k\} = N_G(y) \cap \{v_1, \ldots, v_k\}$, then $(v_1, \ldots, v_k, x)$ is an $\mathcal{A}$-prefix of G if and only if $(v_1, \ldots, v_k, y)$ is an $\mathcal{A}$-prefix of G.

A graph search $\mathcal{A}$ is called *strictly comp-monotone* if it has both the comp-inclusion property and the comp-tie property.

Shier [39] defined the search *Maximal Element in Component Scheme (MEC)*. It chooses a vertex next whose set of visited neighbors is inclusion maximal among all the vertices in the same component of the graph that is induced by the unvisited vertices. Thus, MEC is the most general graph search having the comp-inclusion property. Shier [39] showed that a vertex ordering of a chordal graph is a PEO if and only if it is a MEC ordering. This implies the following theorem.

Theorem 2.4 (see Shier [39, Theorem 2])**.** *A vertex ordering σ of a chordal graph G is a PEO of G if and only if σ has the comp-inclusion property.*

Thus, graph searches having the comp-inclusion property compute PEOs of chordal graphs. In particular, this holds for all subsearches of MNS, for example, LBFS, LDFS, and MCS. Note that all these searches also have the comp-tie property since all instances of Label Search have this property. This holds because two vertices with the same neighborhood in the visited vertices have the same label and, therefore, they are treated equally.

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 $\sigma = (v_1, \ldots, v_n)$ of G is the $\mathcal{A}^+(\rho)$-ordering of G if σ fulfills the following condition: for any $i \in \{0, \ldots, n-1\}$, the vertex v_{i+1} is the *leftmost* vertex in ρ such that $(v_1, \ldots, v_i, v_{i+1})$ is a prefix of an $\mathcal{A}$-ordering of G.[2]

Searches as BFS and DFS are often used to compute corresponding graph search trees. Beisegel et al. [3] formalized the different concepts of search trees as follows.

Definition 2.5 (Beisegel et al. [3])**.** Let σ be a connected search 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 *first* 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 *last* neighbor w in σ with $w \prec_\sigma v$.

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

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

Lemma 2.6 (Tarjan [41])**.** *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 2.7 (Beisegel et al. [5])**.** *Let T be an $\mathcal{L}$-tree of some DFS ordering of G starting with 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 2.8. Let σ be a connected search ordering of a connected graph G. A vertex $t \in V(G)$ is an *$\mathcal{F}$-leaf ($\mathcal{L}$-leaf)* of σ if t is a leaf in the $\mathcal{F}$-tree ($\mathcal{L}$-tree) of σ. If t 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.

3 Root leaves

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

Observation 3.1. *Let G be a connected graph with $n(G) \geq 2$. The following statements are equivalent for a vertex $t \in V(G)$.*

(1) *Vertex t is the $\mathcal{F}$-root leaf of some GS ordering of G.*

(2) *Vertex t is the $\mathcal{F}$-root leaf of every GS ordering of G starting with t.*

(3) *Vertex t 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 3.2. *Let G be a connected graph with $n(G) \geq 2$. The following statements are equivalent for a vertex $t \in V(G)$.*

(1) *Vertex t is the $\mathcal{L}$-root leaf of some DFS ordering of G.*

(2) *Vertex t is the $\mathcal{L}$-root leaf of every DFS ordering of G starting in t.*

(3) *Vertex t is the $\mathcal{L}$-root leaf of some MCS ordering of G.*

(4) *Vertex t is the $\mathcal{L}$-root leaf of some GS ordering of G.*

(5) *Vertex t is an $\mathcal{L}$-branch leaf of some GS ordering of G.*

(6) *Vertex t is an $\mathcal{F}$-branch leaf of some GS ordering of G.*

(7) *Vertex t is the end-vertex of some GS ordering of G.*

(8) *Vertex t is not a cut vertex of G.*

Proof. First we show the equivalence of all statements concerning GS and statement (8). Charbit et al. [9] showed that a vertex t is an end-vertex of a GS ordering if and only if it is not a cut vertex of G. Thus, (7) $\Leftrightarrow$ (8). Clearly, the end-vertex of a GS ordering is also a branch leaf of its $\mathcal{F}$-tree and its $\mathcal{L}$-tree. Hence, (7) $\Rightarrow$ (5), (6).

Now assume that vertex t is not a cut vertex of G. Then let σ be a GS ordering of $G-t$ that starts in a neighbor w of t. Obviously, $\sigma' = t +\!\!+ \sigma$ is a GS ordering of G. As $G-t$ is connected, every vertex of $G-t$ except from w has a neighbor to its left in σ. Thus, the only neighbor of t in the $\mathcal{L}$-tree of σ' is w and t is $\mathcal{L}$-root leaf of some GS ordering of G. Hence, (8) $\Rightarrow$ (4).

Assume that t is a cut vertex of G and let A and B be two components of $G-t$. If t is the start vertex of the GS ordering σ of G, then the leftmost vertices of A and B in σ are neighbors of t in the $\mathcal{L}$-tree of σ. Therefore, t is not the $\mathcal{L}$-root leaf of σ and (4) $\Rightarrow$ (8).

If the GS ordering σ does not start with the cut vertex t, then w.l.o.g. we may assume that σ starts with a vertex of A. Then the parent of t 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 t in the $\mathcal{F}$-tree and in the $\mathcal{L}$-tree of σ. Therefore, t is neither an $\mathcal{F}$-branch leaf nor an $\mathcal{L}$-branch leaf of a GS ordering of G. Hence, (5) $\Rightarrow$ (8) and (6) $\Rightarrow$ (8).

As DFS orderings and MCS orderings are also GS orderings, the statements (1) and (3) directly imply (4) and, thus, they also imply (8).

To show that (8) implies (3), assume that t is not a cut vertex. Then let ρ be a vertex ordering of G that starts with t and has all neighbors of t to the right of all non-neighbors of t. Let σ be the MCS$^+(\rho)$ ordering of G. Assume for contradiction that t has more than one child in the $\mathcal{L}$-tree T of σ. Let x be the leftmost child of t in σ and let y be another child of t in T. Since T is an $\mathcal{L}$-tree and t is the first vertex of σ, both x and y have only one neighbor to its left in σ, namely t. As t is not a cut vertex, there is an x-y-path P in $G-t$. Let z be the vertex nearest to x on P that is to the right of y in σ. Vertex z exists since t 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. By the choice of z, the neighbor of z on P that is nearer to x on P is to the left of y in σ. If there would be another neighbor of z to the left of y, then the label of z would have been larger than the label of y when y was visited; a contradiction to the fact that σ is an MCS ordering. Hence, vertex z is not a neighbor of t and, therefore, 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, MCS$^+(\rho)$ would have visited z instead of y. Hence, t is an $\mathcal{L}$-root leaf of the MCS ordering σ.

Statement (2) trivially implies (1). It remains to show that (8) implies (2). We show the contraposition. Let σ be a DFS ordering starting with t and let T be the $\mathcal{L}$-tree of σ. Assume t has two children in T. Any path in G between these two children runs through t, due to Lemma 2.6. Thus, t is a cut vertex of G. □

Note that DFS differs from GS and MCS in this result. While for the latter two 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.

Theorem 3.2 directly implies an easy characterization of $\mathcal{L}$-leaves of several searches.

Theorem 3.3. *For any search $\mathcal{A} \in \{$GS, DFS, LDFS, MCS, MNS$\}$ and any vertex t of a connected graph G with $n(G) \geq 2$, the following statements are equivalent.*

(1) *Vertex t is the $\mathcal{L}$-root leaf of some $\mathcal{A}$-ordering of G.*

(2) *Vertex t is an $\mathcal{L}$-leaf of some $\mathcal{A}$-ordering of G.*

(3) *Vertex t is not a cut vertex of G.*

Proof. Statement (1) trivially implies statement (2). As all searches that are possible choices for $\mathcal{A}$ are subsearches of GS, the implication (2) $\Rightarrow$ (3) follows directly from the implications (4) $\Rightarrow$ (8) and (5) $\Rightarrow$ (8) given in Theorem 3.2. Depending on the considered search, the implication (3) $\Rightarrow$ (1) follows from the implications (8) $\Rightarrow$ (2), (3), (4) of Theorem 3.2, respectively. Note that any MCS ordering is an MNS ordering and any LDFS ordering is an DFS ordering. □

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 Section 4 that the recognition of $\mathcal{L}$-*branch* leaves of at least DFS, MCS, and MNS is NP-complete.

The characterization of $\mathcal{L}$-root leaves given in Theorem 3.2 does not work for BFS as the following theorem shows.

Theorem 3.4. *Let G be a connected graph with $n(G) \geq 2$. A vertex $t \in V(G)$ is the $\mathcal{L}$-root leaf of some BFS ordering of G if and only if $G[N_G(t)]$ is connected.*

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

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

As the BFS does not restrict the ordering of the neighbors of the first vertex, the first vertex is not necessarily an $\mathcal{L}$-root leaf of every BFS ordering starting with t even if $G[N_G(t)]$ is connected. However, for LBFS (and other special subsearches of BFS), the connectedness of $G[N_G(t)]$ guarantees that t is an $\mathcal{L}$-root leaf of any search ordering starting with t.

Theorem 3.5. *Let G be a connected graph with $n(G) \geq 2$. Let $\mathcal{A}$ be a subsearch of BFS having the comp-inclusion property. Let $t \in V(G)$ be the start vertex of at least one $\mathcal{A}$-ordering of G. Then the following statements are equivalent.*

(1) *Vertex t is the $\mathcal{L}$-root leaf of some $\mathcal{A}$-ordering of G.*

(2) *Vertex t is the $\mathcal{L}$-root leaf of every $\mathcal{A}$-ordering of G starting with t.*

(3) *The graph $G[N_G(t)]$ is connected.*

Proof. Statement (2) trivially implies statement (1) as, by assumption, there is at least one $\mathcal{A}$-ordering starting with t. Since every $\mathcal{A}$-ordering is a BFS ordering, statement (1) implies (3), due to Theorem 3.4.

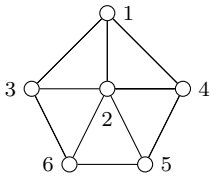


Figure 1: The given vertex ordering is a BFS ordering that has the comp-inclusion property. However, it is not an LBFS ordering as the two vertices with index 5 and 6 would have been swapped in an LBFS.

It remains to show that (3) implies (2). Let $\sigma = (t = w_1, \ldots, w_n)$ be an $\mathcal{A}$-ordering of G. Since $\mathcal{A}$ is a subsearch of BFS, it holds that $N_G(t) = \{w_2, \ldots, w_k\}$ for some $2 \leq k \leq n$. If $k = 2$, then t is a leaf in G and, thus, t is an $\mathcal{L}$-root leaf of σ. Therefore, we may assume that $k > 2$. Let w_i be some vertex with $2 < i \leq k$ and let $S = \{w_1, \ldots, w_{i-1}\}$. Let C be the component of $G - S$ containing w_i. Since $G[N_G(t)]$ is connected, there is a vertex $w_j \in S \setminus \{t\}$ and a vertex $w_\ell \in C$ with $i \leq \ell \leq k$ such that $w_j w_\ell \in E(G)$. Thus, $N_G(w_\ell) \cap S \supseteq \{t, w_j\}$. As $\mathcal{A}$ has the comp-inclusion property, the neighborhood of w_i in S is not a strict subset of the neighborhood of w_ℓ in S. Hence, w_i has at least one neighbor in S that is not t and the edge tw_i is not an element of the $\mathcal{L}$-tree T of σ. Therefore, t has only one child in T, namely w_2. □

Note that LBFS is not the only subsearch of BFS having the comp-inclusion property. Figure 1 presents a BFS ordering that has the comp-inclusion property but is not an LBFS ordering.

4 Complexity of branch leaf recognition

4.1 Branch leaves of DFS

DFS $\mathcal{L}$-trees can be recognized in linear time [23, 26]. As we have seen in Corollary 3.3, 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 4.1. *Let G be a graph with $n(G) \geq 2$. A vertex $t \in V(G)$ is an $\mathcal{L}$-branch leaf of some DFS ordering of G if and only if t is the end-vertex of some DFS ordering of G.*

Proof. If t 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 t 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 unvisited child, then the children that are not ancestors of t are to the left of t in σ. This means that the DFS always visits non-ancestors of t before ancestors of t if this is possible. Due to Lemma 2.7, σ is also a DFS ordering of G.

We claim that t is the end-vertex of σ. Assume for contradiction that $w \neq t$ is the end-vertex of σ. Then, w is a branch leaf of T. Therefore, w is not an ancestor of t. Let x be the common ancestor of t and w that has the largest distance to the root vertex r in T. Let t' be the child of x that is an ancestor of t and let w' be the child of x that is an ancestor

of w (t' could be equal to t and w' could be equal to w). By construction of σ, the vertex w' is to the left of t' in σ. Since the descendants of w' appear consecutively directly after w' in σ, w is to the left of t in σ; a contradiction. □

Charbit et al. [9, Corollary 4.2] 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 4.1, we can replace the term end-vertex in their result by the term $\mathcal{L}$-branch leaf.

Theorem 4.2. *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.*

This result cannot be applied to bipartite graphs as they are not closed under the insertion of universal vertices. However, Gorzny [20, Theorem 4.1.1] showed that the end-vertex problem is also hard on bipartite graphs. It is easy to check that his construction also works for chordal bipartite graphs which leads to the following result.

Theorem 4.3. *Given a chordal bipartite graph G and a vertex $t \in V(G)$, it is* NP*-complete to decide whether t is an $\mathcal{L}$-branch leaf of some DFS ordering of G.*

A result similar to Corollary 4.2 can be given for $\mathcal{F}$-branch leaves of DFS. By adapting the proof given in [35] 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 4.4. *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 $\{v_1, \ldots, v_n\}$. W.l.o.g. we may assume that $n \geq 2$. We construct the graph G' by adding vertices to G as follows: First we add the vertex v_i' for all $i \in \{1, \ldots, n\}$ and connect it to v_i. We call v_i' the *pendant* of v_i. Then we add the vertex t and make it a universal vertex of G'. 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 t 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 t is a branch leaf.

Now assume that t is a branch leaf of the $\mathcal{F}$-tree T of some DFS ordering σ of G'. Assume for contradiction that there is a vertex $v \in V(G)$ to the right of t in σ. As its pendant v' is not a child of t in T, vertex v' is to the left of t in σ. Thus, v' is to the left of both its neighbors in σ and, therefore, v' must be the start vertex of σ. However, this implies that t is the second vertex of σ and, since $n \geq 2$, vertex t has at least one child in T; a contradiction. It follows that all vertices $v \in V(G)$ are to the left of t in σ. Now let T' be the $\mathcal{L}$-tree of σ. Due to Lemma 2.6, all the vertices of G are ancestors of t in T' and, thus, they all lie on a path P from the root of T' to t. If some pendant vertex v' lies on this path, then either v' is the first vertex of P or the successor of v' in σ is vertex t. Thus, only the first and the last vertex of the path P could be one of these pendant vertices. Therefore, P contains a Hamiltonian path of G.

The Hamiltonian path problem is NP-complete on chordal graphs [7]. 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. □

If we compare Corollary 4.2 and Theorem 4.4, 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 5.18).

4.2 Branch leaves of BFS and LBFS

Similar to the results in the last section, we will show here that the recognition of $\mathcal{F}$-branch leaves and $\mathcal{L}$-branch leaves of BFS and LBFS is NP-hard. To this end, we consider the following problem.

Problem 4.5 (Beginning-end-vertex problem of graph search $\mathcal{A}$)**.**

Instance: A connected graph G and two vertices $r, t \in V(G)$.

Question: Is there an $\mathcal{A}$-ordering of G starting with r and ending with t?

This problem is NP-hard for BFS and LBFS even if G is a bipartite graph [9, 21].

The NP-hardness proofs for the recognition of $\mathcal{F}$-branch leaves and $\mathcal{L}$-branch leaves of (L)BFS use similar constructions. We first prove a structural property of these constructions.

Lemma 4.6. *Let G be a bipartite graph and let $r, t \in V(G)$ be two vertices with $dist_G(r,t) = ecc_G(r) = k$. Let G' be the graph that is constructed from G by adding a path $P = (r, x_1, \ldots, x_{k+3}, t)$ to G, where $x_i \notin V(G)$ for all $i \in \{1, \ldots, k+3\}$. Let H be a graph that contains G' as subgraph such that for every vertex x_i it holds that $N_H(x_i) = N_{G'}(x_i)$.*

If vertex t is an $\mathcal{F}$-branch leaf or an $\mathcal{L}$-branch leaf of some BFS ordering σ of H, then the first vertex of σ is an element of the set $X = \{x_1, \ldots, x_{k+3}\}$.

Proof. Let σ be a BFS ordering of H starting with some vertex $s \notin X$. If $dist_H(s, x_{k+3}) < dist_H(s, x_{k+2})$, then x_{k+2} is to the right of x_{k+3} in σ. Thus, the other neighbor of x_{k+3}, namely t, is the parent of x_{k+3} in both the $\mathcal{F}$-tree and the $\mathcal{L}$-tree of σ and, therefore, t is not an $\mathcal{F}$-branch leaf or $\mathcal{L}$-branch leaf of σ.

Thus, we may assume that $dist_H(s, x_{k+3}) \geq dist_H(s, x_{k+2})$. This implies that no shortest paths from s to x_{k+2} in H uses x_{k+3}. By the condition on the neighborhoods of the vertices of X in H, all other paths from s to x_{k+2} run through r. Hence, it holds that $dist_H(s, x_{k+2}) = dist_H(s, r) + k + 2$. However, it holds that $dist_H(s, x_{k+3}) \leq dist_H(s, r) + dist_H(r, t) + 1 \leq dist_H(s, r) + k + 1$; a contradiction to the assumption that $dist_H(s, x_{k+3}) \geq dist_H(s, x_{k+2})$. □

We now prove that recognizing $\mathcal{F}$-branch leaves of BFS is NP-hard even on bipartite graphs.

Theorem 4.7. *It is NP-complete to decide whether a vertex of a bipartite graph G is an $\mathcal{F}$-branch leaf of some BFS ordering or some LBFS ordering of G.*

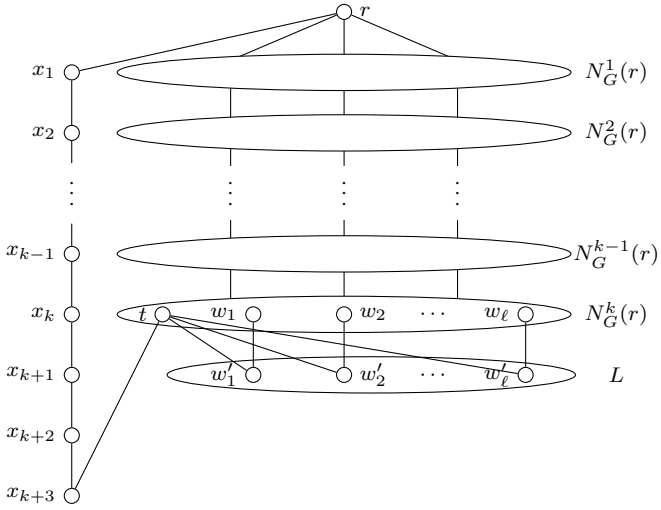


Figure 2: A graphical representation of the graph H constructed in the proof of Theorem 4.7.

Proof. We reduce the beginning-end-vertex problem of (L)BFS on bipartite graphs to the respective $\mathcal{F}$-branch leaf recognition problem. Note again that both problems are NP-hard [9, 21].

Let G and $r, t \in V(G)$ be an instance of this problem. Let k be the eccentricity of r in G. W.l.o.g. we may assume that t is in $N_G^k(r)$ since, otherwise, t is obviously not the end-vertex of any BFS ordering of G starting with r. We construct the graph H from G as follows (see Figure 2). We add a path $P = (r, x_1, \ldots, x_{k+3}, t)$ to G. Furthermore, for every vertex $w_i \in N_G^k(r) \setminus \{t\}$ we add a vertex w_i' and the edges tw_i' and $w_i w_i'$ to G. We collect the vertices w_i' in the set L. Note that H is also bipartite.

First assume that t is the end-vertex of some (L)BFS ordering σ of G starting with r. Let $\rho = (x_1, x_2, \ldots, x_{k+3}) +\!\!+ \sigma$ and let τ be the (L)BFS$^+(\rho)$ ordering of H. 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 $v \in N_G^j(r)$ 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 t 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 t in τ since $dist_H(x_1, x_{k+3}) = k+2$ and $dist_H(x_1, t) = k+1$. For the same reason, all the vertices $w_i' \in L$ are also to the right of any vertex of $V(G)$ in τ. Thus, none of the vertices in $V(H) \setminus V(G)$ had an influence on the ordering of the vertices of $V(G)$ in τ. Therefore, the restriction of τ to the vertices of the graph G is exactly σ and every $w_i \in N_G^k(r) \setminus \{t\}$ is to the left of t in τ. Hence, t is an $\mathcal{F}$-branch leaf of τ.

Now assume that t is an $\mathcal{F}$-branch leaf of some (L)BFS ordering σ of H. Due to Lemma 4.6, σ starts with a vertex x_i. For any vertex $w_j' \in L$, it holds $dist_H(x_i, w_j') > dist_H(x_i, t)$. Thus, all elements of L are to the right of t in σ. Therefore, all the vertices in $N_G^k(r)$ are to the left of t in σ since t 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 t in σ, then $dist_H(x_i, y) \geq dist_H(x_i, t)$. Thus, the shortest path from x_i to t runs through x_{k+3} which implies that $dist_H(x_i, x_{k+3}) < dist(x_i, z)$ for all $z \in N_G^{k-1}(r)$. However, as x_{k+3} is adjacent to t in G but not adjacent to any other

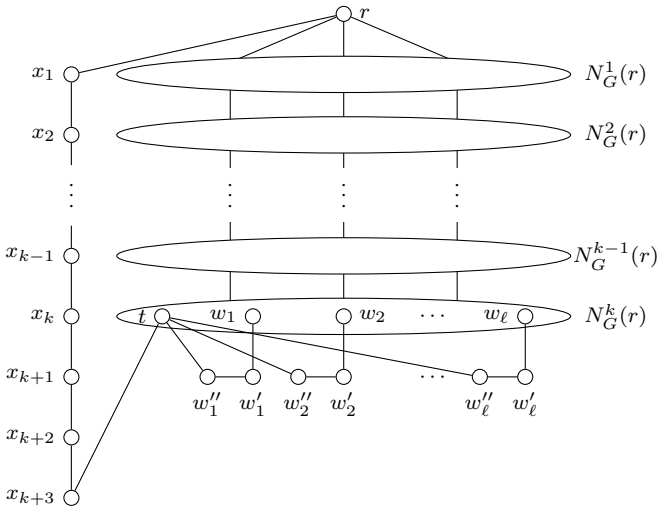


Figure 3: A graphical representation of the graph H constructed in the proof of Theorem 4.8.

element of $N_G^k(r)$, vertex t is the leftmost vertex of $N_G^k(r)$ in σ; a contradiction to the observation above.

Hence, t 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 t. None of the vertices in $V(H) \setminus V(G)$ had an influence on the order of the vertices of G in σ. Thus, σ^* is an (L)BFS ordering of G starting with r and ending with t. □

With a similar construction we can prove that recognizing $\mathcal{L}$-branch leaves of BFS is also NP-hard. Note that we prove this result for arbitrary graphs and not for bipartite graphs.

Theorem 4.8. *It is* NP*-complete to decide whether a vertex of a graph G is an $\mathcal{L}$-branch leaf of some BFS ordering or some LBFS ordering of G.*

Proof. We use a similar construction as for Theorem 4.7. Again, let G, r, t be an instance of the beginning-end-vertex problem of (L)BFS where G is a connected bipartite graph. We construct the graph H from G as follows (see Figure 3). We add a path $P = (r, x_1, \ldots, x_{k+3}, t)$ to G. Furthermore, for every vertex $w_i \in N_G^k(r) \setminus \{t\}$ we add a vertex w_i' and a vertex w_i'' as well as the edges tw_i'', $w_i'w_i''$ and ww_i' to G.

First assume that t is the end-vertex of some (L)BFS ordering σ of G starting with r. Let $\rho = (x_1) +\!\!+ \sigma +\!\!+ (x_2, \ldots, x_{k+3})$ and let τ be the (L)BFS$^+(\rho)$ ordering of H. 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 $v \in N_G^j(r)$ is in layer $j+1$. The choice of ρ also implies that every vertex x_i is the last vertex of its layer in τ. In particular, vertex x_{k+2} is to the right of t in τ and, thus, x_{k+2} is the parent of x_{k+3} in the $\mathcal{L}$-tree of τ. Furthermore, x_{k+3} is to the right of t in τ since $dist_H(x_1, x_{k+3}) = k+2$ and $dist_H(x_1, t) = k+1$. For the same reason, all the vertices w_i' and w_i'' are also to the right of any vertex of $V(G)$ in τ. As t is the last vertex

in $N_G^k(r)$, the vertex w_i' is to the left of vertex w_i'' for any i. Hence, w_i' is the parent of w_i'' in the $\mathcal{L}$-tree of τ and t is a branch leaf in this $\mathcal{L}$-tree.

Now assume that t is an $\mathcal{L}$-branch leaf of the (L)BFS ordering σ of H. Due to Lemma 4.6, the first vertex of σ is some vertex x_i. We claim that t is to the right of every vertex $w_j \in N_G^k(r) \setminus \{t\}$. Assume for contradiction that $t \prec_\sigma w_j$ for some $w_j \in N_G^k(r)$. Obviously, $dist_H(x_i, t) < dist_H(x_i, w_j'')$. Thus, $t \prec_\sigma w_j''$. Furthermore, $w_j'' \prec_\sigma w_j'$ since $t \prec_\sigma w_j$. This implies that w_j'' is a child of t in the $\mathcal{L}$-tree of σ; a contradiction. Therefore, we know that t is the rightmost vertex of $N_G^k(r)$ in σ. This also implies that the vertex x_{k+3} had no influence on the ordering of N_G^k in σ. Hence, the restriction of σ to the vertices of G forms an (L)BFS ordering of G starting with r and ending with t. □

While Theorem 4.7 shows NP-hardness of $\mathcal{F}$-branch leaf recognition even for bipartite graphs, Theorem 4.8 only states NP-hardness for general graphs. It is easy to check that the graph H constructed in the proof of Theorem 4.8 is not bipartite. In fact, the $\mathcal{L}$-branch leaf recognition of BFS is not NP-hard on bipartite graphs (unless P = NP) since there is a simple characterization of BFS $\mathcal{L}$-branch leaves of bipartite graphs.

Theorem 4.9. *Let G be a connected bipartite graph with $n(G) \geq 2$. A vertex $t \in V(G)$ is an $\mathcal{L}$-branch leaf of some BFS ordering of G if and only if there is a vertex $r \in V(G) \setminus \{t\}$ such that $dist_G(r, w) = dist_{G-t}(r, w)$ for all $w \in V(G) \setminus \{t\}$.*

Proof. Assume that there is a vertex $r \in V(G) \setminus \{t\}$ such that $dist_G(r, w) = dist_{G-t}(r, w)$ for all $w \in V(G) \setminus \{t\}$. Let $P = (r = w_0, \ldots, w_k = t)$ be a shortest path from r to t, i.e., t has distance k to r. Let ρ be a vertex ordering of G starting with P. It is easy to see that in the BFS$^+(\rho)$ ordering σ of G 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 t. Therefore, x has a neighbor y in the k-th layer that is not t. Since $t \prec_\sigma y \prec_\sigma x$, vertex t is not the parent of x in T. Since G is bipartite, the layers of σ are independent sets and, thus, t is also not the parent of any vertex in the k-th layer. Hence, t is a branch leaf of T.

Now assume that t is a branch leaf of the $\mathcal{L}$-tree T of a BFS ordering σ. Let r be the start vertex of σ. Let w be a vertex different from t 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, the parent of every vertex is in its preceding layer. Thus, P has $dist_G(r, w)$ edges. Since t is a leaf of T, P does not contain t. Therefore, P is also contained in $G - t$ and $dist_{G-t}(r, w) = dist_G(r, w)$. □

To check whether the condition of Theorem 4.9 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 4.10. *Given a connected bipartite graph G and a vertex $t \in V(G)$, we can decide in time $\mathcal{O}(n(G) \cdot m(G))$ whether t is the $\mathcal{L}$-branch leaf of some BFS ordering of G.*

The results of Theorem 4.7 and Corollary 4.10 are quite surprising since the $\mathcal{L}$-tree recognition problem of BFS is NP-hard on bipartite graphs [35] while the $\mathcal{F}$-tree recognition problem of BFS can be solved in linear time even on general graphs [24, 30].

We have to leave open the complexity of he $\mathcal{L}$-branch leaf recognition problem of LBFS on bipartite graphs. However, we can show that the problem is polynomial-time solvable if we consider chordal bipartite graphs. To this end, we use the following two lemmas.

Lemma 4.11 (Scheffler [34])**.** *Let G be a connected chordal bipartite graph and let r be a vertex of 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 $N_G(x) \cap N_G^{i-1}(r) \subseteq N_G(y)$ or $N_G(y) \cap N_G^{i-1}(r) \subseteq N_G(x)$.*

Lemma 4.12. *Let G be a graph and ρ be a vertex ordering of G. Let σ be the LBFS$^+(\rho)$ ordering of G. If $x \prec_\sigma y$ but $y \prec_\rho x$, then there is a vertex z with $z \prec_\sigma x$ and $xz \in E(G)$ and $yz \notin E(G)$.*

Proof. Since y is to the left of x in ρ, the label of y was strictly smaller than the label of x when x was visited by the LBFS$^+$. Thus, the label of x contains a number that is not contained in the label of y. This implies the existence of a vertex z with $z \prec_\sigma x$, $xz \in E(G)$ and $yz \notin E(G)$. □

Theorem 4.13. *Let G be a connected chordal bipartite graph G with $n(G) \geq 2$ and let $r, t \in V(G)$ be two vertices of G with $r \neq t$. Let $k = dist_G(r,t)$. There is an LBFS ordering of G starting with r that has $\mathcal{L}$-branch leaf t if and only if for all vertices $v \in N_G(t) \cap N_G^{k+1}(r)$ there is a vertex $w \in N_G(v) \cap N_G^k(r)$ such that $N_G(w) \cap N_G^{k-1}(r) \subseteq N_G(t)$.*

Proof. First assume that the condition on the neighbors of t is fulfilled. Then let ρ be a vertex ordering of G where r is the first vertex and t is second vertex. Let σ be the LBFS$^+(\rho)$ ordering of G. Note that σ starts with r. We claim that t is an $\mathcal{L}$-leaf of σ. Assume for contradiction that this is not the case and let v be a child of t in the $\mathcal{L}$-tree T of σ. Due to the condition on the neighbors of t, there is a vertex $w \in N_G(v) \cap N_G^k(r)$ such that $N_G(w) \cap N_G^{k-1}(r) \subseteq N_G(t)$. It must hold that $w \prec_\sigma t$ since, otherwise, t would not be the parent of v in T. However, $t \prec_\rho w$. By Lemma 4.12, there is a vertex z with $z \prec_\sigma w$, $zw \in E(G)$ and $zt \notin E(G)$. As G is bipartite and σ is a BFS ordering, it follows that $z \in N_G^{k-1}(r)$; a contradiction to the fact that $N_G(w) \cap N_G^{k-1}(r) \subseteq N_G(t)$.

Now assume that t is a branch leaf of the $\mathcal{L}$-tree T of some LBFS ordering σ of G starting with r. Let v be an arbitrary vertex in $N_G(t) \cap N_G^{k+1}(r)$. Let w be the parent of v in T. It holds that $w \in N_G^k(r)$. Furthermore, it must hold that $t \prec_\sigma w$. Since v is adjacent to both t and w, Lemma 4.11 implies that $N_G(w) \cap N_G^{k-1}(r) \subseteq N_G(t)$ or $N_G(t) \cap N_G^{k-1}(r) \subseteq N_G(w)$. Thus, if $N_G(w) \cap N_G^{k-1}(r) \not\subseteq N_G(t)$, then $N_G(t) \cap N_G^{k-1}(r) \subsetneq N_G(w)$. However, this contradicts that fact that $t \prec_\sigma w$ since LBFS fulfills the comp-inclusion property. Therefore, $N_G(w) \cap N_G^{k-1}(r) \subseteq N_G(t)$. □

A similar result can be given for $\mathcal{F}$-branch leaves.

Theorem 4.14. *Let G be connected chordal bipartite graph G with $n(G) \geq 2$ and let $r, t \in V(G)$ be two vertices of G. Let $k = dist_G(r,t)$. There is an LBFS ordering of G starting with r that has $\mathcal{F}$-branch leaf t if and only if for all vertices $v \in N_G(t) \cap N_G^{k+1}(r)$ there is a vertex $w \in N_G(v) \cap N_G^k(r)$ such that $N_G(t) \cap N_G^{k-1}(r) \subseteq N_G(w)$.*

Proof. The theorem can be proven in the same way as Theorem 4.13. We only have to put t to the end of ρ and swap the positions of t and w in σ. The details are left to the reader. □

The conditions given in Theorems 4.13 and4.14 can be checked in time $\mathcal{O}(n(G) + m(G))$ for any connected chordal bipartite graph G and fixed vertices r and t. Thus, the overall time to recognize $\mathcal{L}$-branch and $\mathcal{F}$-branch leaves is $\mathcal{O}(n(G) \cdot m(G))$.

Corollary 4.15. *Given a connected chordal bipartite graph G and a vertex $t \in V(G)$, we can decide in time $\mathcal{O}(n(G) \cdot m(G))$ whether t is an $\mathcal{F}$-branch leaf or an $\mathcal{L}$-branch leaf of some LBFS ordering of G.*

4.3 Branch leaves of strictly comp-monotone searches

Recognizing $\mathcal{F}$-branch leaves is NP-hard for any strictly comp-monotone graph search.

Theorem 4.16. *Let $\mathcal{A}$ be a strictly comp-monotone graph search. 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. [3]. 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 $X = \{x_1, \dots, x_k, \overline{x}_1, \dots, \overline{x}_k\}$ 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 $\overline{x}_i$ 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 visit all literal vertices of literals that are set to true in $\mathcal{B}$ and then we visit t. These vertices form a clique. Furthermore, the subgraph induced by the unvisited vertices stays connected until t is visited. Therefore, this ordering is a prefix of an $\mathcal{A}$-ordering, due to the comp-inclusion property and the comp-tie property. We visit 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 an $\mathcal{A}$-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 $x_i\overline{x}_i$ are missing in $G(\mathcal{I})$, 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 $\{x_i, \overline{x}_i\}$ forms a two-pair in $G(\mathcal{I})$. Due to Lemma 2.1, we may add all the edges between these pairs. The resulting graph is a split graphs with clique $X \cup \{t\}$ and independent set C. Hence, $G(\mathcal{I})$ is weakly chordal. □

Unfortunately, we were not able to find a similar general proof of NP-hardness for the problem of recognizing $\mathcal{L}$-branch leaves. However, we can show that the problem is hard at least for MCS and MNS.

Theorem 4.17. *It is NP-complete to decide whether a vertex of a weakly chordal graph G is an $\mathcal{L}$-branch leaf of some MCS ordering or some MNS ordering.*

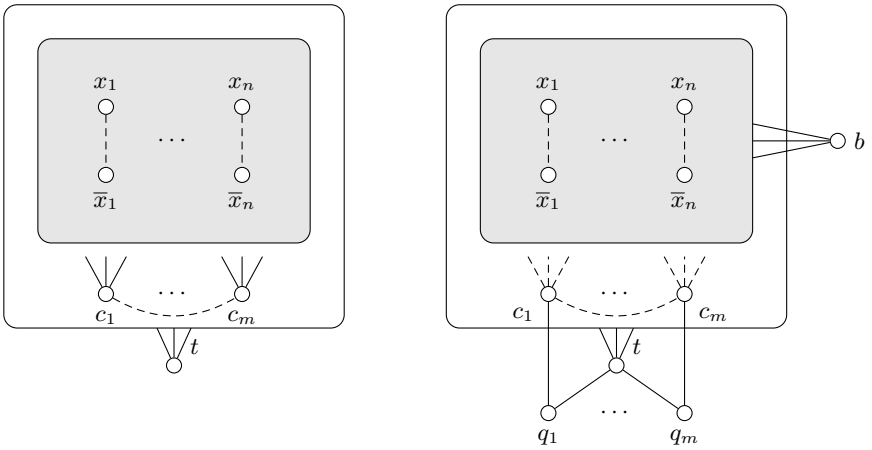


Figure 4: The construction of the proofs of Theorem 4.16 (left) and Theorem 4.17 (right). In the gray boxes, only non-edges are displayed by dashed lines. On the left side, the clause vertices have exactly three neighbors in the gray box, on the right side the clause vertices have exactly three non-neighbors in the gray box. The connection of a vertex with a box means that the vertex is adjacent to all vertices in the box.

Proof. Again we reduce 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 four variables, i.e., there are at least eight different literals. We construct the corresponding graph $G(\mathcal{I})$ as follows (see Figure 4). Let $X = \{x_1, \ldots, x_k, \overline{x}_1, \ldots, \overline{x}_k\}$ 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 $\overline{x}_i$ for every $i \in \{1, \ldots, k\}$. Let $C = \{c_1, \ldots, c_\ell\}$ 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 *not* contained in the clause associated with c_i. Additionally, we add a vertex b that is adjacent to all vertices in X. For every vertex c_i, we add a vertex q_i that is adjacent to c_i. Finally, we add a vertex t that is adjacent to every other vertex except from b.

First assume that there is a fulfilling assignment $\mathcal{B}$ of $\mathcal{I}$. We define an MCS ordering σ of $G(\mathcal{I})$ with $\mathcal{L}$-branch leaf t as follows (note that this ordering is also an MNS ordering). We start with the literal vertices that correspond to the assignment $\mathcal{B}$. Then we visit t. Since $\mathcal{B}$ is fulfilling, every clause vertex and every literal vertex has some vertex to the left of t that is not its neighbor. Thus, b has the maximal number of visited neighbors and we can visit b next. Now all remaining literal vertices have the maximal number of visited neighbors and can be visited in any order. Finally, we visit the clause vertices and then the vertices q_i. We observe that every vertex to the right of t that is adjacent to t has another neighbor that is to the right of t. The literal vertices have b, the clause vertices have some literal vertex (as there is at least one variable that appears neither negated nor non-negated in the respective clause) and every vertex q_i has vertex c_i. Hence, t is an $\mathcal{L}$-branch leaf of σ.

Now assume that t is an $\mathcal{L}$-branch leaf of some MNS ordering σ of $G(\mathcal{I})$. It is an easy observation that t cannot be the end-vertex of σ since at most one vertex q_i can be to the

left of t. Therefore, there is a vertex z that is the direct successor of t in σ. Vertex z is not a neighbor of t in $G(\mathcal{I})$ as, otherwise, z would be a child of t in the $\mathcal{L}$-tree of σ. This implies that vertex z is b. In particular, this is also implies that all vertices to the left of t in σ are neighbors of t.

Next we show that none of the vertices c_i or q_i are to the left of t in σ. Assume for contradiction that there is one of these vertices to the left of t. If vertex c_i is to the left of t, then q_i is also to the left of t since, otherwise, q_i would be a child of t in the $\mathcal{L}$-tree of σ. Thus, we may assume that there is a vertex q_i to the left of t. The only vertex that could be to the left of q_i in σ is c_i because, otherwise, q_i would never be visited before t, due to the comp-inclusion property of σ. For the same reason, after c_i and q_i vertex t has to be visited in σ. However, then b could not be the direct successor of t in σ as none of the neighbors of b have been visited at this point.

Summarizing, all vertices to the left of t are literal vertices. We observe that at most one literal vertex of every variable can be to the left of t, due to the comp-inclusion property. As we have observed above, vertex b is the direct successor of t in σ. Vertex b is not adjacent to t but all clause vertices are adjacent to t. Thus, for all clause vertices c_i there must be a literal vertex to the left of t that is not adjacent to c_i. By construction, the corresponding literal is an element of the clause C_i. Therefore, we can define a fulfilling assignment of $\mathcal{I}$ in the following way: The variable x_i is set to true if and only if the vertex x_i is to the left of t in σ.

To show that $G(\mathcal{I})$ is weakly chordal, we use Lemmas 2.1 and 2.2 to modify $G(\mathcal{I})$ without changing the status of $G(\mathcal{I})$ of being weakly chordal or not. All q_i are simplicial and can be removed from $G(\mathcal{I})$, due to Lemma 2.2. Now we claim that all the pairs $\{x_i, \overline{x}_i\}$ are two-pairs. It is clear that no induced path between x_i and $\overline{x}_i$ of length ≥ 3 uses t, b, or any literal vertex as inner vertex because all these vertices are adjacent both to x_i and $\overline{x}_i$. However, this implies that all inner vertices of such a path are clause vertices; a contradiction as they form an independent set. Therefore, we can add all the edges $\{x_i, \overline{x}_i\}$ to the graph, due to Lemma 2.1. The resulting graph is a split graph with clique $X \cup \{t\}$ and independent set $C \cup \{b\}$. Therefore, $G(\mathcal{I})$ is weakly chordal. □

Summarizing the results given so far, the only case of a concrete search whose complexity on general graphs is still open is the recognition of $\mathcal{L}$-branch leaves of LDFS orderings.

5 Branch leaves and chordal graphs

Chordal graphs are strongly related to graph searches. The searches LBFS and MCS were initially introduced to recognize chordal graphs and can also be used to recognize subclasses of them such as (unit) interval graphs [11, 19]. For this reason, graph searching problems as for example the end-vertex problem or the search tree recognition problems were extensively studied on chordal graphs (see, e.g., [2, 3, 31]). In this section, we will extend these results also to the branch leave recognition problems.

5.1 Branch leaves of strictly comp-monotone searches

As we have seen in Theorem 2.4, strictly comp-monotone graph searches compute PEOs of chordal graphs. Thus, any $\mathcal{F}$-tree or $\mathcal{L}$-tree of such an ordering is also an $\mathcal{F}$-tree or $\mathcal{L}$-tree of some PEO. Beisegel et al. [3] showed that this also holds the other way around, 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.[3] This also implies that the $\mathcal{F}$-branch leaves and $\mathcal{L}$-branch leaves of these searches correspond to those of PEOs.

Proposition 5.1. *Given a chordal graph G and a strictly comp-monotone graph search $\mathcal{A}$, a vertex $t \in V(G)$ is an $\mathcal{F}$-branch leaf ($\mathcal{L}$-branch leaf) of some $\mathcal{A}$-ordering of G if and only if t is an $\mathcal{F}$-branch leaf ($\mathcal{L}$-branch leaf) of some PEO of G.*

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

Theorem 5.2. *Let G be a connected chordal graph with $n(G) \geq 2$ and let $\mathcal{A}$ be a strictly comp-monotone graph search. A vertex $t \in V(G)$ is an $\mathcal{L}$-branch leaf of some $\mathcal{A}$-ordering of G if and only if t is simplicial.*

Proof. If t is a simplicial vertex, then there is a PEO σ that ends with t. Obviously, t is an $\mathcal{L}$-branch leaf of σ. Due to Proposition 5.1, t is also an $\mathcal{L}$-branch leaf of some $\mathcal{A}$-ordering.

For the other direction, let σ be an $\mathcal{A}$-ordering of G and let t be a non-simplicial vertex of G. By Theorem 2.4, σ is a PEO of G and, therefore, not all neighbors of t are to the left of t in σ. Let w be the leftmost neighbor of t in σ that is to the right of t in σ. Let x be the parent of w in the $\mathcal{L}$-tree T of σ. If x is not equal t, then it holds $t \prec_\sigma x \prec_\sigma w$. As σ is a PEO and $tw, xw \in E(G)$, the edge tx is also in $E(G)$; a contradiction to the choice of w. Hence, t is the parent of w in T and t is not an $\mathcal{L}$-branch leaf of σ. □

Since we can decide in linear time whether a vertex is simplicial [2, Lemma 12], we can recognize $\mathcal{L}$-branch leaves of strictly comp-monotone graph searches in linear time.

Corollary 5.3. *Given a connected chordal graph G and a vertex $t \in V(G)$, we can decide in time $\mathcal{O}(n(G) + m(G))$ whether t is the $\mathcal{L}$-branch leaf of a search ordering of some strictly comp-monotone graph search.*

With the same argument as in the proof of Theorem 5.2 we can see that simplicial vertices are also $\mathcal{F}$-branch leaves of search orderings of strictly comp-monotone graph searches. However, there are further $\mathcal{F}$-branch leaves.

Theorem 5.4. *Let G be a connected chordal graph with $n(G) \geq 2$ and let $\mathcal{A}$ be a strictly comp-monotone graph search. A vertex $t \in V(G)$ is an $\mathcal{F}$-branch leaf of some $\mathcal{A}$-ordering of G if and only if the graph $G[N_G(t)]$ has a dominating clique.*

Proof. First assume that $G[N_G(t)]$ has a dominating clique C. It is obvious that there is an LBFS ordering σ of G that starts with the vertices of C.[4] Since all neighbors of t have a neighbor in C or are elements of C, t is an $\mathcal{F}$-branch leaf of σ. Due to Theorem 2.4, the ordering σ is a PEO and by Proposition 5.1 vertex t is also an $\mathcal{F}$-branch leaf of $\mathcal{A}$.

Now let t be an $\mathcal{F}$-branch leaf of the $\mathcal{A}$-ordering σ of G. Due to Theorem 2.4, the ordering σ is a PEO. Let S be the set of neighbors of t that are to the left of t in σ. The set S induces a clique of G. Thus, if $S = N_G(t)$, then $G[N_G(t)]$ is a clique and we are done. Hence, we may assume that there is a vertex $w \in N_G(t) \setminus S$. As w is not a child of t in the $\mathcal{F}$-tree of σ, there is a vertex $x \in N_G(w)$ with $x \prec_\sigma t$. Since σ is a PEO, vertex x is a neighbor of t and, thus, $x \in S$. Therefore, any neighbor of t that is not in S has a neighbor in S and, hence, S induces a dominating clique of $G[N_G(t)]$. □

[3] Beisegel et al. [3] showed this for a family of searches that they call *edge-forced PEO-finders*. On chordal graphs, this family of searches is equivalent to strictly-comp monotone searches (see [37, Proposition 0.2.35]).

[4] Note that this does not necessarily hold for any strictly comp-monotone graph search.

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. [27] gave the following characterization of these graphs.

Theorem 5.5 (Kratsch et al. [27])**.** *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,[5] 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. [13].

Theorem 5.6 (Corneil et al. [13])**.** *Let G be a chordal graph and let $t \in V(G)$ be the end-vertex of some LBFS ordering of G. If $ecc_G(t) < diam(G)$, then $ecc_G(t)$ is even and $ecc_G(t) = diam(G) - 1$.*

Theorem 5.6 implies that we can decide for any odd k in linear time whether a chordal graph's diameter is at most k.

Theorem 5.7. *Given a chordal graph G and an odd natural number k, we can decide in time $\mathcal{O}(n(G) + m(G))$ whether the diameter of G is at most k.*

Proof. We compute an LBFS ordering σ of G in linear time (see [33]). Let t be the end-vertex of σ. We can compute the eccentricity of t in linear time by simply compute a BFS ordering starting with t. If $ecc(t) > k$, then the diameter of G is larger than k. If $ecc(t) = k$, then, by Theorem 5.6, $diam(G) = k$. If $ecc(t) < k$, then $diam(G) \leq ecc(t) + 1 \leq k$, due to Theorem 5.6. □

Combining Theorems 5.5 and 5.7 gives a linear-time algorithm that decides whether a chordal graph has a dominating clique.

Corollary 5.8. *Given a chordal graph G, we can decide in time $\mathcal{O}(n(G)+m(G))$ whether G has a dominating clique.*

As chordal graphs are hereditary, the neighborhood of some vertex in a chordal graph G induces a chordal subgraph of G. Hence, the combination of Theorem 5.4 and Corollary 5.8 leads to a linear-time algorithm that decides whether a vertex of a chordal graph can be an $\mathcal{F}$-branch leave of some strictly comp-monotone graph search's ordering.

Corollary 5.9. *Given a chordal graph G, a vertex $t \in V(G)$ and a strictly comp-monotone graph search $\mathcal{A}$, we can decide in time $\mathcal{O}(n(G) + m(G))$ whether t is an $\mathcal{F}$-branch leaf of some $\mathcal{A}$-ordering of G.*

5.2 Branch leaves of BFS

The condition given in Theorem 5.4 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 Figure 5. To characterize BFS $\mathcal{F}$-branch leaves of chordal graphs, we start with the following two lemmas.

[5] Even on split graphs, the diameter cannot be computed in subquadratic time unless the Strong Exponential Time Hypothesis fails [8].

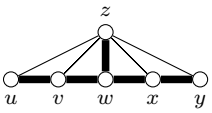


Figure 5: The given graph G is chordal. There is no dominating clique in the graph $G[N_G(z)]$. However, the given spanning tree is the $\mathcal{F}$-tree of the BFS ordering (w, v, x, z, u, y) and, thus, z is a $\mathcal{F}$-branch leaf of BFS.

Lemma 5.10. *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. □

Lemma 5.11. *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, $N_G(y) \cap N_G^{i-1}(r) \subseteq N_G(x)$. □

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

Lemma 5.12. *Let G be a connected chordal graph and let $v \in V(G)$ such that $G - v$ is connected. 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 $N_G(v)$. 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. □

Using Lemmas 5.10, 5.11, and 5.12, we can characterize BFS $\mathcal{F}$-branch leaves of chordal graphs.

Theorem 5.13. *Let G be a connected chordal graph with $n(G) \geq 2$. A vertex $t \in V(G)$ is an $\mathcal{F}$-branch leaf of some BFS ordering of G if and only if the radius of $G[N_G(t)]$ is at most two.*

Proof. First assume that $G[N_G(t)]$ has radius at most two and let w be a central vertex of $G[N_G(t)]$, i.e., w has distance at most two to any vertex in $G[N_G(t)]$. There is a BFS ordering σ that starts with w followed by all neighbors of w except from t. Vertex t is an $\mathcal{F}$-branch leaf of σ since all neighbors of t have some neighbor in $N_G[w] \setminus \{t\}$ or are equal to w.



Figure 6: Two cases of the proof of Theorem 5.13. 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 5.10 or Lemma 5.11.

Now assume that $t \in N_G^i(r)$ 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 t' be the parent of t in T. Since T is an BFS $\mathcal{F}$-tree rooted in r, it holds that $t' \in N_G^{i-1}(r)$.

We claim that in $G - t$ vertex t' has distance at most two to every element of $N_G(t)$. Let $w \in N_G(t) \setminus \{t'\}$. If $t'w \in E(G)$, then t' and w have distance one in $G - t$. Therefore, we may assume in the following that $t'w \notin E(G)$. Then Lemma 5.10 implies that $w \notin N_G^{i-1}(r)$. Furthermore, the parent of w in T, say w', is different from t and t'. If $t'w' \in E(G)$, then t' and w have distance two in $G - t$ via the path (t', w', w). Thus, we may also assume that $t'w' \notin E(G)$.

First assume that $w \in N_G^i(r)$ (see left part of Figure 6). Then $w' \in N_G^{i-1}(r)$. Lemma 5.11 implies that w' is adjacent to t because tw, tt', and ww' are in $E(G)$ but $t'w \notin E(G)$. Now the non-existence of $t'w'$ contradicts Lemma 5.10.

Next assume that $w \in N_G^{i+1}(r)$ (see right part of Figure 6). Then $w' \in N_G^i(r)$ and, due to Lemma 5.10, $tw' \in E(G)$. Since $t'w' \notin E(G)$, the parent of w', say w'', is different from t'. Since w' is the parent of w, it holds that $w' \prec_\sigma t$. This implies that $w'' \prec_\sigma t'$. Therefore, w'' is not adjacent to t since, otherwise, t' would not be the parent of t. The non-existence of both $t'w'$ and tw'' contradicts Lemma 5.11.

Summarizing, t' has distance at most two in $G - t$ to any neighbor of t. Since t is an $\mathcal{F}$-branch leaf of some BFS ordering, $G - t$ is connected, due to Theorem 3.2. By Lemma 5.12, t' has distance at most two to any vertex in $G[N_G(t)]$ and, thus, $G[N_G(t)]$ has radius at most two. □

Chepoi and Dragan [10] 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 $G[N_G(t)]$ in linear time. Thus, Theorem 5.13 implies a linear-time algorithm for the BFS $\mathcal{F}$-branch leaf recognition on chordal graphs.

Corollary 5.14. *Given a connected chordal graph G and a vertex $t \in V(G)$, we can decide in time $\mathcal{O}(n(G) + m(G))$ whether t is an $\mathcal{F}$-branch leaf of some BFS ordering of G.*

Unfortunately, we were not able to resolve the BFS $\mathcal{L}$-branch leaf recognition problem on chordal graphs. However, we can characterize the BFS $\mathcal{L}$-branch leaves of split graphs.

Theorem 5.15. *Let G be a connected split graph with $n(G) \geq 2$. A vertex t of G is an $\mathcal{L}$-branch leaf of some BFS ordering of G if and only if one of the following two conditions holds:*

(1) *there is a vertex $w \in V(G) \setminus \{t\}$ with $N_G(t) \subseteq N_G[w]$*

(2) *vertex t is not a cut vertex of G and there is a vertex $w \in V(G) \setminus N_G[t]$ with $d_G(w) \geq 2$.*

Proof. Let (C, I) be a partition of $V(G)$ into a clique C and an independent set I First assume that Condition (1) is fulfilled. Then let σ be a BFS ordering starting with w followed by all vertices in $N_G(w) \setminus \{t\}$. Obviously, t is an $\mathcal{L}$-branch leave of σ since all neighbors of t are to the left of t in σ.

In the following we assume that (1) is not fulfilled but Condition (2) holds. This implies that vertex t is an element of C as, otherwise, Condition (1) would trivially hold true for t. Thus, the vertex w of Condition (2) is an element of I. Let x and y be two neighbors of w in G. Both x and y are in C. We construct the BFS ordering σ as follows. First we visit vertex x, then all elements of $(N_G(x) \cap N_G(t) \cap I) \setminus \{w\}$ and then t. Now we visit w and then y. Afterwards we visit all the remaining vertices of C. The rest of σ follows an arbitrary BFS ordering. We claim that t is an $\mathcal{L}$-branch leaf of σ. It is sufficient to show that for any vertex z to the right of t in σ, there is a neighbor of z between t and z in σ. Vertex w is a neighbor of y and lies between t and y in σ. For any other vertex of C, vertex y lies between t and this vertex. Any neighbor z of t in I is adjacent to at least one other vertex since t is not a cut vertex of G. If this vertex is x, then z is to the left of t in σ. Otherwise, the neighbor of z lies between t and z in σ. Thus, in any case, t is not the parent of any of its neighbors.

Now assume that t is an $\mathcal{L}$-branch leaf of some BFS ordering σ of G. By Theorem 3.2, vertex t is not a cut vertex of G. If $t \in I$, then Condition (1) is fulfilled. The same holds if $n(G) = 2$. Thus, we may assume that $t \in C$ and $n(G) \geq 3$. Let u be the leftmost vertex in σ such that $C \subseteq N_G[u]$. Note that $\sigma^{-1}(u) \leq 2$. In any case, the vertex after u in σ is a neighbor of u in G and, thus, a child of u in T. Hence, $u \neq t$ and, since $C \subseteq N_G[u]$, edge $tu \in E(G)$. If all neighbors of t are to the left of t in σ, then all neighbors of t are adjacent to u and, thus, Condition (1) is fulfilled. Hence, we may assume that there is at least one neighbor of t to the right of t in σ. Let x be the leftmost neighbor of t in σ that is to the right of t in σ. Let w be the parent of x in the $\mathcal{L}$-tree of σ. Since $t \prec_\sigma w \prec_\sigma x$, the choice of x implies that vertex w is not a neighbor of t. Furthermore, w has at least two neighbors, namely, vertex x and the parent of w in the $\mathcal{L}$-tree of σ. Therefore, w fulfills the properties given in Condition (2). □

All conditions given in Theorem 5.15 can straightforwardly be checked in linear time. Thus, we can recognize BFS $\mathcal{F}$-branch leaves within the same time bound.

Corollary 5.16. *Given a connected split graph G and a vertex $t \in V(G)$, we can decide in time $\mathcal{O}(n(G) + m(G))$ whether t is an $\mathcal{F}$-branch leaf of some BFS ordering of G.*

As mentioned above, we were not able to resolve the recognition problem of $\mathcal{L}$-branch leaves of BFS for chordal graphs. This is the only of the branch leaf recognition problems that remains open on chordal graphs.

5.3 Branch leaves of DFS

As we have seen in Theorem 4.4, the $\mathcal{F}$-branch leaf recognition problem of DFS is NP-complete on chordal graphs. However, we can give a simple characterization of DFS $\mathcal{F}$-branch leaves of split graphs and interval graphs. To this end, we use the following general result.

Theorem 5.17. *Let $\mathcal{G}$ be a hereditary graph class whose connected elements all have a dominating path. Then it holds for any connected graph $G \in \mathcal{G}$ with $n(G) \geq 2$:*

A vertex $t \in V(G)$ is an $\mathcal{F}$-branch leaf of some DFS ordering if and only if t is not a cut vertex of G.

Proof. Let $G \in \mathcal{G}$ be a connected graph with $n(G) \geq 2$ and t be a vertex of G. If t is an $\mathcal{F}$-branch leaf of some DFS ordering, then, due to Theorem 3.2, t is not a cut vertex.

If t is not a cut vertex, then $G - t$ is connected. As $\mathcal{G}$ is hereditary, the graph $G - t$ is also an element of $\mathcal{G}$ and, thus, $G - t$ has a dominating path P. There is a DFS ordering σ of G starting with P. It is obvious that t is an $\mathcal{F}$-branch leaf of σ. □

In connected split graphs, any Hamiltonian path of the clique forms a dominating path of the whole graph. In connected interval graphs, every path from the vertex with the first ending interval to the vertex with last starting interval is a dominating path. Thus, Theorem 5.17 can be applied to both graph classes. As cut vertices can be identified in linear time, this leads directly to a linear-time algorithm for the DFS $\mathcal{F}$-branch leaf recognition on split graphs and on interval graphs.

Corollary 5.18. *Given a connected split graph G or a connected interval graph G and a vertex $t \in V(G)$, we can decide in time $\mathcal{O}(n(G) + m(G))$ whether t is an $\mathcal{F}$-branch leaf of some DFS ordering of G.*

Note that the linear-time result for interval graphs can be extended to the super class of AT-free graphs since connected AT-free graphs also have dominating paths [17].

On interval graphs, the DFS end-vertex problem can be solved in linear time [2]. By Theorem 4.1, this implies that the DFS $\mathcal{L}$-branch leaves of interval graphs can also be recognized in linear time. In contrast, it is NP-hard to decide whether a vertex of a split graph is a DFS $\mathcal{L}$-branch leaf (see Corollary 4.2). Thus, on split graphs the $\mathcal{L}$-branch leaf recognition of DFS is harder than the $\mathcal{F}$-branch leaf recognition of DFS (if P $\neq$ NP), a surprising contrast to the hardness of the DFS $\mathcal{F}$-tree recognition problem [35] and the easiness of the DFS $\mathcal{L}$-tree recognition problem on general graphs [23, 26]. Recall that we have made a similar observation for the complexity of the branch leaf and tree recognition of BFS (see Theorem 4.7 and Corollary 4.10).

6 Open Questions

We have studied the complexity of the problem that asks whether a vertex of a graph can be the leaf of a search tree of particular searches. An overview of the obtained results can be found in Table 1 in Section 1. This table also presents several open cases. One of the most intriguing of them is the complexity of the $\mathcal{L}$-branch leaf recognition of LDFS on general graphs. As mentioned at the end of Section 4, it is the only combination of a search tree paradigm and a graph search considered here for which the complexity of the branch leaf recognition problem is left open.

Open Question 1. *What is the computational complexity of deciding whether a vertex of graph G is an $\mathcal{L}$-branch leaf of some LDFS ordering of G?*

In Section 5, we have considered the complexity of these problems on chordal graphs. We were able to present linear-time algorithms for almost all of the considered problems on this graph class. The only remaining open case is the $\mathcal{L}$-branch leaf recognition of BFS.

Open Question 2. *What is the computational complexity of deciding whether a vertex of a chordal graph G is an $\mathcal{L}$-branch leaf of some BFS ordering of G?*

Note that this problem is also open for the subclass of interval graphs.

ORCID iDs

Robert Scheffler https://orcid.org/0000-0001-6007-4202

References

[1] J. Beisegel, *Convexity in Graphs: Vertex Order Characterisations and Graph Searching*, Ph.D. thesis, BTU Cottbus-Senftenberg, 2020, https://nbn-resolving.org/urn:nbn:de:kobv:co1-opus4-51452.

[2] J. Beisegel, C. Denkert, E. Köhler, M. Krnc, N. Pivač, R. Scheffler and M. Strehler, On the end-vertex problem of graph searches, *Discrete Math. Theor. Comput. Sci.* **21** (2019), Paper No. 13, 20 pp., doi:10.23638/DMTCS-21-1-13, https://doi.org/10.23638/DMTCS-21-1-13.

[3] J. Beisegel, C. Denkert, E. Köhler, M. Krnc, N. Pivač, R. Scheffler and M. Strehler, The recognition problem of graph search trees, *SIAM J. Discrete Math.* **35** (2021), 1418–1446, doi:10.1137/20M1313301, https://doi.org/10.1137/20M1313301.

[4] J. Beisegel, E. Köhler, F. Ratajczak, R. Scheffler and M. Strehler, Graph search trees and the intermezzo problem, in: *49th International Symposium on Mathematical Foundations of Computer Science*, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, volume 306 of *Leibniz Int. Proc. Inform.*, pp. Art. No. 22, 18 pp., 2024, doi:10.4230/lipics.mfcs.2024.22, https://doi.org/10.4230/lipics.mfcs.2024.22.

[5] J. Beisegel, E. Köhler, R. Scheffler and M. Strehler, Linear time LexDFS on chordal graphs, in: *28th Annual European Symposium on Algorithms*, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, volume 173 of *Leibniz Int. Proc. Inform.*, pp. Art. No. 13, 13 pp., 2020, doi:10.4230/lipics.esa.2020.13, https://doi.org/10.4230/lipics.esa.2020.13.

[6] A. Berry, J. R. S. Blair, P. Heggernes and B. W. Peyton, Maximum cardinality search for computing minimal triangulations of graphs, *Algorithmica* **39** (2004), 287–298, doi:10.1007/s00453-004-1084-3, https://doi.org/10.1007/s00453-004-1084-3.

[7] A. A. Bertossi and M. A. Bonuccelli, Hamiltonian circuits in interval graph generalizations, *Inform. Process. Lett.* **23** (1986), 195–200, doi:10.1016/0020-0190(86)90135-3, https://doi.org/10.1016/0020-0190(86)90135-3.

[8] M. Borassi, P. Crescenzi and M. Habib, Into the square: on the complexity of some quadratic-time solvable problems, in: *Proceedings of ICTCS 2015, the 16th Italian Conference on Theoretical Computer Science*, Elsevier Sci. B. V., Amsterdam, volume 322 of *Electron. Notes Theor. Comput. Sci.*, 2016 pp. 51–67, doi:10.1016/j.entcs.2016.03.005, https://doi.org/10.1016/j.entcs.2016.03.005.

[9] P. Charbit, M. Habib and A. Mamcarz, Influence of the tie-break rule on the end-vertex problem, *Discrete Math. Theor. Comput. Sci.* **16** (2014), 57–72, doi:10.46298/dmtcs.2081, https://doi.org/10.46298/dmtcs.2081.

[10] V. Chepoi and F. Dragan, A linear-time algorithm for finding a central vertex of a chordal graph, in: *Algorithms—ESA '94 (Utrecht)*, Springer, Berlin, volume 855 of *Lecture Notes in Comput. Sci.*, pp. 159–170, 1994, doi:10.1007/BFb0049406, https://doi.org/10.1007/BFb0049406.

[11] D. G. Corneil, A simple 3-sweep LBFS algorithm for the recognition of unit interval graphs, *Discrete Appl. Math.* **138** (2004), 371–379, doi:10.1016/j.dam.2003.07.001, https://doi.org/10.1016/j.dam.2003.07.001.

[12] D. G. Corneil, B. Dalton and M. Habib, LDFS-based certifying algorithm for the minimum path cover problem on cocomparability graphs, *SIAM J. Comput.* **42** (2013), 792–807, doi:10.1137/11083856X, https://doi.org/10.1137/11083856X.

[13] D. G. Corneil, F. F. Dragan, M. Habib and C. Paul, Diameter determination on restricted graph families, *Discrete Appl. Math.* **113** (2001), 143–166, doi:10.1016/S0166-218X(00)00281-X, https://doi.org/10.1016/S0166-218X(00)00281-X.

[14] D. G. Corneil, J. Dusart, M. Habib, A. Mamcarz and F. de Montgolfier, A tie-break model for graph search, *Discrete Appl. Math.* **199** (2016), 89–100, doi:10.1016/j.dam.2015.06.011, https://doi.org/10.1016/j.dam.2015.06.011.

[15] D. G. Corneil, E. Köhler and J.-M. Lanlignel, On end-vertices of lexicographic breadth first searches, *Discrete Appl. Math.* **158** (2010), 434–443, doi:10.1016/j.dam.2009.10.001, https://doi.org/10.1016/j.dam.2009.10.001.

[16] D. G. Corneil and R. M. Krueger, A unified view of graph searching, *SIAM J. Discrete Math.* **22** (2008), 1259–1276, doi:10.1137/050623498, https://doi.org/10.1137/050623498.

[17] D. G. Corneil, S. Olariu and L. Stewart, Asteroidal triple-free graphs, *SIAM J. Discrete Math.* **10** (1997), 399–430, doi:10.1137/S0895480193250125, https://doi.org/10.1137/S0895480193250125.

[18] D. G. Corneil, S. Olariu and L. Stewart, Linear time algorithms for dominating pairs in asteroidal triple-free graphs, *SIAM J. Comput.* **28** (1999), 1284–1297, doi:10.1137/S0097539795282377, https://doi.org/10.1137/S0097539795282377.

[19] D. G. Corneil, S. Olariu and L. Stewart, The LBFS structure and recognition of interval graphs, *SIAM J. Discrete Math.* **23** (2009), 1905–1953, doi:10.1137/S0895480100373455, https://doi.org/10.1137/S0895480100373455.

[20] J. Gorzny, *On End Vertices of Search Algorithms*, Master's thesis, University of Victoria, 2015, http://hdl.handle.net/1828/6506.

[21] J. Gorzny and J. Huang, End-vertices of LBFS of (AT-free) bigraphs, *Discrete Appl. Math.* **225** (2017), 87–94, doi:10.1016/j.dam.2017.02.027, https://doi.org/10.1016/j.dam.2017.02.027.

[22] M. Habib, R. McConnell, C. Paul and L. Viennot, Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing, *Theor. Comput. Sci.* **234** (2000), 59–84, doi:10.1016/S0304-3975(97)00241-7, https://doi.org/10.1016/S0304-3975(97)00241-7.

[23] T. Hagerup, Biconnected graph assembly and recognition of DFS trees (1985), doi:10.22028/D291-26437, https://doi.org/10.22028/D291-26437.

[24] T. Hagerup and M. Nowak, *Recognition of spanning trees defined by graph searches*, Technical Report A 85/08, Universität des Saarlandes, 1985.

[25] J. Hopcroft and R. Tarjan, Efficient planarity testing, *J. Assoc. Comput. Mach.* **21** (1974), 549–568, doi:10.1145/321850.321852, https://doi.org/10.1145/321850.321852.

[26] E. Korach and Z. Ostfeld, DFS tree construction: Algorithms and characterizations (preliminary version), in: *Graph-Theoretic Concepts in Computer Science. 14th International Workshop, WG '88, Amsterdam, The Netherlands, 1988.*, Springer-Verlag, Berlin, volume 344 of *Lecture Notes in Comput. Sci.*, pp. 87–106, 1989, doi:10.1007/3-540-50728-0_37, https://doi.org/10.1007/3-540-50728-0_37.

[27] D. Kratsch, P. Damaschke and A. Lubiw, Dominating cliques in chordal graphs, *Discrete Math.* **128** (1994), 269–275, doi:10.1016/0012-365X(94)90118-X, https://doi.org/10.1016/0012-365X(94)90118-X.

[28] D. Kratsch, M. Liedloff and D. Meister, End-vertices of graph search algorithms, in: *Algorithms and Complexity. 9th International Conference, CIAC 2015, Paris, France, 2015*, Springer, Cham, volume 9079 of *Lecture Notes in Comput. Sci.*, pp. 300–312, 2015, doi:10.1007/978-3-319-18173-8_22, https://doi.org/10.1007/978-3-319-18173-8_22.

[29] R. Krueger, G. Simonet and A. Berry, A general label search to investigate classical graph search algorithms, *Discrete Appl. Math.* **159** (2011), 128–142, doi:10.1016/j.dam.2010.02.011, https://doi.org/10.1016/j.dam.2010.02.011.

[30] U. Manber, Recognizing breadth-first search trees in linear time, *Inform. Process. Lett.* **34** (1990), 167–171, doi:10.1016/0020-0190(90)90155-Q, https://doi.org/10.1016/0020-0190(90)90155-Q.

[31] G. Rong, Y. Cao, J. Wang and Z. Wang, Graph searches and their end vertices, *Algorithmica* **84** (2022), 2642–2666, doi:10.1007/s00453-022-00981-5, https://doi.org/10.1007/s00453-022-00981-5.

[32] D. J. Rose, Triangulated graphs and the elimination process, *J. Math. Anal. Appl.* **32** (1970), 597–609, doi:10.1016/0022-247X(70)90282-9, https://doi.org/10.1016/0022-247X(70)90282-9.

[33] D. J. Rose, R. E. Tarjan and G. S. Lueker, Algorithmic aspects of vertex elimination on graphs, *SIAM J. Comput.* **5** (1976), 266–283, doi:10.1137/0205021, https://doi.org/10.1137/0205021.

[34] R. Scheffler, Linearizing partial search orders, in: *Graph-Theoretic Concepts in Computer Science. 48th International Workshop, WG 2022, Tübingen, Germany, 2022*, Springer, Cham, volume 13453 of *Lecture Notes in Comput. Sci.*, pp. 425–438, 2022, doi:10.1007/978-3-031-15914-5_31, https://doi.org/10.1007/978-3-031-15914-5_31.

[35] R. Scheffler, On the recognition of search trees generated by BFS and DFS, *Theoret. Comput. Sci.* **936** (2022), 116–128, doi:10.1016/j.tcs.2022.09.018, https://doi.org/10.1016/j.tcs.2022.09.018.

[36] R. Scheffler, Graph search trees and their leaves, in: *Graph-Theoretic Concepts in Computer Science. 49th International Workshop, WG 2023, Fribourg, Switzerland, 2023*, Springer, Cham, volume 14093 of *Lecture Notes in Comput. Sci.*, pp. 462–476, 2023, doi:10.1007/978-3-031-43380-1_33, https://doi.org/10.1007/978-3-031-43380-1_33.

[37] R. Scheffler, *Ready to Order? On Vertex and Edge Orderings of Graphs*, Ph.D. thesis, BTU Cottbus-Senftenberg, 2023, doi:10.26127/BTUOpen-6301, https://doi.org/10.26127/BTUOpen-6301.

[38] R. Scheffler, Recognizing LBFS trees of bipartite graphs, *Inform. Process. Lett.* **186** (2024), Paper No. 106483, 7 pp., doi:10.1016/j.ipl.2024.106483, https://doi.org/10.1016/j.ipl.2024.106483.

[39] D. R. Shier, Some aspects of perfect elimination orderings in chordal graphs, *Discrete Appl. Math.* **7** (1984), 325–331, doi:10.1016/0166-218X(84)90008-8, https://doi.org/10.1016/0166-218X(84)90008-8.

[40] J. Spinrad and R. Sritharan, Algorithms for weakly triangulated graphs, *Discrete Appl. Math.* **59** (1995), 181–191, doi:10.1016/0166-218X(93)E0161-Q, https://doi.org/10.1016/0166-218X(93)E0161-Q.

[41] R. Tarjan, Depth-first search and linear graph algorithms, *SIAM J. Comput.* **1** (1972), 146–160, doi:10.1137/0201010, https://doi.org/10.1137/0201010.

[42] R. E. Tarjan and M. Yannakakis, Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs, *SIAM J. Comput.* **13** (1984), 566–579, doi:10.1137/0213035, https://doi.org/10.1137/0213035.

[43] M. Zou, Z. Wang, J. Wang and Y. Cao, End vertices of graph searches on bipartite graphs, *Inform. Process. Lett.* **173** (2022), Paper No. 106176, 7 pp., doi:10.1016/j.ipl.2021.106176, https://doi.org/10.1016/j.ipl.2021.106176.