The Partial Search Order Problem

Robert Scheffler

Submitted: Dec 20, 2023; Accepted: Jul 27, 2025; Published: Oct 3, 2025

Abstract

In recent years, questions about the construction of special orderings of a given graph search were studied by several authors. On the one hand, the so called End-Vertex Problem introduced by Corneil et al. in 2010 asks for search orderings ending in a particular vertex. On the other hand, the problem of finding orderings that induce a given search tree was introduced already in the 1980s by Hagerup and received new attention most recently. Here, we consider a generalization of some of these problems by studying the question whether there is a search ordering that is a linear extension of a given partial order on a graph's vertex set. We show that this problem can be solved in polynomial time for Generic Search on general graphs as well as for searches called forgetful and partial orders of bounded width. Furthermore, we present polynomial-time algorithms on the classes of chordal bipartite graphs and split graphs for some searches including (Lexicographic) Breadth First Search. These algorithms generalize known algorithmic results for the End-Vertex Problem and the Search Tree Recognition Problem.

Mathematics Subject Classifications: 05C05, 05C85, 68Q25

1 Introduction

The graph searches *Breadth First Search* (BFS) and *Depth First Search* (DFS) are considered as some of the most basic algorithms in both graph theory and computer science. Taught in many undergraduate courses around the world, they are an elementary component of several graph algorithms. There are other more sophisticated graph searches, e.g., the *Lexicographic Breadth First Search* (LBFS) [33], the *Lexicographic Depth First Search* (LDFS) [16], or the *Maximum Cardinality Search* (MCS) [38] which are also used to solve several graph problems, among them the recognition problems of graph classes [9, 17, 19], the computation of minimum path covers [13] as well as of minimal triangulations [7].

In recent years, different problems of finding special search orderings have gained attention from several researchers. One of these problems is the End-Vertex Problem

Institute of Mathematics, Brandenburg University of Technology, Cottbus, Germany (robert.scheffler@b-tu.de).

introduced in 2010 by Corneil et al. [15]. It asks whether a given vertex in a graph can be visited last by some graph search. The problem was motivated by *multi-sweep algorithms* where a search is applied several times to a graph such that every application starts in the vertex where the preceding search ordering has ended. Corneil et al. [15] showed that the End-Vertex Problem is NP-complete for LBFS on weakly chordal graphs. Similar results were obtained for other searches such as BFS, DFS, LDFS, and MCS [2, 11, 30], while for several graph classes, among them split graphs, polynomial-time algorithms were presented in [2, 11, 15, 20, 30]. An overview of these results is given in [36, Table 1].

An important structure closely related to a graph search is the corresponding search tree. Such a tree contains all the vertices of the graph and for every vertex different from the start vertex exactly one edge to a vertex preceding it in the search ordering. These 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 the trees generated by DFS can be used for fast planarity testing [24]. The problem of deciding whether a given spanning tree of a graph can be obtained by a particular search was introduced by Hagerup [21] in 1985, who presented a linear-time algorithm for recognizing DFS trees. In the same year, Hagerup and Nowak [22] gave a similar result for the BFS tree recognition. Independently from Hagerup and Nowak, linear-time recognition algorithms were obtained by Korach and Ostfeld [25] for DFS trees and by Manber [28] for BFS trees. Peng et al. [29] studied the recognition of shortest-path trees of weighted graphs, a generalization of BFS Tree Recognition.

In the light of the extensive results on the End-Vertex Problem, Beisegel et al. [4] reconsidered the Search Tree Recognition Problem most recently. To generalize the problem to other graph searches, they 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 neighbor visited before it, i.e., DFS-like trees. They showed, among other things, that the $\mathcal{F}$-tree recognition is NP-complete for LBFS, LDFS, and MCS on weakly chordal graphs, while the $\mathcal{L}$-tree recognition is polynomial for LDFS on general graphs. Furthermore, both the $\mathcal{F}$-tree recognition and the $\mathcal{L}$-Tree Recognition Problem can be solved in polynomial time for LBFS, LDFS, and MCS on chordal graphs. These results were complemented in [35] by showing that the $\mathcal{L}$-tree recognition of BFS and the $\mathcal{F}$-tree recognition of DFS are NP-complete. Recently, the $\mathcal{L}$-tree recognition of *Generic Search*, i.e., the search where every vertex can be visited next as long as it has an already visited neighbor, was proven to be NP-complete [5].

Our Contribution. There seems to be a strong relationship between the complexity of the End-Vertex Problem and the recognition problem of $\mathcal{F}$-trees. There are many combinations of graph classes and graph searches where both problems are NP-complete or both problems are solvable in polynomial time (compare Table 1 and Table 2 in [36]). To further study this relationship, we present a generalization of these two problems by introducing the *Partial Search Order Problem (PSOP)* of a graph search $\mathcal{A}$. Given a graph G and a partial order π on its vertex set, it asks whether there is a search ordering produced by $\mathcal{A}$ which is a linear extension of π. Note that this problem has already been presented at a conference [34]. Afterwards, Rong et al. [32] provided a polynomial-time algorithm for the PSOP of MCS on chordal graphs.

We show that a greedy algorithm solves the PSOP of Generic Search. As a direct consequence of the complexity of the End-Vertex Problem, the PSOP is NP-hard for many searches even if the height and the dimension of the partial order is bounded by 2. We contrast this result by showing that at least for *forgetful* searches we can solve the PSOP in polynomial time if the width of the partial order is bounded. Here, forgetful means that the order of the visited vertices does not matter to decide which vertex has to be visited next. One example of these searches is MCS.

Furthermore, we present a polynomial-time algorithm for the PSOP of LBFS on chordal bipartite graphs, i.e., bipartite graphs without induced cycles of length larger than four. This result also implies a polynomial-time algorithm for the End-Vertex Problem on chordal bipartite graphs, a generalization of the result by Gorzny and Huang [20] on the End-Vertex Problem of LBFS on AT-free bipartite graphs. Extending results given by Beisegel et al. [6], we show that the PSOP of DFS, LDFS, and MCS cannot be solved in polynomial time on chordal bipartite graphs, unless $\mathsf{P} = \mathsf{NP}$.

Finally, we give polynomial-time algorithms for the PSOP of BFS, LBFS, and MCS on split graphs. These algorithms generalize known results on the End-Vertex Problem [2, 11] and the $\mathcal{F}$-Tree Recognition Problem [3, 22] of these searches on this graph class.

Note that the algorithms for PSOP of LBFS and BFS on chordal bipartite graphs and split graphs make use of the *One-Before-All (OBA) Problem*, a useful auxiliary problem introduced in this paper. It is a generalization of the problem of finding a linear extension of a partial order. Instead of forcing one element of the ground set to be to the left of another – as the constraints in partial orders do – the constraints of the OBA Problem require that some element of a subset of the ground set has to be to the left of all elements of another subset. We will show here that this problem can be solved in linear time.

Note that an extended abstract of this paper appeared in the proceedings of the workshop WG 2022 [34].

Structure of the Paper. The paper is organized as follows. In Section 2, we present basic definitions and results on partial orders and graphs and introduce the graph searches considered here. Section 3 introduces the Partial Order Search Order Problem and the One-Before-All Problem. Section 4 presents the algorithmic results for Generic Search and forgetful searches on general graphs. In Section 5, we give the polynomial-time algorithm for the PSOP of LBFS on chordal bipartite graphs and show that this result cannot be extended to DFS, LDFS, or MCS, unless $\mathsf{P} = \mathsf{NP}$. The algorithmic results for split graphs are presented in Section 6. The final section presents some open questions and further research directions.

2 Preliminaries

2.1 Partial Orders and Linear Orderings

A *linear ordering* of a finite set X is a bijection $\sigma : \{1, 2, \dots, |X|\} \to X$. We will often refer to linear orderings simply as orderings. For an ordering σ, we denote by $\sigma^{-1}(x)$ the position of the element $x \in X$ in σ. Given two elements x and y in X we say that x is *to the left* (resp. *to the right*) of y if $\sigma^{-1}(x) < \sigma^{-1}(y)$ (resp. $\sigma^{-1}(x) > \sigma^{-1}(y)$) and we denote this by $x \prec_\sigma y$ (resp. $x \succ_\sigma y$). Given a linear ordering σ of a set X and a subset $Y \subseteq X$, we call an element $y \in Y$ the *σ-leftmost* (resp. *σ-rightmost*) element of Y if $y \prec_\sigma z$ (resp. $y \succ_\sigma z$) holds for all elements $z \in Y \setminus \{y\}$. Given two orderings σ of X and ρ of Y with $X \cap Y = \emptyset$, the ordering $\tau = \sigma \oplus \rho$ is the *concatenation* of σ and ρ, i.e., $\tau(i) = \sigma(i)$ if $1 \leqslant i \leqslant |X|$ and $\tau(i) = \rho(i - |X|)$ if $|X| < i \leqslant |X \cup Y|$. If $v \notin X$, then $v \oplus \sigma$ (resp. $\sigma \oplus v$) denotes the concatenation of σ with the linear ordering of the set $\{v\}$.

A *(binary) relation* $\mathcal{R}$ on a set X is a subset of the set $X^2 = \{(x, y) \mid x, y \in X\}$. The set X is called the *ground set of* $\mathcal{R}$. A *partial order* π on a set X is a reflexive, antisymmetric and transitive relation on X. We also denote $(x, y) \in \pi$ by $x \prec_\pi y$ if $x \neq y$. The tuple (X, π) is also called *partially ordered set*. For a relation $\mathcal{R}$ on X, the *reflexive and transitive closure* $\mathcal{R}'$ of $\mathcal{R}$ is the smallest relation $\mathcal{R}' \supseteq \mathcal{R}$ that is reflexive and transitive. A *minimal element* of a partial order π on X is an element $x \in X$ for which there is no element $y \in X$ with $y \prec_\pi x$.

A *linear extension* of a relation $\mathcal{R}$ is a linear ordering σ of X that fulfills all conditions of $\mathcal{R}$, i.e., if $(x, y) \in \mathcal{R}$, then $x \prec_\sigma y$. We will often use the term "σ extends $\mathcal{R}$" in that case. It is easy to see that a relation has a linear extension if and only if its reflexive and transitive closure is a partial order.

The *dimension* of a partial order π on a set X is the smallest number ℓ for which there is a set $\{\sigma_1, \dots, \sigma_\ell\}$ of ℓ linear extensions of π such that π is equal to the intersection of these linear extensions, i.e., $x \prec_\pi y$ if and only if $x \prec_{\sigma_i} y$ for all $i \in \{1, \dots, \ell\}$. A *chain* of a partial order π on a set X is a set of elements $\{x_1, \dots, x_k\} \subseteq X$ such that $x_1 \prec_\pi x_2 \prec_\pi \dots \prec_\pi x_k$. The *height* of π is number of elements of the largest chain of π. An *antichain* of π is a set of elements $\{x_1, \dots, x_k\} \subseteq X$ such that $x_i \not\prec_\pi x_j$ for every $i, j \in \{1, \dots, k\}$. The *width* of π is the number of elements of the largest antichain of π.

Theorem 1 (Dilworth [18]). *Every partially ordered set (X, π) whose partial order π has width k can be partitioned into k disjoint chains.*

2.2 Graphs

The graphs considered in this paper are finite, undirected, simple and connected. Given a graph G, we denote by $V(G)$ the set of vertices and by $E(G)$ the set of edges. We denote the size of $V(G)$ by $n(G)$ and the size of $E(G)$ by $m(G)$. 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 an edge between u and v in G is denoted by uv. The *closed neighborhood* of v in G is the set $N_G[v] = N_G(v) \cup \{v\}$. Similarly, we define the *neighborhood $N_G(A)$ of a set*

$A \subseteq V(G)$ as follows: $N_G(A) := (\bigcup_{v\in A} N_G(v)) \setminus A$. We denote the *degree* of a vertex, i.e., the size of its neighborhood, by $d_G(v)$.

The *distance* of a vertex v to a vertex w is the number of edges of the shortest path from v to w. The *eccentricity* of a vertex v in the graph G is the maximum distance of v to any vertex w and is denoted by $ecc_G(v)$. The set $N_G^\ell(v)$ contains all vertices whose distance to the vertex v is equal to ℓ, in particular, $N_G^1(v) = N_G(v)$. A *vertex ordering* of G is a linear ordering of $V(G)$.

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 *split graph* G is a graph whose vertex set can be partitioned into sets C and I, such that C is a clique in G and I is an independent set in G. We call such a partition a *split partition.* A graph is *bipartite* if its vertex set can be partitioned into two independent sets X and Y. If $|X| = |Y|$, then we call the graph *balanced.* A bipartite graph G is called *chordal bipartite* if every induced cycle contained in G has a length of four.

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 the *parent* of vertex w if v is an element of the unique path from w to the root r and the edge vw is contained in T. A vertex w is called *child* of v if v is the parent of w.

2.3 Graph Searches

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. Every prefix of an $\mathcal{A}$-ordering of G is an $\mathcal{A}$*-prefix of* G. Most graph searches considered in this paper can be formalized adapting a framework introduced by Corneil et al. [14] (a similar framework is given in [27]). 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 elements of $\mathcal{P}(\mathbb{N}^+)$ (see Algorithm 1). The set of $\mathcal{A}$-orderings of a graph G for an instance $\mathcal{A}$ of *Label Search* is the set of vertex orderings that can be the output of the search if the graph G is given.

In the following, we define the searches considered in this paper by presenting suitable partial orders $\prec_\mathcal{A}$ (see [14]). 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 orderings* for GS orderings and *connected graph search* for graph searches whose search orderings are GS orderings.

The partial label order $\prec_{BFS}$ for *Breadth First Search* (BFS) is defined as follows: $A \prec_{BFS} B$ if and only if $A = \emptyset$ and $B \neq \emptyset$ or $\min(A) > \min(B)$. For the *Lexicographic Breadth First Search* (LBFS) [33] we consider the partial order $\prec_{LBFS}$ with $A \prec_{LBFS} B$ if and only if $A \subsetneq B$ or $\min(A \setminus B) > \min(B \setminus A)$.

The partial label order $\prec_{DFS}$ for the well-known *Depth First Search* (DFS) is defined as follows: $A \prec_{DFS} B$ if and only if $A = \emptyset$ and $B \neq \emptyset$ or $\max(A) < \max(B)$. The *Lexicographic Depth First Search* (LDFS) was introduced in 2008 by Corneil and Krueger [16]

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

```
Input: A graph $G$
Output: A search ordering $\sigma$ of $G$
begin
    foreach $v \in V(G)$ do $label(v) \leftarrow \emptyset$;
    for $i \leftarrow 1$ to $|V(G)|$ do
        $Eligible \leftarrow \{x \in V(G) \mid x$ unvisited and $\nexists$ unvisited $y \in V(G)$
                                such that $label(x) \prec_{\mathcal{A}} label(y)\}$;
        let $v$ be an arbitrary vertex in $Eligible$;
        $\sigma(i) \leftarrow v$;                /* visits $v$ and assigns the number $i$ to it */
        foreach unvisited vertex $w \in N(v)$ do $label(w) \leftarrow label(w) \cup \{i\}$;
```

as a DFS equivalent to LBFS. It can be defined why the partial order $\prec_{LDFS}$ where $A \prec_{LDFS} B$ if and only if $A \subsetneq B$ or $\max(A \setminus B) < \max(B \setminus A)$.

The *Maximum Cardinality Search* (MCS) [38] 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 $A \subsetneq B$. If $A \prec_{MNS} B$, then it also holds that $A \prec_{LBFS} B$, $A \prec_{LDFS} B$, and $A \prec_{MCS} B$. Thus, every ordering produced by LBFS, LDFS, or MCS is also an MNS ordering.

Search orderings of some of these searches can be characterized using so-called *4-point conditions* (see [16]). These conditions consider special triples of vertices in the search ordering. A triple (x, y, z) with $x, y, z \in V(G)$ and $x \prec_\sigma y \prec_\sigma z$ is called *curious triple of* σ if $xz \in E(G)$ and $xy \notin E(G)$.[1] The 4-point conditions of MNS, BFS, and LBFS are given in the following lemma.

Lemma 2 (Brandstädt et al. [8], Corneil and Krueger [16]). *Let G be a graph and σ be a vertex ordering of G.*

(i) σ is an MNS ordering of G if and only if for every curious triple (x, y, z) of σ there is a vertex $w \in V(G)$ with $w \prec_\sigma y$ and $wy \in E(G)$ and $wz \notin E(G)$.

(ii) σ is a BFS ordering of G if and only if for every curious triple (x, y, z) of σ there is a vertex $w \in V(G)$ with $w \prec_\sigma x$ and $wy \in E(G)$.

(iii) σ is an LBFS ordering of G if and only if for every curious triple (x, y, z) of σ there is a vertex $w \in V(G)$ with $w \prec_\sigma x$ and $wy \in E(G)$ and $wz \notin E(G)$.

In the search algorithms following the framework given in Algorithm 1, any of the vertices in the set *Eligible* can be chosen as the next vertex. Some applications use special variants of these searches that involve tie-breaking. For an instantiation $\mathcal{A}$ of Algorithm 1, we define the graph search $\mathcal{A}^+$ as follows: Add a vertex ordering ρ of graph

[1]The term *curious triple* is taken from a figure caption in [16]. The word "curious" is used here in the sense of "odd" and not in the sense of "inquisitive".

G as additional input and replace line 6 in Algorithm 1 with "let v be the vertex in *Eligible* that is leftmost in ρ".[2] Note that this corresponds to the algorithm TBLS given in [14]. The search ordering $\mathcal{A}^+(\rho)$ is unique since there are no ties to break.

The following lemma gives a strong property for all $\mathcal{A}^+$-searches considered here and will be used several times.

Lemma 3. *Let G be a graph and ρ be a vertex ordering of G. Let $\mathcal{A}$ be a graph search in {GS, BFS, LBFS, LDFS, MCS, MNS} and σ the $\mathcal{A}^+(\rho)$ ordering of G. If $u \prec_\sigma v$ and $v \prec_\rho u$, then there is a vertex x with $x \prec_\sigma u$ and $xu \in E(G)$ and $xv \notin E(G)$.*

Proof. Assume for contradiction that for every vertex x with $x \prec_\sigma u$ and $xu \in E(G)$ it also holds $xv \in E(G)$. It can be checked easily that the strict partial order $\prec_\mathcal{A}$ given for any of the given graph searches $\mathcal{A}$ has the following property: If $A \subseteq B$, then for every C with $B \prec_\mathcal{A} C$ it also holds $A \prec_\mathcal{A} C$. Consider the step i in the computation of σ where u was visited by the search $\mathcal{A}$. Since $v \prec_\rho u$, the vertex v was not an element of *Eligible*. Therefore, there was an unvisited vertex x with $label(v) \prec_\mathcal{A} label(x)$. Due to the assumption on u and v, it held that $label(u) \subseteq label(v)$. By the property of $\prec_\mathcal{A}$ mentioned above, it held that $label(u) \prec_\mathcal{A} label(x)$ and u was not an element of *Eligible*; a contradiction. $\square$

Some $\mathcal{A}^+$-searches can be implemented as linear-time algorithms.

Theorem 4 (see, e.g., Scheffler [36, Theorem 0.2.21])**.** *Given a graph G and a vertex ordering ρ of G, the BFS$^+(\rho)$ ordering and the LBFS$^+(\rho)$ ordering of G can be computed in $\mathcal{O}(n(G) + m(G))$ time.*

Note that, although GS, MCS, and MNS can be implemented as linear-time algorithm, no such implementation is known for GS$^+$, MCS$^+$, and MNS$^+$.

Finally, we formalize a property that is fulfilled by only some of the considered searches. In these searches, the ordering of the already visited vertices does not matter for the choice of the next vertex. We will call those searches *forgetful* as they do not have to remember the ordering of the visited vertices.

Definition 5. A graph search $\mathcal{A}$ is called *forgetful on a graph G* if the following condition holds for every $\mathcal{A}$-prefix $\sigma = (v_1, \dots, v_{k+1})$ of G and every ordering σ^* of the vertices $v_1, \dots, v_k$: If σ^* is an $\mathcal{A}$-prefix of G, then $\sigma^* \oplus v_{k+1}$ is an $\mathcal{A}$-prefix of G. A graph search $\mathcal{A}$ is called *forgetful* if it is forgetful on every graph G.

It is straightforward to check that GS, MCS, and MNS are forgetful.

3 Two Ordering Problems

3.1 The Partial Search Order Problem

We start by formally introducing the problem considered in this paper.

[2]Note that other authors define v to be the rightmost vertex in ρ.

Partial Search Order Problem (PSOP) of graph search $\mathcal{A}$

Instance: A graph G, a partial order π on $V(G)$.

Question: Is there an $\mathcal{A}$-ordering of G that is a linear extension of π?

We also consider a special variant where the start vertex of the search ordering is fixed. We call this problem the *rooted Partial Search Order Problem* or rooted PSOP for short. Note that the general problem and the rooted problem are polynomial-time equivalent. If we have a polynomial-time algorithm to solve the rooted problem, then we can apply it $n(G)$ times to the graph G to solve the general problem. On the other hand, the rooted problem with fixed start vertex r can be solved by a general algorithm. To this end, we add all the tuples (r, v), $v \in V(G)$, to the partial order π. In the following we always assume that a given start vertex r is a minimal element of the partial order π because otherwise we can reject the input immediately.

The *End-Vertex Problem* of a graph search $\mathcal{A}$ was introduced by Corneil et al. [15]. It asks whether a particular vertex $t \in V(G)$ can be the last vertex of an $\mathcal{A}$-ordering of a given graph G. Corneil et al. also introduced the *Beginning-End-Vertex Problem*[3] which gets an additional vertex s as input and asks whether there is an $\mathcal{A}$-ordering beginning with s and ending with t. The End-Vertex Problem can be encoded by the partial order $\pi := \{(u, v) \mid u, v \in V(G), u = v \text{ or } v = t\}$ while the Beginning-End-Vertex Problem can be encoded by $\pi' := \{(u, v) \mid u, v \in V(G), u = v \text{ or } v = t \text{ or } u = s\}$. This leads to the following result.

Theorem 6. *Given a graph G, the (Beginning-)End-Vertex Problem of a graph search $\mathcal{A}$ can be solved by solving the PSOP of $\mathcal{A}$ on G for a partial order of size $\mathcal{O}(n(G))$.*

It follows directly that the PSOP is NP-complete for BFS, DFS, LBFS, LDFS, MCS, and MNS since the corresponding End-Vertex Problem is NP-complete [2, 11, 15]. The partial order π given for the End-Vertex Problem has dimension 2 as it is the intersection of its linear extensions $(v_1, \ldots, v_{n-1}, t)$ and $(v_{n-1}, \ldots, v_1, t)$. Furthermore, it has height 2. Thus, the PSOP of the mentioned searches is also hard for partial orders of dimension and height 2.

Corollary 7. *The PSOP of BFS, DFS, LBFS, LDFS, MCS, and MNS is* NP*-complete even if the given partial order has dimension 2 and height 2.*

Note that a partial order with dimension 1 is a linear ordering and, thus, the PSOP with such a partial order reduces to the recognition problem of search orderings of the respective search. This problem can be solved in polynomial time for all the searches considered here. On the other hand, for a partial order π with height 1 any linear ordering of their ground set is a linear extension of π. Thus, the PSOP is trivial for these partial orders (as long as there is a search ordering for each graph). The width of the partial

[3]Note that Corneil et al. [15] call the problem *Start-vertex, End-vertex Problem.* Subsequent publications use the term *Beginning-End-Vertex Problem* [11, 20]

order given for the End-Vertex Problem is not bounded by a constant. In fact, we will show in Section 4.2 that the PSOP of MCS and MNS can be solved in polynomial time if the width of the partial order is bounded.

Another problem that is generalized by the PSOP is the recognition of certain search trees, called $\mathcal{F}$-trees.

Definition 8 (Beisegel et al. [4]). Given a connected search ordering σ of a connected graph G, we define the *$\mathcal{F}$-tree* of σ to be the tree consisting of the vertex set $V(G)$ and an edge from each vertex v with $\sigma^{-1}(v) > 1$ to the σ-leftmost vertex in $N_G(v)$.

The *$\mathcal{F}$-Tree Recognition Problem* [4] of graph search $\mathcal{A}$ asks for a given graph G and given spanning tree T of G whether T is the $\mathcal{F}$-tree of some $\mathcal{A}$-ordering of G. The *rooted $\mathcal{F}$-Tree Recognition Problem* of $\mathcal{A}$ has a vertex $r \in V(G)$ as additional input and the question is whether there is an $\mathcal{A}$-ordering starting with r that has T as its $\mathcal{F}$-tree. The rooted $\mathcal{F}$-Tree Recognition Problem is also a special case of the (rooted) PSOP as the following result shows.

Proposition 9. *Let $\mathcal{A}$ be a connected graph search. Given a connected graph G and a spanning tree T of G rooted in r, we define the relation $\mathcal{R}_T := \{(x, y) \mid x$ is parent of y in T or there is a child z of x in T with $yz \in E(G)\}$. Let σ be an $\mathcal{A}$-ordering of G starting with r. Then the tree T is the $\mathcal{F}$-tree of σ if and only if σ is a linear extension of $\mathcal{R}_T$.*

Proof. Assume T is the $\mathcal{F}$-tree of the $\mathcal{A}$-ordering σ starting in r. It is sufficient to show that the parent of every vertex $v \neq r$ in T is the σ-leftmost neighbor of v. Assume for contradiction that this is not the case. Let x be the σ-leftmost vertex $\neq r$ whose σ-leftmost neighbor y is not its parent. Note that $y \prec_\sigma x$ since $\mathcal{A}$ is a connected search and G is connected. Due to the definition of the $\mathcal{F}$-tree, the edge xy is an element of T and, thus, y has to be a child of x. However, if $r = y$, then this cannot be as r has no parent. Otherwise, the σ-leftmost neighbor of y is to the left of y and is not its parent, contradicting the choice of x.

For the other direction, let the $\mathcal{A}$-ordering σ be a linear extension of $\mathcal{R}_T$. Let y be an arbitrary vertex in G and let x be the parent of y in T. Due to the construction of $\mathcal{R}_T$, vertex x is to the left of y in σ. Furthermore, every other neighbor of y is to the right of x in σ. Therefore, x is the σ-leftmost neighbor of y and the edge xy is part of the $\mathcal{F}$-tree of σ. Hence, this $\mathcal{F}$-tree is equal to T. □

This implies the following theorem.

Theorem 10. *Given a graph G, the rooted $\mathcal{F}$-Tree Recognition Problem of a graph search $\mathcal{A}$ can be solved by solving the (rooted) PSOP of $\mathcal{A}$ on G for a partial order of size $\mathcal{O}(n(G)^2)$.*

Note that the general $\mathcal{F}$-Tree Recognition Problem without fixed start vertex can be solved by solving the PSOP for every possible root.

3.2 One-Before-All Orderings

When tackling the PSOP of BFS and LBFS, the following observation is essential. If some vertex u is to the left of another vertex v in a BFS (LBFS) ordering, then the leftmost (exclusive) neighbor of u has to be to the left of all exclusive neighbors of v. The algorithms for the PSOP of BFS and LBFS that we will present in Sections 5 and 6 have to ensure that this property holds for a bunch of pairs u and v. To avoid redundancies in the description of these algorithms, we introduce the following auxiliary problem.

One-Before-All Problem (OBAP)

Instance: A set M, a family $\mathcal{Q} \subseteq \mathcal{P}(M)$ of subsets of M, a relation $\mathcal{R}$ on $\mathcal{Q}$ where $(A, B) \in \mathcal{R}$ implies that $A \cap B = \emptyset$.

Question: Is there a *One-Before-All ordering (OBA ordering)* of M, i.e., a linear ordering σ of M that fulfills the following condition: for all $A, B \in \mathcal{Q}$ with $(A, B) \in \mathcal{R}$, the σ-leftmost vertex of $A \cup B$ is not in B?

The OBAP generalizes the problem of finding a linear extension of a partial order on a set M. Instead of considering tuples containing two elements of M, we consider tuples containing two disjoint subsets of M. Then we are looking for a linear ordering of the elements of M such that for every tuple (A, B) there is one element of A that is to the left of every element of B. In fact, we can encode the problem of finding a linear extension of some partial order π on M as an OBAP instance by setting $\mathcal{Q} = \{\{x\} \mid x \in M\}$ and $\mathcal{R} = \{(\{x\},\ \{y\}) \mid (x, y) \in \pi,\ x \neq y\}$.

In the following we describe how we can solve the OBAP in time linear in the input size $|M| + |\mathcal{R}| + \sum_{A \in \mathcal{Q}} |A|$ (see Algorithm 2 for the pseudocode). For every set $B \in \mathcal{Q}$, we introduce a counter $r(B)$ containing the number of tuples $(A, B) \in \mathcal{R}$. For every element $x \in M$, the variable $t(x)$ counts the number of sets $A \in \mathcal{Q}$ with $x \in A$ and $r(A) > 0$. Our algorithm creates the ordering σ greedily from left to right. It is not difficult to see that an element x can be chosen next if and only if $t(x) = 0$. As long as such an element exists, the algorithm chooses one, deletes all tuples (A, B) with $x \in A$ from $\mathcal{R}$ and updates the r- and the t-values. If no such element exists, then the algorithm returns "No ordering".

Theorem 11. *Given a set M, a family $\mathcal{Q} \subseteq \mathcal{P}(M)$ of subsets of M and a relation $\mathcal{R} \subseteq \mathcal{Q} \times \mathcal{Q}$, Algorithm 2 returns an OBA ordering of M if and only if such an ordering exists. The running time of the algorithm is $\mathcal{O}(|M| + |\mathcal{R}| + \sum_{A \in \mathcal{Q}} |A|)$.*

Proof. Assume for contradiction that Algorithm 2 returns the ordering σ but σ is not a feasible OBA ordering. Then there are sets $A, B \in \mathcal{Q}$ with $(A, B) \in \mathcal{R}$ such that there is an $x \in B$ that is to the left of each element of A in σ. Tuples are only deleted from $\mathcal{R}$ if an element of the left set of the tuple received an index in σ. Therefore, the tuple (A, B) was still in $\mathcal{R}$ when x was inserted into S. This is a contradiction, because $t(x)$ would have been greater than zero.

Now assume that there is an OBA ordering σ' but Algorithm 2 returns "No ordering". Let X be the subset of M containing all elements that have not been inserted into σ

Algorithm 2: Greedy OBA

```
   Input: A set $M$, a family $\mathcal{Q} \subseteq \mathcal{P}(M)$ of subsets of $M$, a relation $\mathcal{R} \subseteq \mathcal{Q} \times \mathcal{Q}$
   Output: An OBA ordering $\sigma$ of the elements in $M$ or "No ordering"
 1 begin
 2     $r(B) \leftarrow |\{(A,B) \in \mathcal{R}\}|$   for all $B \in \mathcal{Q}$;
 3     $t(x) \leftarrow |\{A \in \mathcal{Q} \mid r(A) > 0, x \in A\}|$   for all $x \in M$;
 4     $S \leftarrow \{x \in M \mid t(x) = 0\}$;
 5     $i \leftarrow 0$;
 6     while $S \neq \emptyset$ do
 7         let $x$ be an element in $S$;
 8         $S \leftarrow S \setminus \{x\}$;
 9         $i \leftarrow i + 1$;
10         $\sigma(i) \leftarrow x$;
11         foreach $A \in \mathcal{Q}$ with $x \in A$ do
12             $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{A\}$;
13             foreach $(A,B) \in \mathcal{R}$ do
14                 $\mathcal{R} \leftarrow \mathcal{R} \setminus \{(A,B)\}$;
15                 $r(B) \leftarrow r(B) - 1$;
16                 if $r(B) = 0$ then
17                     foreach $y \in B$ do
18                         $t(y) \leftarrow t(y) - 1$;
19                         if $t(y) = 0$ then $S \leftarrow S \cup \{y\}$;
20     if $i = |M|$ then return $\sigma$;
21     else return "No ordering";
```

during the execution of the algorithm. Let x be the σ'-leftmost element of X. Since x was not inserted into S by the algorithm, the condition $t(x) > 0$ still holds at the end of Algorithm 2. This implies that there are sets A and B with $(A, B) \in \mathcal{R}$, $A \subseteq X$ and $x \in B$. By the choice of x, all elements of A are to the right of x in σ'. This is a contradiction to the fact that σ' is an OBA ordering.

It remains to show that the running time of the algorithm is linear in the input size. Let $\gamma = \sum_{A \in \mathcal{Q}} |A|$. Note that $|\mathcal{Q}| \leqslant \gamma + 1$. We use the following data structure. Every element $x \in M$ has a linked list with pointers to the sets in $\mathcal{Q}$ containing x. Every set A in $\mathcal{Q}$ has a linked list with pointers to its position in the linked lists of all its elements. Furthermore, A is associated with a linked list containing pointers to the tuples $(A, B) \in \mathcal{R}$. All these data structures can be created in $\mathcal{O}(|M| + |\mathcal{R}| + \gamma)$ time. The same holds for all the steps before the `while`-loop. We iterate at most $|M|$ times through the `while`-loop. Every set A in $\mathcal{Q}$ is visited at most once in total in the `foreach`-loop starting in line 11 because the set is removed from $\mathcal{Q}$ afterwards. Note that this removal can be done in $\mathcal{O}(|A|)$ time, due to our data structure. Similarly, each tuple of $\mathcal{R}$ is visited at most once in total in the `foreach`-loop starting in line 13. As the r-value of each set B

becomes zero at most once, the set B is visited at most once in the `foreach`-loop starting in line 17. Thus, the running time of the algorithm is bounded by $\mathcal{O}(|M| + |\mathcal{R}| + \gamma)$. □

4 Partial Search Orders of General Graphs

4.1 Greedy Algorithm for GS

We start with a simple algorithm for the rooted PSOP of GS. First it visits the given start vertex r. Afterwards it looks for a vertex with an already visited neighbor among all vertices that are minimal in the remaining partial order. If no such vertex exists, then it rejects. Otherwise, it visits one of these vertices next. The pseudocode of this procedure is given in Algorithm 3.

Algorithm 3: Rooted PSOP of GS

```
   Input: Connected graph $G$, a vertex $r \in V(G)$, a partial order $\pi$ on $V(G)$
   Output: GS ordering of $G$ starting in $r$ that extends $\pi$ or "$\pi$ cannot be extended"
1  begin
2      $L^{<}(v) := \{u \mid u \prec_\pi v\}$      for all $v \in V(G)$;
3      $L^{>}(v) := \{u \mid v \prec_\pi u\}$      for all $v \in V(G)$;
4      $S \leftarrow \{r\}$;
5      $i \leftarrow 0$;
6      while $S \neq \emptyset$ do
7          let $v$ be some element of $S$;
8          remove $v$ from $S$ and from all lists $L^{<}(w)$ with $w \in L^{>}(v)$;
9          $i \leftarrow i + 1$;
10         $\sigma(i) \leftarrow v$;
11         foreach $w \in N_G(v)$ do
12             mark $w$;
13             if $L^{<}(w)$ is empty then $S \leftarrow S \cup \{w\}$;
14         foreach $w \in L^{>}(v)$ do
15             if $w$ is marked and $L^{<}(w)$ is empty then $S \leftarrow S \cup \{w\}$;
16     if $i = n(G)$ then return $\sigma$;
17     else return "$\pi$ cannot be extended";
```

Theorem 12. *Given a connected graph G and a partial order on $V(G)$, Algorithm 3 solves the rooted PSOP of GS in $\mathcal{O}(n(G) + m(G) + |\pi|)$ time.*

Proof. Algorithm 3 does not insert a vertex x different from r into S before x is marked. The vertex x is marked when a neighbor of x is visited. Since only vertices contained in S are visited, we know that an ordering σ returned by Algorithm 3 is a GS ordering of G. Furthermore, σ is a linear extension of π since a vertex x is only visited in σ if for every $(v, x) \in \pi$ vertex v is already visited in σ.

If Algorithm 3 returns "π cannot be extended", then there is at least one vertex in $V(G)$ that was not visited and, thus, this vertex was not inserted into S. Assume for contradiction that there is a GS ordering σ of G starting in r that is a linear extension of π and let v be the σ-leftmost vertex that was not inserted into S. Since σ is a GS ordering, there is a neighbor w of v which is to the left of v in σ. Due to the choice of v, vertex w has been added to S and, therefore, v was marked by Algorithm 3. As v was not inserted into S, there is a vertex x that was also not inserted into S with $x \prec_\pi v$. However, due to the choice of v, vertex x is to the right of v in σ. This is a contradiction because σ was chosen to be a linear extension of π.

It remains to show that Algorithm 3 can be implemented such that it has a linear running time. For every vertex $v \in V(G)$, we have two lists of vertices, the list $L^<(v)$ contains all vertices with $u \prec_\pi v$ and the list $L^>(v)$ contains all vertices with $v \prec_\pi u$. Additionally, the entry of u in $L^>(v)$ contains a pointer to the entry of v in $L^<(u)$. This ensures that line 8 costs only $\mathcal{O}(|L^>(v)|)$ time. Overall, the algorithm iterates through every neighborhood and every set $L^>(v)$ only a constant number of times. Therefore, the total running time is bounded by $\mathcal{O}(n(G) + m(G) + |\pi|)$. □

It remains open how we can find a sufficient start vertex of the GS ordering in linear time. Thus, the unrooted PSOP can only be solved in $\mathcal{O}(n(G) \cdot m(G))$ time so far.

4.2 Dynamic Program for Forgetful Searches

Due to Corollary 7, it is unlikely that there is a polynomial-time algorithm for the PSOP of (L)BFS, (L)DFS, MCS, or MNS. Nevertheless, one may ask for the best running time of an algorithm that solves these problems. The trivial algorithm for the PSOP of a search $\mathcal{A}$ checks for every vertex ordering of a graph G that extends the given partial order whether it is an $\mathcal{A}$-ordering of G. This may need $\Omega(n!)$ time for an n-vertex graph. However, Kratsch et al. [26] showed that at least for the End-Vertex Problems of BFS and DFS the running time can be improved to $\mathcal{O}(2^n \cdot p(n))$ for some polynomial p. One may ask whether such an algorithm can also be given for the PSOP of some searches. Similarly, Corollary 7 implies that it is unlikely that we can find a polynomial-time algorithm for the PSOP of (L)BFS, (L)DFS, MCS, or MNS if the height or the dimension of the given partial order is bounded by some constant. However, fixing the width of the partial order may simplify the PSOP significantly.

In this section, we tackle both questions for the PSOP of forgetful searches, in particular for MCS and MNS. We present an algorithm that on the one hand, has single exponential running time for arbitrary partial orders and, on the other hand, has polynomial running time for partial orders of fixed width k. The algorithm is inspired by an idea of Colbourn and Pulleyblank [12] for solving a scheduling problem with precedence constraints (see also [6] for details).

The approach uses dynamic programming (see Algorithm 4 for the pseudocode). Assume we are given a forgetful search $\mathcal{A}$, a graph G and a partial order π on $V(G)$ of width k. Due to Theorem 1, we can partition the vertex set $V(G)$ into k disjoint chains $(C_1, \ldots, C_k)$ of π. We define $C_i[j]$ to be the j-th element of chain C_i, i.e., it holds that

$C_i[j] \prec_\pi C_i[\ell]$ if and only if $j < \ell$. If a vertex $C_i[\ell]$ is contained in a prefix of a linear extension of π, then all the vertices $C_i[j]$ with $j < \ell$ are also contained in that prefix. Thus, we can encode the elements of such a prefix via a tuple $X = (x_1, \ldots, x_k)$ where $0 \leqslant x_i \leqslant |C_i|$ for all $i \in \{1, \ldots, k\}$. The set of those tuples is called S. We define the *weight* of such a tuple X as the sum of its entries. Furthermore, $\mathcal{Z}(X)$ is the set of vertices encoded by X, i.e., $v \in \mathcal{Z}(X)$ if and only if $v = C_i[j]$ with $j \leqslant x_i$ for some $i \in \{1, \ldots, k\}$ and some $j \in \{1, \ldots, |C_i|\}$.

The procedure uses a vector $M \in \{0,1\}^{|S|}$ that stores for every of those tuples $X = (x_1, \ldots, x_k)$ whether there is an $\mathcal{A}$-prefix of G that is a prefix of a linear extension of π and that contains exactly the elements of $\mathcal{Z}(X)$. If such an $\mathcal{A}$-prefix exists, then $M(X)$ is 1 and the variable $\sigma(X)$ holds such an $\mathcal{A}$-prefix. Additionally, for every tuple $X \in S$ we have an array $F(X)$ that contains for every vertex $v \in V(G)$ the number of vertices $u \in V(G) \setminus \mathcal{Z}(X)$ with $u \prec_\pi v$.

We fill the vector M, the prefix variables σ and the arrays F inductively. The entry $M(0, \ldots, 0)$ is set to 1, $\sigma(0, \ldots, 0)$ is set to the empty ordering and $F(0, \ldots, 0)$ contains the numbers of predecessors of every vertex in π. Assume we have found the values of $M(Y)$, $\sigma(Y)$ and $F(Y)$ for every tuple Y of weight $\ell - 1$. Let $X = (x_1, \ldots, x_k)$ be a tuple of weight ℓ. For every entry $x_i > 0$, we check the following three conditions for the tuple $Y = (x_1, \ldots, x_{i-1}, x_i - 1, x_{i+1}, \ldots, x_k)$:

- Is $M[Y]$ equal 1, i.e., is there an $\mathcal{A}$-prefix that contains exactly the vertices of $\mathcal{Z}(Y)$ and that is the prefix of a linear extension of π?
- If so, is the entry of $C_i[x_i]$ in $F(Y)$ equal to 0, i.e., is the x_i-th vertex of C_i minimal when all vertices of $\mathcal{Z}(Y)$ are deleted from π?
- If so, is $\sigma(Y) \oplus C_i[x_i]$ an $\mathcal{A}$-prefix of G?

If there is an entry x_i in X for which all three conditions are fulfilled, then we set $M(X)$ to 1 and $\sigma(X)$ to $\sigma(Y) \oplus C_i[x_i]$. We create $F(X)$ from $F(Y)$ by decrementing all entries of vertices u with $C_i[x_i] \prec_\pi u$ by one. If no entry x_i fulfills all three conditions, then $M(X)$ is set to 0.

Theorem 13. *Let $\mathcal{A}$ be a forgetful graph search. Given a graph G and a partial order π of width k, Algorithm 4 solves the PSOP of $\mathcal{A}$ on G and π in $\mathcal{O}(\min\{n(G)^k, 2^{n(G)}\} \cdot (k \cdot t_\mathcal{A}(G) + n(G)))$ time, where $t_\mathcal{A}(G)$ is the time that is needed to decide for a given $\mathcal{A}$-prefix σ of G and a vertex $v \in V(G)$ whether $\sigma \oplus v$ is an $\mathcal{A}$-prefix of G.*

Proof. Let G be a graph and π be a partial order on $V(G)$. For some tuple $X \in S$, we call an $\mathcal{A}$-prefix σ a *nice prefix of* X if σ contains exactly the elements of $\mathcal{Z}(X)$ and there are no vertices $v \in \mathcal{Z}(X)$ and $u \in V(G) \setminus \mathcal{Z}(X)$ with $u \prec_\pi v$. To prove the correctness of the algorithm, we show that the following claim holds for every tuple $X \in S$.

Claim 14. *There is a nice $\mathcal{A}$-prefix of X if and only if $M(X) = 1$. If $M(X) = 1$, then $\sigma(X)$ is a nice $\mathcal{A}$-prefix of X and for every $v \in V(G)$ the entry $F(X)[v]$ contains the number of vertices $u \in V(G) \setminus \mathcal{Z}(X)$ with $u \prec_\pi v$.*

Algorithm 4: PSOP of forgetful search $\mathcal{A}$

```
   Input: Connected graph $G$, partial order $\pi$ on $V(G)$ of width $k \geqslant 2$
   Output: $\mathcal{A}$-ordering $\sigma$ of $G$ extending $\pi$ or "$\pi$ cannot be extended"
 1 begin
 2     $(C_1, \dots, C_k) \leftarrow$ decomposition of $(V(G), \pi)$ into $k$ disjoint chains;
 3     $S \leftarrow \{(x_1, \dots, x_k) \mid 0 \leqslant x_i \leqslant |C_i|\}$;
 4     $\sigma(0, \dots, 0) \leftarrow ()$;
 5     $F(0, \dots, 0)[v] \leftarrow |\{u \mid u \prec_\pi v\}| \quad \forall v \in V(G)$;
 6     $M(x_1, \dots, x_k) \leftarrow 0 \quad \forall (x_1, \dots, x_k) \in S$;
 7     $M(0, \dots, 0) \leftarrow 1$;
 8     for $\ell \leftarrow 1$ to $n(G)$ do
 9         foreach $X = (x_1, \dots, x_k) \in S$ with $\sum_{i=1}^k x_i = \ell$ do
10             foreach $i \in \{1, \dots, k\}$ with $x_i > 0$ do
11                 $v \leftarrow C_i[x_i]$;
12                 $Y \leftarrow (x_1, \dots, x_{i-1}, x_i - 1, x_{i+1}, \dots, x_k)$;
13                 if $M(Y) = 1$ and $F(Y)[v] = 0$ and $\sigma(Y) \oplus v$ is $\mathcal{A}$-prefix of $G$ then
14                     $M(X) \leftarrow 1$;
15                     $\sigma(X) = \sigma(Y) \oplus v$;
16                     $F(X)[u] \leftarrow F(X)[u] \quad \forall u \in V(G)$;
17                     foreach $u \in V(G)$ with $v \prec_\pi u$ do $F(X)[u] \leftarrow F(X)[u] - 1$;
18                     break;
19     if $M(|C_1|, \dots, |C_n|) = 1$ then return $\sigma(|C_1|, \dots, |C_n|)$;
20     else return "$\pi$ cannot be extended";
```

Proof of Claim. We prove the claim by induction over the weight of X. For $X = (0, \dots, 0)$, the claim trivially holds true. Thus, assume that for all tuples $X \in S$ with weight $\ell - 1$ the claim holds true. Let $X = (x_1, \dots, x_k)$ be a tuple in S with weight ℓ.

If $M(X) = 1$, then Algorithm 4 found an index i such that for $Y = (x_1, \dots, x_{i-1}, x_i - 1, x_{i+1}, \dots, x_k)$ and $v = C_i[x_i]$ it holds that $\sigma(Y) \oplus v$ is an $\mathcal{A}$-prefix of G. Furthermore, $F(Y)[v]$ was equal to 0 and, thus, there is no vertex $u \in V(G) \setminus \mathcal{Z}(X)$ with $u \prec_\pi v$. Therefore, $\sigma(X) = \sigma(Y) \oplus v$ is a nice $\mathcal{A}$-prefix of X. The correctness of the entries of $F(X)$ follows directly from the construction.

It remains to show that $M(X) = 1$ if there is a nice $\mathcal{A}$-prefix of X. Let $\tau = (v_1, \dots, v_q)$ be a nice $\mathcal{A}$-prefix of X. It holds that v_q is the x_i-th entry of C_i for some $i \in \{1, \dots, k\}$. Furthermore, the ordering $(v_1, \dots, v_{q-1})$ is a nice $\mathcal{A}$-prefix of $Y = (x_1, \dots, x_{i-1}, x_i - 1, x_{i+1}, \dots, x_k)$. As Y has weight $\ell - 1$, it follows by induction that $M(Y) = 1$ and $\sigma(Y)$ is a nice $\mathcal{A}$-prefix of Y. Note that $\sigma(Y)$ does not have to be equal to $(v_1, \dots, v_{q-1})$. However, as $\mathcal{A}$ is forgetful and τ is a nice $\mathcal{A}$-prefix of G, the ordering $\sigma(Y) \oplus v_k$ is also a nice $\mathcal{A}$-prefix of G. Hence, Algorithm 4 will set $M(X)$ to 1. ◀

We conclude the proof of the theorem by showing the claimed running time. The algorithm iterates at most $|S| \cdot k$ times through the `foreach`-loop starting in 10. The conditions in line 13 can be checked in $\mathcal{O}(t_{\mathcal{A}}(G))$ time. Thus, every iteration that does not enter the `if`-block starting in line 13 needs $\mathcal{O}(t_{\mathcal{A}}(G))$ time. For every tuple X, this `if`-block is entered at most once. Processing the `if`-block needs $\mathcal{O}(n(G))$ time to copy $\sigma(Y)$ and the values of $F(Y)$ and $\mathcal{O}(n(G))$ time to update the values of $F(X)$ for all u with $v \prec_\pi u$. Thus, the overall running time is bounded by $\mathcal{O}(|S| \cdot (k \cdot t_{\mathcal{A}}(G) + n(G))$. It remains to bound the size of S. As the k entries of the tuples in S lie between 0 and $n(G)$, the size of S is bounded by $\mathcal{O}(n(G)^k)$. Furthermore, for every subset of $V(G)$ there is at most one tuple in S. Therefore, the size of S is also bounded by $2^{n(G)}$. This leads to the claimed running time. □

In the following we give concrete bounds for MNS and MCS.

Theorem 15. *The PSOP of MNS can be solved in* $\mathcal{O}(\min\{n(G)^k, 2^{n(G)}\} \cdot (k \cdot n(G)^2))$ *time on a graph G and a partial order π of width k.*

Proof. We have to bound the size of $t_{\mathcal{A}}(G)$ in Theorem 13. We iterate through the given MNS-prefix σ and construct the labels of all unvisited vertices. This can straightforwardly be done in $\mathcal{O}(n(G)^2)$ time as we only have to iterate through all adjacency lists at most once. Afterwards, we have to check whether the label of v is strictly contained in the label of another unvisited vertex. This can also be done in $\mathcal{O}(n(G)^2)$ time by marking the elements of the label of v and iterating through all the other labels. □

For MCS we could use the same approach. However, we can improve Algorithm 4 to get an even better running time bound.

Theorem 16. *The PSOP of MCS can be solved in* $\mathcal{O}(\min\{n(G)^k, 2^{n(G)}\} \cdot n(G)$ *time on a graph G.*

Proof. Additionally to the prefix $\sigma(X)$ and the array $F(X)$ we also store the MCS label (i.e. the number of neighbors in $\mathcal{Z}(X)$) of all vertices in $V(G) \setminus \mathcal{Z}(X)$ as well as the maximal MCS label among all these vertices. With this information, we can check in $\mathcal{O}(1)$ whether vertex v can be visited next in the MCS ordering. If we have found such a vertex, then we can update the MCS label of every vertex and the maximal MCS label in $\mathcal{O}(n(G))$ time. As this has to be done at most once for every tuple $X \in S$, we get the claimed time bound. □

Due to Theorem 6, we can apply the algorithm also to the End-Vertex Problem of these two searches. As the partial order used to solve the End-Vertex Problem has width $n(G) - 1$, it holds that $\min\{n(G)^k, 2^{n(G)}\} = 2^{n(G)}$.

Corollary 17. *Given a graph G, the End-Vertex Problem of MNS can be solved in* $\mathcal{O}(2^{n(G)} \cdot n(G)^3)$ *time and the End-Vertex Problem of MCS can be solved in* $\mathcal{O}(2^{n(G)} \cdot n(G))$ *time on G.*

This complements the sub-factorial-time algorithms for the End-Vertex Problems of BFS and DFS given by Kratsch et al. [26]. Note that Cao et al. [10] already gave a sub-factorial running time bound for the MCS End-Vertex Problem. However, they did not specify the exact exponent of the polynomial factor.

5 Partial Search Orders of Chordal Bipartite Graphs

5.1 Polynomial-time Algorithm for LBFS

Gorzny and Huang [20] showed that the End-Vertex Problem of LBFS is NP-complete on bipartite graphs but can be solved in polynomial time on AT-free bipartite graphs, a subclass of chordal bipartite graphs. In this section we generalize their result by showing that the PSOP of LBFS can be solved in polynomial time on chordal bipartite graphs.

The following result will be a key ingredient of our approach. It shows that for two vertices x and y in the same layer $N_G^i(r)$ of a BFS starting in r that have a common neighbor in the succeeding layer $N_G^{i+1}(r)$, it holds that the neighborhoods of x and y in the preceding layer $N_G^{i-1}(r)$ are comparable.

Lemma 18. *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 $N_G(y) \cap N_G^{i-1}(r) \subseteq N_G(x)$.*

Proof. Assume x has a neighbor u in $N_G^{i-1}(r)$ which is not a neighbor of y and y has a neighbor v in $N_G^{i-1}(r)$ which is not a neighbor of x. Let p_u and p_v be shortest paths from u to r and v to r, respectively, which shares the maximum number of edges. Let w be the first vertex (starting from u and v, respectively) that is both in p_u and p_v. Let p_u^* be the subpath of p_u between u and w and let p_v^* be the subpath of p_v between v and w. The paths p_u^* and p_v^* and the edges xu, yv, xz and yz form a cycle C of length at least six in G. Due to the choice of p_u and p_v and the fact that two vertices of G can only be adjacent if they lie in consecutive layers, the cycle C is an induced cycle; a contradiction since G is chordal bipartite. □

Algorithm 5 presents the pseudocode of the algorithm for the rooted PSOP of LBFS on chordal bipartite graphs (see Figure 1 for an illustration). We assume that the partial order π contains only tuples where both elements are in the same layer of a BFS ordering starting with r. Otherwise, the tuple is trivially fulfilled by any BFS ordering starting in r or no such BFS ordering fulfills the tuple. The algorithm constructs an OBAP instance with set $\mathcal{Q}_i \subseteq \mathcal{P}(N_G^i(r))$ and $\mathcal{R}_i \subseteq \mathcal{Q}_i \times \mathcal{Q}_i$ for any layer i of the BFS. First we add the tuple $(\{x\}, \{y\})$ to the set $\mathcal{R}_i$ for every tuple $(x, y) \in \pi$ with $x, y \in N_G^i(r)$. Now the algorithm iterates through all layers, starting in the last one. For any element $(A, B) \in \mathcal{R}_i$, the algorithm computes the sets A' and B' which contain the neighbors of set A and B in layer $i-1$ that are not neighbors of the respective other set. The algorithm inserts a tuple (A'', B') to the relation $\mathcal{R}_{i-1}$ where the set A'' contains those elements of A' that are adjacent to any neighbor of A' in layer $i-2$. Note that such vertices exists, due to Lemma 18. At the end, the algorithm checks whether for every OBAP instance $(N_G^i(r), \mathcal{Q}_i, \mathcal{R}_i)$ there is an OBA ordering. If this is not the case, then the algorithm rejects. Otherwise, it concatenates the computed OBA orderings. The resulting ordering ρ is used as tie-breaker for an LBFS$^+$ whose result is returned by the algorithm. The following lemma is a consequence of the construction of the elements of $\mathcal{R}_i$ and Lemma 18.

Algorithm 5: Rooted PSOP of LBFS on chordal bipartite graphs

Input: Connected chordal bipartite graph G, vertex $r \in V(G)$, partial order π on $V(G)$
Output: LBFS ordering σ of G starting with r and extending π or "π cannot be extended"

```
begin
    $k \leftarrow ecc_G(r)$;
    $\mathcal{Q}_i \leftarrow \{\{x\} \mid x \in N_G^i(r)\} \quad \forall i \in \{1, \dots, k\}$;
    $\mathcal{R}_i \leftarrow \{(\{x\}, \{y\}) \mid x, y \in N_G^i(r),\ x \prec_\pi y\} \quad \forall i \in \{1, \dots, k\}$;
    for $i \leftarrow k$ downto 2 do
        foreach $(A, B) \in \mathcal{R}_i$ do
            $A' \leftarrow [N_G(A) \cap N_G^{i-1}(r)] \setminus N_G(B)$;
            $A'' \leftarrow \{v \in A' \mid N_G(v) \cap N_G^{i-2}(r) = N_G(A') \cap N_G^{i-2}(r)\}$;
            $B' \leftarrow [N_G(B) \cap N_G^{i-1}(r)] \setminus N_G(A)$;
            $\mathcal{Q}_{i-1} \leftarrow \mathcal{Q}_{i-1} \cup \{A'', B'\}$;
            $\mathcal{R}_{i-1} \leftarrow \mathcal{R}_{i-1} \cup \{(A'', B')\}$;
    let $\rho$ be an empty vertex ordering;
    for $i \leftarrow k$ downto 1 do
        if there is an OBA ordering $\tau$ for input $(N_G^i(r),\ \mathcal{Q}_i,\ \mathcal{R}_i)$ then $\rho \leftarrow \tau \oplus \rho$;
        else return "$\pi$ cannot be extended";
    $\rho \leftarrow r \oplus \rho$;
    return LBFS$^+(\rho)$ ordering $\sigma$ of $G$;
```

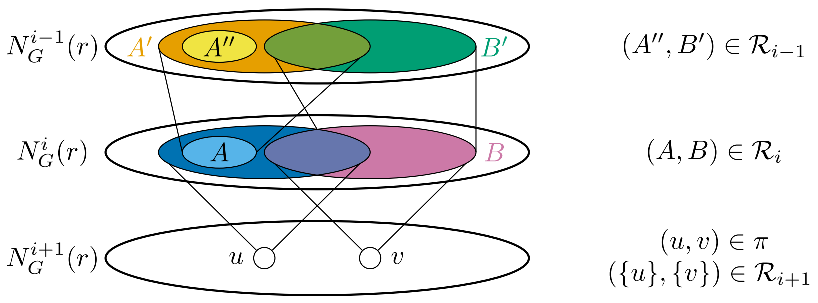
$N_G^{i-1}(r)$
A'
A''
B'
$(A'', B') \in \mathcal{R}_{i-1}$
$N_G^i(r)$
A
B
$(A, B) \in \mathcal{R}_i$
$N_G^{i+1}(r)$
u
v
$(u, v) \in \pi$
$(\{u\}, \{v\}) \in \mathcal{R}_{i+1}$

Figure 1: Example for the construction of the $\mathcal{R}$-relations in Algorithm 5. The sets connected to a vertex or a set represent the neighborhood of the vertex or the set in the respective layer. The small sets that are drawn within larger sets contain those vertices with the maximal neighborhood in the preceding layer. As shown, the tuple $(u, v) \in \pi$ with $u, v \in N_G^{i+1}(r)$ implies the tuple $(A, B) \in \mathcal{R}_i$ and this tuple implies the tuple $(A'', B') \in \mathcal{R}_{i-1}$.

Lemma 19. *Let $(A, B) \in \mathcal{R}_i$. For any $x \in A$ it holds that $N_G(A) \cap N_G^{i-1}(r) \subseteq N_G(x)$ and if $B \neq \emptyset$ then there is a vertex $y \in B$ with $N_G(B) \cap N_G^{i-1}(r) \subseteq N_G(y)$.*

Proof. The first claim follows immediately from the construction of $\mathcal{R}_i$ in Algorithm 5 (see lines 4, 8, and 11).

For the second claim we first observe that the claim trivially holds if (A, B) was inserted into $\mathcal{R}_i$ in line 4 as $|B| = 1$. Furthermore, it follows directly that the claim holds for all elements of $\mathcal{R}_k$ where $k = ecc_G(r)$. Now assume that the claim holds for all elements of $\mathcal{R}_i$ and let (A', B') be an element of $\mathcal{R}_{i-1}$ with $B' \neq \emptyset$ that was inserted into $\mathcal{R}_{i-1}$ in line 11. Then there is an element $(A, B) \in \mathcal{R}_i$ with $B' \subseteq N_G(B) \cap N_G^{i-1}(r)$ and, by assumption on $\mathcal{R}_i$, there is a vertex $x \in B$ with $B' \subseteq N_G(x)$. This means that all elements of B' have a common neighbor in $N_G^i(r)$. It follows from Lemma 18 that the neighborhoods of the elements of B' in $N_G^{i-2}(r)$ can be ordered by inclusion. Thus, there is a vertex $y \in B'$ with $N_G(B') \cap N_G^{i-2}(r) = N_G(y) \cap N_G^{i-2}(r) \subseteq N_G(y)$. □

Using this lemma, we can show the correctness of Algorithm 5.

Theorem 20. *Given a connected chordal bipartite graph G, a partial order π on $V(G)$ and a vertex $r \in V(G)$, Algorithm 5 returns in $\mathcal{O}(|\pi| \cdot n(G)^2)$ time an LBFS ordering of G that starts with r and is a linear extension of π if and only if such an ordering exists.*

Proof. We first assume that Algorithm 5 returns an ordering σ. Clearly, this ordering is an LBFS ordering. In the following we will show that the ordering σ fulfills all OBA constraints given by the relations $\mathcal{R}_i$. Since we have assumed that all $x, y \in V(G)$ with $(x, y) \in \pi$ are elements of the same layer, all constraints of π are covered by constraints of some $\mathcal{R}_i$. Thus, the result implies that σ is a linear extension of π.

Assume for contradiction that σ does not fulfill all constraints of the relations. Let i be the *minimal* index such that there is a tuple $(A, B) \in \mathcal{R}_i$ that is not fulfilled by σ, i.e., there is a vertex $y \in B$ that is to the left of every element of A in σ. Consider the ordering ρ that was used in Algorithm 5 as tie-breaker to construct the LBFS ordering σ (see line 17). The ordering ρ was constructed by concatenating the OBA orderings of all layers including the i-th layer. Therefore, the OBA constraint (A, B) is fulfilled in ρ, i.e., there is a vertex $x \in A$ that is to the left of y in ρ. Due to Lemma 3, there is a vertex $z \prec_\sigma y$ with $zy \in E(G)$ and $zx \notin E(G)$. If $i = 1$, then this is not possible. Since all layers are independent sets, the only neighbor of the vertices in $N_G^1(r)$ that is to the left of some of them is the vertex r and this vertex is adjacent to all vertices in $N_G^1(r)$. Thus, we may assume that $i > 1$. Then there is a tuple $(A'', B') \in \mathcal{R}_{i-1}$ for which it holds that $A'' \subseteq [N_G(A) \setminus N_G(B)] \cap N_G^{i-1}(r)$ and $B' = [N_G(B) \setminus N_G(A)] \cap N_G^{i-1}(r)$ (see lines 7–11). The vertex z is in $N_G(B)$. Due to Lemma 19, the neighborhood of x in $N_G^{i-1}(r)$ is equal to the neighborhood of A in $N_G^{i-1}(r)$ and, thus, x is adjacent to all elements of A''. As $z \notin N_G(x)$, vertex z is not in $N_G(A)$ and, thus, vertex z is in B' and B' is not empty. It follows from the choice of i that the tuple (A'', B') is fulfilled by σ, i.e., there is a vertex $v \in A''$ that is to the left of any element of B' in σ. Vertex v is not adjacent to y but vertex v is adjacent to x. Furthermore, any vertex w with $w \prec_\sigma v$ is not in B' and, thus, w is either adjacent to x or not adjacent to y. Hence, (v, y, x) forms curious triple of σ

that does not fulfill the 4-point condition of LBFS given in Lemma 2 (iii); a contradiction since σ is an LBFS ordering of G.

Now assume that the LBFS ordering τ of G is a linear extension of π. We will show that τ fulfills all OBA constraints in every set $\mathcal{R}_i$. Therefore, for every $\mathcal{R}_i$ the subordering of τ containing the elements of $N_G^i(r)$ is an OBA ordering for input $(N_G^i(r), \mathcal{Q}_i, \mathcal{R}_i)$ and Algorithm 5 never reaches line 15. Hence, it returns some LBFS ordering which is a linear extension of π as was shown above.

Assume for contradiction that τ does not fulfill all OBA constraints. Let i be the *maximal* index such that there is a tuple $(A'', B') \in \mathcal{R}_i$ that is not fulfilled by τ, i.e., there is a vertex $y' \in B'$ that is to the left of every element of A'' in τ. Since τ is a linear extension of π, the tuple (A'', B') was inserted into $\mathcal{R}_i$ in line 11 and not in line 4. Therefore, there is a tuple $(A, B) \in \mathcal{R}_{i+1}$ with $A'' \subseteq [N_G(A) \setminus N_G(B)] \cap N_G^i(r)$ and $B' = [N_G(B) \setminus N_G(A)] \cap N_G^i(r)$. Due to the choice of i, the constraint given by (A, B) is fulfilled, i.e, there is a vertex $x \in A$ that is to the left of every element of B in τ. As B' is not empty, the set B is also not empty. Due to Lemma 19, there is a vertex $y \in B$ such that $N_G(B) \cap N_G^i(r) \subseteq N_G(y)$. It holds that $yy' \in E(G)$ and $xy' \notin E(G)$. The tuple (y', x, y) is a curious triple of τ. Due to Lemma 2 (iii), there is a vertex $z \prec_\tau y'$ with $zx \in E(G)$ and $zy \notin E(G)$. Let z be the τ-leftmost vertex fulfilling this property. Since τ is a BFS ordering and G is bipartite, it holds that $z \in N_G^i(r)$. The fact that $zy \notin E(G)$ implies that vertex z is not an element of $N_G(B)$. Since z is to the left of y', vertex z is not in A'', due to the choice of y'. However, $z \in [N_G(A) \setminus N_G(B)] \cap N_G^i(r)$. Due to Lemma 19, there is a vertex in A that is adjacent to all vertices in $N_G(A) \cap N_G^i(r)$. It follows from Lemma 18 that the neighborhoods in $N_G^{i-1}(r)$ of the vertices contained in $N_G(A) \cap N_G^i(r)$ are ordered by inclusion. Therefore, there is a vertex z' in the set $X = [N_G(A) \setminus N_G(B)] \cap N_G^i(r)$ whose neighborhood in $N_G^{i-1}(r)$ is equal to the neighborhood of the set X in $N_G^{i-1}(r)$. Due to the construction of A'' in Algorithm 5, the vertex z' is in A''. Furthermore, it holds that $z \prec_\tau z'$, due to the choice of z. However, it holds that $N_G(z) \cap N_G^{i-1}(r) \subsetneq N_G(z') \cap N_G^{i-1}(r)$ as z is in $[N_G(A) \setminus N_G(B)] \cap N_G^i(r)$ but not in A''. Note that each neighbor of z to the left of z in τ is in $N_G^{i-1}(r)$ and, thus, it is also a neighbor of z'. Furthermore, there is at least one neighbor a of z' in $N_G^{i-1}(r)$ that is not adjacent to z. As $a \prec_\tau z$, the tuple (a, z, z') is a curious triple of τ that does not fulfill the 4-point condition of LBFS given in Lemma 2 (iii); a contradiction.

It remains to show that the algorithm has a running time within the given bound. In a pre-processing step, we compute for each vertex v the size $d_{\text{pred}}(v)$ of the neighborhood of v in its preceding layer. This can be done in $\mathcal{O}(n(G) + m(G))$ time by computing a BFS ordering starting with r. For every tuple $(A, B) \in \mathcal{R}_i$, the vertices $v \in A$ (or $v \in B$) with the largest value $d_{\text{pred}}(v)$ are adjacent to each neighbor of A (or B, respectively) in $N_G^{i-1}(r)$, due to Lemma 19. Thus, to compute the sets A' and B' in lines 7 and 9, we only have to compare the neighborhoods of two vertices. This can be done in $\mathcal{O}(n(G))$ time. For every tuple in π, there is at most one tuple in every $\mathcal{R}_i$. Thus, the total number of tuples in the $\mathcal{R}$-sets is bounded by $|\pi| \cdot n(G)$ and the total size of the sets in these tuples is bounded by $|\pi| \cdot n(G)^2$. By Theorem 11, we need $\mathcal{O}(|\pi| \cdot n(G)^2)$ time in total to solve all OBA instances. The final LBFS$^+$ needs only linear time, due to Theorem 4. □

Due to Theorem 6, we can solve the Beginning-End-Vertex Problem of LBFS on chordal bipartite graphs by solving the rooted PSOP with a partial order of size $\mathcal{O}(n(G))$. This leads to the following time bound.

Corollary 21. *Given a connected chordal bipartite graph G, the Beginning-End-Vertex Problem of LBFS on G can be solved in $\mathcal{O}(n(G)^3)$ time. The End-Vertex Problem of LBFS on G can be solved in $\mathcal{O}(n(G)^4)$ time.*

Algorithm 5 can also be used to solve the rooted $\mathcal{F}$-Tree Recognition Problem of LBFS on chordal bipartite graphs. However, this would lead to a running time of $\mathcal{O}(n(G)^4)$ while a linear-time algorithm for the problem is given in [37].

5.2 NP-hardness for (L)DFS and MCS

In difference to the polynomial-time algorithm for the PSOP of LBFS, the PSOP of DFS and LDFS is NP-hard on chordal bipartite graphs. This follows immediately from a result of Beisegel et al. [6]. They considered special partial orders on the vertex set of a bipartite graph. If X and Y are the sets of the bipartition of the graph, they say that a partial order is *oriented from X to Y* if $u \prec_\pi v$ implies that $u \in X$ and $v \in Y$.

Theorem 22 (Beisegel et al. [6]). *The Partial Search Order Problems of DFS and LDFS are NP-complete on balanced complete bipartite graphs with bipartition $X \dot\cup Y$ and partial orders that are oriented from X to Y.*

It is easy to see that the End-Vertex Problem and the $\mathcal{F}$-Tree Recognition Problem of these searches are trivial on complete bipartite graphs. Therefore, Theorem 22 answers a question raised in [31, 34] whether the PSOP is harder than the End-Vertex Problem and the $\mathcal{F}$-Tree Recognition Problem for some searches.

Here, we extend this result also to the PSOP of MCS.

Theorem 23. *The Partial Search Order Problem of MCS is NP-complete on balanced complete bipartite graphs with bipartition $X \dot\cup Y$ and partial orders that are oriented from X to Y.*

Proof. Let G be a complete bipartite graph with $V(G) = X \dot\cup Y$. Let π be a partial order that is oriented from X to Y. We prove that there is an MCS ordering of G extending π if and only if there is an DFS ordering of G extending π.

It is easy to see that every DFS ordering of G alternates between X and Y and, thus, it is also an MCS ordering of G.

Now assume there is an MCS ordering σ of G that extends π. First we consider the start vertex of σ. If σ does not start with a vertex of set X, then the second vertex of σ is in X. If we swap the first and the second vertex of σ, the result is again an MCS ordering. As the elements of X are minimal in π, the resulting ordering is also a linear extension of π. Thus, we may assume that σ starts with a vertex of X.

If σ alternates between X and Y, then σ forms a DFS ordering of G. Thus, we may assume that there are two vertices of the same partition set that are consecutive in σ.

We call two vertices that fulfill this a *same-set pair*. Let σ be the MCS ordering of G extending π that starts with a vertex of X where the first same-set pair appears rightmost.

Since σ starts with a vertex of X, the MCS label of the vertices in Y is at least as large as the MCS label of the vertices in X as long as there is no same-set pair in σ. Hence, the first same-set pair, say v_1 and v_2, must be in Y. The next vertex after this pair, say w, must be in X. The MCS label of w was equal to the MCS label of v_2 when v_2 was visited by the MCS. Hence, we may swap v_2 and w in σ and get another MCS ordering σ'. Since π is oriented from X to Y, vertex w is minimal in π and, thus, σ' is a linear extension of π. This contradicts the choice of σ as the first same-set pair in σ' appears later than in σ. □

The result cannot be extended to (L)BFS or MNS (unless $\mathsf{P} = \mathsf{NP}$). As we have seen above, the PSOP of LBFS can be solved in polynomial time on chordal bipartite graphs and, thus, also on complete bipartite graphs. The same holds for BFS since a vertex ordering of a complete bipartite graph is a BFS ordering if and only if it is an LBFS ordering. Even better, we can solve both problems in linear time as a vertex ordering of a complete bipartite graph is an (L)BFS ordering if and only if the first vertex is followed by all the vertices of the other partition set in an arbitrary order and they are followed by the remaining vertices of the partition set of the start vertex also in arbitrary order. It is easy to check in $\mathcal{O}(n(G) + |\pi|)$ time whether such a linear extension of π exists. For MNS, we can observe that any vertex ordering of a complete bipartite graph is an MNS ordering if and only if the first two vertices of the ordering are in different partition sets. Again, the existence of such a linear extension can be checked straightforwardly in $\mathcal{O}(n(G) + |\pi|)$ time, leading to the following result.

Proposition 24. *Given a complete bipartite graph G and a partial order π, the Partial Search Order Problems of BFS, LBFS, and MNS can be solved in $\mathcal{O}(n(G) + |\pi|)$ time.*

6 Partial (L)BFS and MCS Orders of Split Graphs

Both the End-Vertex Problem and the $\mathcal{F}$-Tree Recognition Problem of several searches are well studied on split graphs (see [2, 4, 11]). In this section, we generalize some of these results to the PSOP.

We will first present some results on MNS orderings of split graphs that we use later to solve the problem for MCS and LBFS. Consider a split graph G with split partition (C, I). During a computation of an MNS ordering of G, every vertex that has labeled some vertex in I has also labeled every unvisited vertex contained in C. Therefore, we can choose a vertex of C as the next vertex as long as there are still unvisited vertices in C. Hence, it is not a problem to force a vertex of C to be to the left of a vertex of I in an MNS ordering. However, forcing a vertex of I to be to the left of a vertex of C is more difficult. Therefore, we will call a vertex of I that is to the left of a vertex of C in a vertex ordering σ a *premature vertex of σ*. The neighbors of such a premature vertex must fulfill strong conditions on their positions in σ as the following lemma shows.

Lemma 25 (Beisegel et al. [1, Lemma 22])**.** *Let G be a connected split graph with split partition (C, I). Let σ be an MNS ordering of G. If the vertex $x \in I$ is a premature vertex of σ, then:*

(i) every vertex of C that is to the left of x in σ is a neighbor of x, and

(ii) every non-neighbor of x that is to the right of x in σ is also to the right of each neighbor of x in σ.

Similar to linear orderings, we call a vertex $x \in I$ a *premature vertex of partial order* π if there is an element $y \in C$ with $x \prec_\pi y$. To decide whether a partial order π can be extended by an MNS ordering, the set of premature vertices of π must fulfill strong properties which we define in the following.

Definition 26. Let G be a connected split graph with split partition (C, I). Let π be a partial order on $V(G)$ and let A be a subset of I. The tuple (π, A) fulfills the *nested property* if the following conditions hold:

(N1) If $y \in C$ and $x \prec_\pi y$, then $x \in C \cup A$, i.e., the premature vertices of π are in A.

(N2) The neighborhoods of the elements of A can be ordered by inclusion, i.e., there are pairwise disjoint sets $C_1, I_1, C_2, I_2, \ldots, C_k, I_k$ with $\bigcup_{j=1}^{k} I_j = A$ and for all $i \in \{1, \ldots, k\}$ and all $x \in I_i$ it holds that $N_G(x) = \bigcup_{j=1}^{i} C_j$.

(N3) If $y \in C_i \cup I_i$ and $x \prec_\pi y$, then $x \in C_j \cup I_j$ with $j \leqslant i$.

(N4) For all $i \in \{1, \ldots, k\}$, there is at most one vertex $x \in I_i$ for which there is a vertex $y \in C_i$ with $x \prec_\pi y$.

The *nested relation* $\mathcal{N}(\pi, A)$ is defined as the relation containing the following tuples:

(P1) (x, y) $\quad \forall x, y \in V(G)$ with $x \prec_\pi y$

(P2) (x, y) $\quad \forall x \in I_i \cup C_i, y \in V(G) \setminus \bigcup_{j=1}^{i}(I_j \cup C_j)$

(P3) (x, y) $\quad \forall x \in C, y \in I \setminus A$

(P4) (x, y) $\quad \forall x, y \in I_i$ for which $\exists z \in C_i$ with $x \prec_\pi z$

In the following we will describe how we can check whether a tuple (π, A) fulfills the nested property. The following lemma is a key ingredient of this procedure.

Lemma 27 (Beisegel et al. [2, proof of Proposition 14])**.** *Given a set X and a family $\mathcal{Q} \subseteq \mathcal{P}(X)$ of subsets of X, we can check in $\mathcal{O}(|X| + \sum_{A \in \mathcal{Q}} |A|)$ time whether the elements of $\mathcal{Q}$ can be ordered by inclusion.*

Now we show that checking for the nested property can be done in linear time. Furthermore, $\mathcal{N}(\pi, A)$ has a linear extension if (π, A) has the nested property and we can find such a linear extension also in linear time.

Lemma 28. *Given a connected split graph G with split partition (C, I), a partial order π on $V(G)$ and a set $A \subseteq I$, the relation $\mathcal{N}(\pi, A)$ has a linear extension if (π, A) fulfills the nested property.*

Furthermore, we can decide whether (π, A) fulfills the nested property and, if so, then we can compute a linear extension of $\mathcal{N}(\pi, A)$ in $\mathcal{O}(n(G) + m(G) + |\pi|)$ time.

Proof. We start by showing that we can check for the nested property in linear time. We first find a split partition of G in linear time (see, e.g., [23]). Then we mark for every vertex whether it is in $I \setminus A$, in C, or in A. This can be done in $\mathcal{O}(n(G))$ time. To check condition (N1), we iterate through π and check for every tuple $(x, y) \in \pi$ with $x \in I$ and $y \in C$ whether $x \in A$. This needs $\mathcal{O}(|\pi|)$ time. To check condition (N2), we use the approach of Lemma 27. During this process, we can label every vertex in $A \cup N_G(A)$ with the index of its I-set or C-set and all other vertices with the label ∞. To check condition (N3) and (N4), we iterate through π a second time. Now we check whether for all $(x, y) \in \pi$ it holds that the label of x is smaller or equal to the label of y and whether there is at most one vertex x in each set I_i for which there exists a vertex y in C_i with $x \prec_\pi y$. Therefore, we can check whether (π, A) fulfills the nested property in $\mathcal{O}(n(G) + m(G) + |\pi|)$ time.

Next we show how we can find a linear extension of $\mathcal{N}(\pi, A)$ if (π, A) fulfills the nested property. Note that we cannot create $\mathcal{N}(\pi, A)$ in linear time as it might have super-linear size.[4] For every $i \in \{1, \ldots, k\}$, we construct the suborders of π containing the elements of $I_i \cup C_i$. We do the same for $C \setminus N_G(A)$ as well as $I \setminus A$, respectively. We can find these suborders in $\mathcal{O}(|\pi|)$ time by iterating though π and putting the tuples in their respective suborder. Now we order the sets $I_i \cup C_i$ as follows. For each $x \in I_i$, let $S_i(x) = \{y \in C_i \mid x \prec_\pi y\}$. All these sets can be computed in total time $\mathcal{O}(|\pi|)$. Since (π, A) fulfills (N4), there is at most one vertex $x' \in I_i$ with $S_i(x') \neq \emptyset$. We start with a linear extension of the suborder of π containing the vertices in $C_i \setminus S_i(x')$, then we take x' (if it exists), then a linear extension of $S_i(x')$ and then a linear extension of the remaining vertices of I_i. Note that this is a linear extension of the suborder π restricted to $C_i \cup I_i$ since none of the vertices in $I_i \setminus \{x'\}$ can be to the left of x' or a vertex of C_i in π, due to condition (N4).

Now we build a linear ordering σ of $V(G)$ as follows. We concatenate the constructed orderings of the sets $I_i \cup C_i$ starting with $I_1 \cup C_1$ and ending with $I_k \cup C_k$. Then we add a linear extension of π restricted to the vertices of $C \setminus N_G(A)$ followed by a linear extension of the vertices of $I \setminus A$. Due to conditions (N1) and (N3), this linear ordering is a linear extension of π. It is straightforward to check that the ordering σ also fulfills the constraints (P1) till (P4) and, thus, σ is a linear extension of $\mathcal{N}(\pi, A)$. □

Now we consider MNS orderings extending a partial order on a split graph's vertex set. We first show that for the set A of the premature vertices of a partial order π, the tuple

[4]Consider a split graph with n vertices in the clique and n^2 vertices in the independent set which all have only one neighbor in the clique. If the partial order forces only one vertex of the independent set to be before one vertex of the clique, then the nested relation has $\Omega(n^3)$ many elements while the size of the input is bounded by $\mathcal{O}(n^2)$.

(π, A) must necessarily fulfill the nested property if there is an MNS ordering extending π. Furthermore, every such MNS ordering fulfills a large subset of the constraints given by the nested relation $\mathcal{N}(\pi, A)$.

Lemma 29. *Let G be a connected split graph with split partition (C, I) and let π be a partial order on $V(G)$. Let $A = \{v \in I \mid \exists w \in C \text{ with } v \prec_\pi w\}$. If there is an MNS ordering σ of G that extends π, then (π, A) fulfills the nested property. If $x \prec_\sigma y$ but $(y, x) \in \mathcal{N}(\pi, A)$, then $x \notin A \cup C$.*

Proof. Due to the definition of A, the property (N1) trivially holds true. Let σ be an MNS ordering of G that extends π.

Claim 30. *If $x \in I$, $y \in A$ and $x \prec_\sigma y$, then $N_G(x) \subseteq N_G(y)$.*

Proof of Claim. Since $y \in A$, there is a vertex $c \in C$ with $y \prec_\pi c$. As σ is a linear extension of π, it also holds that $x \prec_\sigma y \prec_\sigma c$. Since both x and y are in I, Lemma 25 implies that every neighbor of x is to the left of y in σ and all these vertices are also neighbors of y. Hence, $N_G(x) \subseteq N_G(y)$. ◀

This claim implies that (N2) holds. Thus, we may assume in the following that the sets $C_1, \dots, C_k$ and $I_1, \dots, I_k$ defined in (N2) exist.

Claim 31. *If $y \in I_i$ and $x \prec_\sigma y$, then either $x \in C_j \cup I_j$ for some $j \leqslant i$ or $x \not\prec_\pi y$ and $x \notin A \cup C$.*

Proof of Claim. As $y \in A$, there is a vertex $c \in C$ with $y \prec_\pi c$ and, thus, $y \prec_\sigma c$. If $x \in C$, then it follows from Lemma 25 that $x \in N_G(y)$ and, therefore, $x \in C_j$ for some $j \leqslant i$. Thus, we may assume that $x \in I$. Due to Claim 30, it holds that $N_G(x) \subseteq N_G(y)$.

If $x \in A$, then $x \in I_j$ for some $j \leqslant i$ since (N2) holds. If $x \prec_\pi y$, then the transitivity of π implies that $x \prec_\pi c$ and, thus, x is an element of A. Thus, in any case Claim 31 holds. ◀

This claim implies that (N3) holds for every y in every I_i since $x \prec_\pi y$ implies that $x \prec_\sigma y$.

Claim 32. *If $y \in C_i$ and $x \prec_\sigma y$, then either $x \in C_j \cup I_j$ for some $j \leqslant i$ or $x \not\prec_\pi y$ and $x \notin A \cup C$.*

Proof of Claim. First assume that $x \in C$. If there is some vertex $z \in I_i$ with $x \prec_\sigma z$, then Claim 31 implies that $x \in C_j$ for some $j \leqslant i$. Thus, we may assume that there is a vertex $z \in I_i$ with $z \prec_\sigma x$. As $y \in N_G(z)$, it follows from Lemma 25 that x is also in $N_G(z)$ and, therefore, x is in some C_j with $j \leqslant i$.

Now assume that $x \in I$. All vertices of I_i are adjacent to y but none of them is adjacent to x. Thus, by Lemma 25, no vertex of I_i is to the left of x in σ since, otherwise, x would lie between this vertex and one of its neighbors in σ. Hence, if $x \notin I_i$, then there is a vertex $z \in I_i$ with $x \prec_\sigma z$. By Claim 31, $x \in I_j$ for some $j \leqslant i$ or $x \notin A$. If $x \notin A$, then $x \not\prec_\pi y$, due to the definition of A. ◀

Similar to Claim 31, Claim 32 implies that (N3) holds for every element of every C_i. Therefore, we have proven that (N3) holds in general. Statement (N4) follows from the following claim.

Claim 33. *If $x, y \in I_i$, $z \in C_i$ and $y \prec_\sigma z$, then $z \prec_\sigma x$.*

Proof of Claim. Both x and y are adjacent to z. Lemma 25 implies that y cannot be between x and z and x cannot be between y and z. Thus, x is to the right of z in σ. ◀

Finally assume that there are vertices $x, y \in V(G)$ with $x \prec_\sigma y$ but $(y, x) \in \mathcal{N}(\pi, A)$. We have to prove that $x \notin A \cup C$. Note that the tuple (y, x) cannot be a tuple of π since σ is a linear extension of π. Thus, (y, x) is not a tuple of type (P1).

We consider two cases. If $y \notin A \cup N_G(A)$, then the tuple (y, x) is a tuple of kind (P3) and, thus, x is an element of $I \setminus A$. Therefore, it holds that $x \notin A \cup C$.

Next assume that $y \in A \cup N_G(A)$. Then $y \in C_i \cup I_i$ for some i. Assume for contradiction that $x \in A \cup C$. It follows from Claims 31 and 32 that $x \in C_j \cup I_j$ for some $j \leqslant i$. Therefore, (y, x) is not a tuple of type (P2) or (P3) but is of type (P4). This implies that $x, y \in I_i$ and there is a $z \in C_i$ with $y \prec_\pi z$. Since σ extends π, it holds that $y \prec_\sigma z$. Claim 33 implies that $z \prec_\sigma x$ and, thus, $y \prec_\sigma x$; a contradiction. Hence, $x \notin A \cup C$. □

The nested property is, in a restricted way, also sufficient for the existence of an MNS ordering extending π. We show that if (π, A) fulfills the nested property, then there is an MNS ordering that fulfills all tuples of π that contain elements of the set A or the clique C. This ordering can be found using a +-search.

Lemma 34. *Let G be a connected split graph with split partition (C, I), let π be a partial order on $V(G)$ and let A be a subset of I. Assume (π, A) fulfills the nested property. Then, for each linear extension ρ of $\mathcal{N}(\pi, A)$ and each graph search $\mathcal{A} \in$ {LBFS, LDFS, MCS, MNS}, the ordering $\sigma = \mathcal{A}^+(\rho)$ of G fulfills the following property: If $(x, y) \in \mathcal{N}(\pi, A)$, then $x \prec_\sigma y$ or both x and y are not in $A \cup C$.*

Proof. Assume for contradiction that σ does not fulfill this property. Let x be the σ-leftmost vertex such that there is a vertex $y \in V(G)$ with $x \prec_\sigma y$, $(y, x) \in \mathcal{N}(\pi, A)$ and at least one of x and y are in $A \cup C$. Since ρ is a linear extension of $\mathcal{N}(\pi, A)$, it holds that $y \prec_\rho x$. Due to Lemma 3, there is a vertex $w \in V(G)$ with $w \prec_\sigma x$, $wx \in E(G)$ and $wy \notin E(G)$.

First assume that $y \in I$. If $x \notin A \cup C$, then $y \in A$ by the choice of x and y. So consider the case that $x \in A \cup C$. If $(y, x) \in \pi$, then properties (N1) and (N3) imply that $y \in A$. If $(y, x) \notin \pi$, then (y, x) is of type (P2) or (P4) and, thus, $y \in A$.

Therefore, in any case, it holds that $y \in A$. Let I_i be the set containing y. Due to the choice of x, it holds that $(y, w) \notin \mathcal{N}(\pi, A)$. It follows from (P2) that $w \in \bigcup_{j=1}^{i} I_j \cup C_i$. Since $wy \notin E(G)$, it holds that $w \in \bigcup_{j=1}^{i} I_j$. Therefore, $x \in \bigcup_{j=1}^{i} C_j$ because $wx \in E(G)$. As $(x, y) \notin \mathcal{N}(\pi, A)$, it follows from (P2) that $x \in C_i$ and, hence, $w \in I_i$. This implies that (y, x) can only be an element of $\mathcal{N}(\pi, A)$ if it is an element of π. However, then, due to (P4), $(y, w) \in \mathcal{N}(\pi, A)$; a contradiction to the observation above.

So we may assume that $y \in C$. As $wx \in E(G)$ and $wy \notin E(G)$, it follows that $w \in I$ and $x \in C$. Since $N_G(w) \subseteq N_G[x]$ and $w \prec_\sigma x$, it follows from Lemma 3 that $w \prec_\rho x$ and, therefore, $(x, w) \notin \mathcal{N}(\pi, A)$. Due to (P3), it holds that $w \in A$. Thus, $w \in I_i$ for some $1 \leqslant i \leqslant k$. Then $x \in C_i$ since $wx \in E(G)$ and $(x, w) \notin \mathcal{N}(\pi, A)$. As $(y, x) \in \mathcal{N}(\pi, A)$, vertex y is in C_j with $j \leqslant i$, due to (N2) and (P2). Therefore, $wy \in E(G)$; a contradiction. □

Lemma 34 implies that the only interesting vertices that remain are the elements of $I \setminus A$. Consider an instance of the generic algorithm Label Search given in Algorithm 1. After this search has visited all the clique vertices of a split graph, the labels of the remaining vertices of I do not change anymore. Thus, a vertex x whose label is now smaller than the label of another vertex y will be visited after y. Therefore, it is not enough to consider only the premature vertices of π. Instead, we must also consider those vertices $x \in \mathcal{I}$ that are forced by π to be to the left of another vertex $y \in I$ whose label is larger than the label of x when all clique vertices have been visited. In the case of MCS, this is sufficient to characterize partial orders that are extendable.

Theorem 35. *Let G be a connected split graph with split partition (C, I). Let π be a partial order on $V(G)$. Let $A := \{u \in I \mid \exists v \in V(G) \text{ with } v \in C \text{ or } d_G(u) < d_G(v) \text{ such that } u \prec_\pi v\}$. Then the following statements are equivalent.*

(i) There is an MCS ordering which is a linear extension of π.

(ii) The tuple (π, A) fulfills the nested property.

(iii) There is a linear extension of $\mathcal{N}(\pi, A)$ and for each linear extension ρ of $\mathcal{N}(\pi, A)$, the ordering MCS$^+(\rho)$ is a linear extension of π.

Proof. First we show that (i) implies (ii). To this end, we create the relation π^* by adding the following tuples to π: If $u, v \in I$, $u \prec_\pi v$ and $d_G(u) < d_G(v)$, then $(u, w) \in \pi^*$ where w is some element of $C \setminus N_G(u)$. Note that such a vertex w exists because v has at least one neighbor in C that u does not have. We claim that every MCS ordering σ that extends π is also a linear extension of π^*.

Assume for contradiction that the tuple (u, w) in $\pi^* \setminus \pi$ is not fulfilled by σ, i.e., $w \prec_\sigma u$. As $w \in C$ and $w \notin N_G(u)$, it follows from Lemma 25 that all vertices of C are to the left of u in σ. Since σ is a linear extension of π, it holds that $u \prec_\sigma v$. However, the number of visited neighbors of u has been strictly smaller than the number of visited neighbors of v when u was visited by MCS; a contradiction.

Therefore, σ is a linear extension of π^* and the reflexive and transitive closure π^T of π^* is a partial order. Note that a vertex $x \in I$ is an element of A if and only if there is a vertex $y \in C$ with $x \prec_{\pi^T} y$. Thus, by Lemma 29, the tuple (π^T, A) fulfills the nested property. Since π is a subset of π^T, the tuple (π, A) also fulfills the nested property.

Next we show that (ii) implies (iii). Assume that (π, A) fulfills the nested property. Due to Lemma 28, there is a linear extension ρ of $\mathcal{N}(\pi, A)$. Let σ be the MCS$^+(\rho)$ ordering. Then, by Lemma 34, for every $x \prec_\pi y$ it holds $x \prec_\sigma y$ if $\{x, y\} \cap (A \cup C) \neq \emptyset$. It remains to show that $x \prec_\sigma y$ also holds for every pair $x, y \notin A \cup C$ with $x \prec_\pi y$. As both x and y are in I but x is not in A, it holds $d_G(x) \geqslant d_G(y)$. Due to (P3) and Lemma 34, for each vertex $z \in C$ it holds that $z \prec_\sigma x$ and $z \prec_\sigma y$. Thus, the MCS label of x is at least as large as the MCS label of y when one of the two vertices is visited by MCS. Since $x \prec_\rho y$, vertex x is to the left of y in σ. Therefore, σ is a linear extension of π.

Statement (iii) trivially implies (i). □

Due to Lemma 28, we can check in linear time whether (π, A) fulfills the nested property. Therefore, Theorem 35 lead directly to a linear-time algorithm for the PSOP of MCS on split graphs. To find a certificate for yes-instances, we can use MCS$^+$. Unfortunately, there is no general linear-time implementation of MCS$^+$ known so far. Nevertheless, we can implement MCS$^+$ with linear running time on split graphs as the following two results show.

Lemma 36. *Given a graph G and a vertex ordering ρ of G, we can transform G in linear time such that the adjacency list of every vertex of G is ordered due to ρ.*

Proof. The idea is to build new adjacency lists for every vertex. We start with empty lists and iterate through ρ starting from the left. If we visit a vertex v, then we iterate through its old adjacency list and add v at the end of the new adjacency list of every neighbor of v. Thus, if $u \prec_\rho w$, then u is to the left of w in every adjacency list that contains both u and w. Clearly, this procedure only takes linear time as we iterate through every old adjacency list exactly once. □

Theorem 37. *Given a connected split graph G and a vertex ordering ρ of G, the MCS$^+(\rho)$ ordering of G can be found in $\mathcal{O}(n(G) + m(G))$ time.*

Proof. We find a split partition (C, I) of G in linear time (see, e.g., [23]) and mark every vertex as C-vertex or I-vertex. Furthermore, we sort the adjacency list of every vertex due to ρ. By Lemma 36, this can be done in linear time. Now we construct σ starting from the empty ordering. We iterate through ρ. As long as the next vertex is unvisited and in C, we append it at the end of σ. We call these vertices *normal C-vertices* in the following. If the next vertex v is in I, then we check whether all vertices of C that have been visited so far are neighbors of v. We can do this in $\mathcal{O}(d_G(v))$ time if we keep track of the count of the visited C-vertices during the process. If not all visited vertices of C are neighbors of v, then we jump to the successor of v in ρ. Otherwise, we append v at the end of σ. Following the terminology used so far, we call vertex v a *premature vertex* in this case. Then we iterate through the neighborhood of v and append all the unvisited neighbors of v to the end of σ following their sorting in the adjacency list of v. Afterwards, we jump to the successor of v in ρ. We iterate this process until we reach the end of ρ. It only remains to order the unvisited vertices of I sufficiently. We order them non-increasingly by their degree such that vertices with the same degree are ordered by ρ. This can straightforwardly be done in $\mathcal{O}(n(G) + m(G))$ time using counting sort. We append this ordering to σ.

We claim that σ is the MCS$^+(\rho)$ ordering of G. Let v be a *normal* C-vertex. If any vertex x with $v \prec_\sigma x$ has a neighbor z with $z \prec_\sigma v$, then z is a neighbor of v. This follows from the fact that the neighbors of every premature vertex were visited directly after the premature vertex. Therefore, v has the largest MCS label, when v was appended to σ. Furthermore, all the C-vertices to the left of v in ρ were already visited at this point. All unvisited I-vertices u to the left of v are not adjacent to at least one visited vertex of C. Thus, v is the ρ-leftmost vertex with maximal MCS label. The same argumentation also works if v is a premature I-vertex.

If v is not a normal C-vertex, then it follows from Lemma 25 that v has to follow its premature neighbor. For every vertex $u \in I$ that was not visited during the first iteration of ρ, there was a C-vertex to its left that was not its neighbor. Due to Lemma 25, u could not be a premature vertex of σ. Thus, the MCS$^+(\rho)$ ordering has to end with these vertices. The final ordering of the unvisited I-vertices ensures that σ is the MCS$^+(\rho)$ ordering of G. □

Combining Theorems 35 and 37, we can give a linear-time algorithm for the PSOP of MCS on split graphs that returns a certificate in the positive case.

Theorem 38. *Given a connected split graph G and a partial order π on $V(G)$, the PSOP of MCS can be solved in $\mathcal{O}(n(G) + m(G) + |\pi|)$ time. If the instance is a yes-instance, then an MCS ordering σ of G that extends π can be computed in $\mathcal{O}(n(G) + m(G) + |\pi|)$ time.*

Proof. First we compute the set A defined in Theorem 35. We iterate through all elements $(u, v) \in \pi$ and if $u \in I$, then we check whether $v \in C$ or $d_G(u) < d_G(v)$. This can be done in $\mathcal{O}(|\pi|)$ time. By Theorem 35, the instance is a yes-instance of the PSOP if and only if (π, A) fulfills the nested property. We can check this in $\mathcal{O}(n(G) + m(G) + |\pi|)$ time, due to Lemma 28.

It follows from the same lemma that we can compute a linear extension ρ of $\mathcal{N}(\pi, A)$ in $\mathcal{O}(n(G) + m(G) + |\pi|)$ time if (π, A) fulfills the nested property. To compute an MCS ordering extending π, we only have to compute the MCS$^+(\rho)$ ordering of G, due to Theorem 35. By Theorem 37, we can do this in $\mathcal{O}(n(G) + m(G))$ time. □

This result is a generalization of the linear-time algorithms for the End-Vertex Problem [2] and the $\mathcal{F}$-Tree Recognition Problem [3] of MCS on split graphs. Note that Rong et al. [32] presented a polynomial-time algorithm for PSOP on chordal graphs, a superclass of split graphs. However, this algorithm has no linear running time.

For LBFS there is a characterization of extendable partial orders that is similar to Theorem 35. However, due to the more complex label structure of LBFS, the result is slightly more complicated and uses OBA orderings.

Theorem 39. *Let G be a connected split graph with split partition (C, I). Let π be a partial order on $V(G)$ and let $A := \{u \in I \mid \exists v \in V(G)$ with $v \in C$ or $N_G(u) \subsetneq N_G(v)$ such that $u \prec_\pi v\}$. We define the following relation $\mathcal{R} \subseteq \mathcal{P}(C) \times \mathcal{P}(C)$:*

$$\begin{aligned}\mathcal{R} = \{(X,Y) \mid \exists x, y \in I \setminus A \text{ with } X = N_G(x) \setminus N_G(y),\ Y = N_G(y) \setminus N_G(x),\ x \prec_\pi y\}\\ \cup \{(\{x\},\{y\}) \mid x, y \in A \cup C,\ (x,y) \in \mathcal{N}(\pi, A)\}.\end{aligned}$$

Let $\mathcal{Q}$ be the ground set of $\mathcal{R}$. Then the following statements are equivalent.

(i) There is an LBFS ordering which is a linear extension of π.

(ii) The tuple (π, A) fulfills the nested property and there is an OBA ordering for $(A \cup C, \mathcal{Q}, \mathcal{R})$.

(iii) There is a linear extension of $\mathcal{N}(\pi, A)$ that contains an OBA ordering for $(A \cup C, \mathcal{Q}, \mathcal{R})$ and for each of those linear extensions ρ, the ordering LBFS$^+$(ρ) is a linear extension of π.

Proof. We start by showing that (i) implies (ii). Assume that the LBFS ordering σ is a linear extension of π. First we show that (π, A) fulfills the nested property in a similar way we have done it in the proof of Theorem 35. We create the relation π^* by adding the following tuples to π: If $u, v \in I$, $u \prec_\pi v$ and $N_G(u) \subsetneq N_G(v)$, then $(u, w) \in \pi^*$ where w is some vertex in $N_G(v) \setminus N_G(u)$. We claim that σ is also a linear extension of π^*. Assume for contradiction that the tuple (u, w) in $\pi^* \setminus \pi$ is not fulfilled by σ, i.e., $w \prec_\sigma u$. As $w \in C$ and $w \notin N_G(u)$, it follows from Lemma 25 that all vertices of C are to the left of u in σ. Since σ is a linear extension of π, it holds that $u \prec_\sigma v$. Therefore, (w, u, v) forms a curious triple of σ. Since σ is an LBFS ordering, Lemma 2 (iii) implies that there is a vertex $z \prec_\sigma w$ with $zu \in E(G)$ but $zv \notin E(G)$; a contradiction to the fact that $N_G(u) \subsetneq N_G(v)$.

Therefore, σ is a linear extension of π^* and the reflexive and transitive closure π^T of π^* is a partial order. Note that a vertex $x \in I$ is an element of A if and only if there is a vertex $y \in C$ with $x \prec_{\pi^T} y$. Thus, by Lemma 29 it follows that (π^T, A) fulfills the nested property. Since π is a subset of π^T, the tuple (π, A) also fulfills the nested property.

To show the existence of an OBA ordering for $(A \cup C, \mathcal{Q}, \mathcal{R})$, we prove that the ordering of the vertices of $A \cup C$ in σ forms such an OBA ordering. Assume for contradiction that there is a tuple $(X, Y) \in \mathcal{R}$ that is not fulfilled by σ, i.e., there is a vertex $b \in Y$ that is to the left of every vertex of X in σ. For every $u, v \in A \cup C$ with $(u, v) \in \mathcal{N}(\pi, A)$ it follows from Lemma 29 that $u \prec_\sigma v$. Therefore, $X = N_G(x) \setminus N_G(y)$ and $Y = N_G(y) \setminus N_G(x)$ for some $x, y \in I \setminus A$ with $x \prec_\pi y$. It holds that $x \prec_\sigma y$ because σ is a linear extension of π. Furthermore, $by \in E(G)$ and $bx \notin E(G)$. Since no vertex of X is to the left of b, for every vertex $z \prec_\sigma b$ the edge yz is in $E(G)$ or the edge xz is not in $E(G)$. If $b \prec_\sigma x$, then (b, x, y) is a curious triple of σ that does not fulfill the 4-point condition of LBFS given in Lemma 2 (iii). Thus, we may assume that $x \prec_\sigma b$. Since $x \notin A$, the set $N_G(x)$ is not a strict subset of $N_G(y)$. Furthermore, $b \in N_G(y) \setminus N_G(x)$ and, hence, it holds $N_G(x) \neq N_G(y)$. Therefore, X is not empty. Let $a \in X$. By the choice of b, it holds $x \prec_\sigma b \prec_\sigma a$. This, however, contradicts Lemma 25 as $x \in I$, $b \in C$, $bx \notin E(G)$ but $ax \in E(G)$.

Now we prove that (ii) implies (iii). Assume that the two conditions of (ii) are fulfilled and let τ be an OBA ordering for the input $(A \cup C, \mathcal{Q}, \mathcal{R})$. Let λ be a linear extension of the suborder of π restricted to $I \setminus A$. The ordering $\tau \oplus \lambda$ forms a linear extension of $\mathcal{N}(\pi, A)$ since all the constraints within $A \cup C$ and within $I \setminus A$ are fulfilled by τ and λ, respectively, and the constraints between these two sets always force the vertex of $A \cup C$ to be to the left of the vertex of $I \setminus A$. Thus, there is a linear extension of $\mathcal{N}(\pi, A)$ that contains an OBA ordering of $(A \cup C, \mathcal{Q}, \mathcal{R})$.

Now let ρ be an arbitrary linear extension of $\mathcal{N}(\pi, A)$ that contains an OBA ordering of $(A \cup C, \mathcal{Q}, \mathcal{R})$. Let σ be the LBFS$^+(\rho)$ ordering of G and let x and y be two vertices with $x \prec_\pi y$. Due to Lemma 34, if any of x or y is in $A \cup C$, then $x \prec_\sigma y$. Thus, we may assume that both x and y are in $I \setminus A$. Assume for contradiction that $y \prec_\sigma x$. Since ρ is a linear extension of π, it holds that $x \prec_\rho y$ and, thus, it follows from Lemma 3 that $N_G(y) \setminus N_G(x)$ is not empty. Since $x \notin A$ and $N_G(x) \neq N_G(y)$, it holds that $N_G(x) \not\subseteq N_G(y)$. Therefore, the set $N_G(x) \setminus N_G(y)$ is also not empty. Due to the construction of $\mathcal{R}$ and ρ, there is a vertex $z \in N_G(x) \setminus N_G(y)$ that is to the left of every element of $N_G(y) \setminus N_G(x)$ in ρ. Note that $z \in C$ and, thus, $(z, y) \in \mathcal{N}(\pi, A)$ since $y \in I \setminus A$. Lemma 34 implies that $z \prec_\sigma y$. Therefore, the tuple (z, y, x) is a curious triple of σ. As σ is an LBFS ordering, there is a vertex $w \in N_G(y) \setminus N_G(x)$ with $w \prec_\sigma z$ (see Lemma 2 (iii)). By the choice of z it holds $z \prec_\rho w$. It follows from Lemma 3 that there is a vertex u with $u \prec_\sigma w$, $uw \in E(G)$ and $uz \notin E(G)$. Since both z and w are elements of C, the vertex u is an element of I. If $u \notin A$, then $(w, u) \in \mathcal{N}(\pi, A)$. This contradicts Lemma 34 since $u \prec_\sigma w$ and $w \in C$. Therefore, we may assume that $u \in A$. However, this means that w is an element of some C_i (since $uw \in E(G)$) and z is not an element of any C_j with $j \leqslant i$ (since $uz \notin E(G)$). Thus, $(w, z) \in \mathcal{N}(\pi, A)$; a contradiction to the facts that $z \prec_\rho w$ and that ρ is a linear extension of $\mathcal{N}(\pi, A)$. Statement (iii) trivially implies (i). $\square$

Again, this characterization leads to an efficient algorithm for the PSOP of LBFS on split graphs.

Theorem 40. *Given a connected split graph G and a partial order π on $V(G)$, the PSOP of LBFS can be solved in $\mathcal{O}(n(G) \cdot |\pi|)$ time. If the instance is a yes-instance, then an LBFS ordering σ of G that extends π can be computed in $\mathcal{O}(n(G) \cdot |\pi|)$ time.*

Proof. First note that $n(G) \leqslant |\pi| \leqslant n(G)^2$ as π contains all reflexive tuples. We can compute the set A defined in Theorem 39 in $\mathcal{O}(n(G) \cdot |\pi|)$ time by iterating through the tuples of π and comparing the neighborhoods of the two vertices in $\mathcal{O}(n(G))$ time. We can check whether (π, A) fulfills the nested property in $\mathcal{O}(n(G) + m(G) + |\pi|)$ time, due to Lemma 28. The nested relation $\mathcal{N}(\pi, A)$ can be computed in $\mathcal{O}(n(G)^2)$ time. We extend the relation $\mathcal{R}$ defined in Theorem 39 by the set $S = \{(\{x\}, \{y\}) \mid x, y \in V(G),\ (x, y) \in \mathcal{N}(\pi, A)\}$ and get the relation $\mathcal{R}'$. The size of S is bounded by $\mathcal{O}(n(G)^2) \subseteq \mathcal{O}(n(G) \cdot |\pi|)$. The size of $\mathcal{R}' \setminus S$ is bounded by $\mathcal{O}(|\pi|)$ since we have inserted at most one tuple to $\mathcal{R}$ for every element of π. Since the size of the elements of the ground set of $\mathcal{R}'$ is bounded by $n(G)$, the total size of $\mathcal{R}'$, its ground set $\mathcal{Q}$ and the elements of $\mathcal{Q}$ is bounded by $\mathcal{O}(n(G) \cdot |\pi|)$. We compute an OBA ordering for the input $(V(G), \mathcal{R}', \mathcal{Q})$ in $\mathcal{O}(n(G) \cdot |\pi|)$ time using Algorithm 2 (see Theorem 11). Note that this ordering is also a linear extension of $\mathcal{N}(\pi, A)$, due to our addition of the tuples of S. By Theorem 39, there is an LBFS ordering extending π if and only if such an OBA ordering ρ exists and we find such an ordering by computing the LBFS$^+(\rho)$ ordering. This can be done in linear time (see Theorem 4). The algorithm has a total running time of $\mathcal{O}(n(G) \cdot |\pi|)$. $\square$

In contrast to the algorithm for MCS the running time of the algorithm for LBFS is not linear in the input size. However, the End-Vertex Problem and the $\mathcal{F}$-Tree Recognition Problem of LBFS can be solved in linear time on split graphs [2, 3].

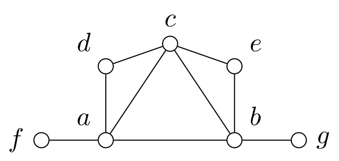

Figure 2: A split graph consisting of clique $\{a, b, c\}$ and independent set $\{d, e, f, g\}$. Let π be the reflexive and transitive closure of the relation $\{(f, e), (g, d)\}$. There is no MCS ordering extending π since the set A defined in Theorem 35 contains both f and g and, thus, (π, A) does not fulfill the nested property. There is neither an LBFS ordering extending π since there is no OBA ordering for the relation $\mathcal{R} = \{(\{a\}, \{b, c\}),\ (\{b\}, \{a, c\})\}$ defined in Theorem 39. However, the LDFS ordering (f, a, c, b, e, g, d) extends π.

Unfortunately, the ideas of Theorems 35 and 39 cannot be directly adapted to the PSOP of LDFS and MNS. A main difficulty of these problems seems to be the identification of those vertices in the independent that have to be premature vertices. To illustrate this, we consider the example given in Figure 2. The defined partial order π has no premature vertices. Furthermore, the set A defined in Theorem 39 is empty for π. Nevertheless, for every MNS ordering σ extending π, one of the vertices f or g has to be a premature vertex of σ.

However, we can modify Theorem 39 to get a characterization of partial orders on a split graph's vertex set that can be extended by a BFS ordering. In difference to Theorems 35 and 39, we fix the start vertex of the BFS ordering. We start with the following simple observation.

Observation 41. *Let G be a connected split graph with split partition (C, I) and let σ be a connected search ordering of G starting with vertex s. The σ-leftmost neighbor of every vertex $v \in V(G) \setminus \{s\}$ is vertex s or an element of C.*

Now we present the characterization of extendable partial BFS orders.

Theorem 42. *Let G be a connected split graph with split partition (C, I) and let π be a partial order on $V(G)$. Let s be a vertex in $V(G)$. Then let $\mathcal{R}$ be the following relation:*

$$\begin{aligned}\mathcal{R} = \{(X, Y) \mid \exists x, y \in V(G) \text{ with } X = N_G(x),\ Y = N_G(y) \setminus N_G(x),\ x \prec_\pi y\} \\ \cup \{(\{x\}, \{y\}) \mid x, y \in V(G),\ x \prec_\pi y\} \\ \cup \{(\{x\}, \{y\}) \mid x \in N_G[s],\ y \in V(G) \setminus N_G[s]\}.\end{aligned}$$

Let $\mathcal{Q}$ be the ground set of $\mathcal{R}$. Then the following statements are equivalent.

(i) There is a BFS ordering of G starting with s which is a linear extension of π.

(ii) There is an OBA ordering for $(V(G), \mathcal{Q}, \mathcal{R})$ starting with s.

(iii) There is an OBA ordering ρ for $(V(G), \mathcal{Q}, \mathcal{R})$ starting with s and for each such ordering the BFS$^+(\rho)$ ordering is a linear extension of π.

Proof. We start by showing that (i) implies (ii). Let σ be a BFS ordering of G starting with s that extends π. We claim that σ is an OBA ordering for the input $(V(G), \mathcal{Q}, \mathcal{R})$. Since σ extends π, all the conditions of the set in the second line of the definition of $\mathcal{R}$ are fulfilled. As σ is a BFS ordering starting with s, the conditions of the set in the third line of the definition of $\mathcal{R}$ are also fulfilled. Thus, it remains to show that the conditions of the set in the first line are fulfilled. Let $x, y \in V(G)$, with $x \prec_\pi y$. Then $x \prec_\sigma y$ because σ is a linear extension of π. Let z be the σ-leftmost vertex that is adjacent to at least one of the vertices in $\{x, y\}$. Since σ is a BFS ordering, z is adjacent to x, due to the 4-point condition of BFS given in Lemma 2 (ii). Thus, the tuple $(X, Y) = (N_G(x),\ N_G(y) \backslash N_G(x))$ is fulfilled by σ.

To show (ii) $\Rightarrow$ (iii), consider an OBA ordering ρ for input $(V(G), \mathcal{Q}, \mathcal{R})$ starting with s. Let σ be the BFS$^+(\rho)$ ordering of G. It is obvious that σ is a BFS ordering starting with s. We claim that σ also extends π. Assume for contradiction that this is not the case, i.e., there are vertices x, y with $x \prec_\pi y$ but $y \prec_\sigma x$. As ρ is a linear extension of π, it holds that $x \prec_\rho y$. Since σ is the BFS$^+(\rho)$ ordering of G, the BFS label of y was larger than the BFS label of x when y was visited. Thus, the σ-leftmost neighbor z of the vertex set $\{x, y\}$ is a neighbor of y but not a neighbor of x. If z is in I, then z is equal to s, due to Observation 41. However, then the tuple $(\{y\}, \{x\})$ is in $\mathcal{R}$ which would imply that $y \prec_\rho x$; a contradiction. Thus, we may assume that $z \in C$. As $xz \notin E(G)$, vertex x is in I. Since $x \prec_\pi y$, there is the tuple $(N_G(x), N_G(y) \setminus N_G(x))$ in $\mathcal{R}$. As $z \in N_G(y) \setminus N_G(x)$ and ρ is an OBA ordering for $\mathcal{R}$, there is a vertex $w \in N_G(x)$ with $w \prec_\rho z$. Since $x \in I$, vertex w is in C. However, by the choice of z it holds that $z \prec_\sigma w$. This implies that the σ-leftmost neighbor of z is not a neighbor of w. As $w \in C$, the σ-leftmost neighbor of z is the start vertex s, due to Observation 41. Therefore, $(\{z\}, \{w\})$ is an element of $\mathcal{R}$ and $z \prec_\rho w$; a contradiction.

Statement (iii) trivially implies (i). $\square$

Theorem 43. *Given a connected split graph G, a partial order π on $V(G)$ and a start vertex $s \in V(G)$, the rooted PSOP of BFS can be solved in $\mathcal{O}(n(G) \cdot |\pi|)$ time. If the instance is a yes-instance, then a BFS ordering σ of G starting with s that extends π can be computed in $\mathcal{O}(n(G) \cdot |\pi|)$ time.*

Proof. We check whether the condition of Theorem 42 is fulfilled. To this end, we create the relation $\mathcal{R}$. The sets in the first and the second line of the definition of $\mathcal{R}$ contain at most two tuples for every element of π. The set in the third line contains at most $\mathcal{O}(n(G)^2)$ tuples. The sum of the sizes of the elements of $\mathcal{Q}$ can be bounded by $\mathcal{O}(n(G) \cdot |\pi|)$ since the $\mathcal{O}(n(G)^2)$ tuples only contain sets of size one and the other $\mathcal{O}(|\pi|)$ tuples contain sets of size $\mathcal{O}(n(G))$. Remember that $|\pi| \geqslant n(G)$. Thus, the overall size of the OBA instance is $\mathcal{O}(n(G) \cdot |\pi|)$ and we can solve the problem within the same time bound using Algorithm 2 (see Theorem 11). By Theorem 42, a respective linear extension can be computed using BFS$^+$ which can be implemented as linear-time algorithm (see Theorem 4). $\square$

7 Further Research

As we have seen in Section 5.2, there are graph classes where the PSOP of DFS, LDFS, and MCS is NP-hard while both the End-Vertex Problem and the $\mathcal{F}$-Tree Recognition Problem of these searches are easy. On the contrary, all three problems are polynomial for GS on general graphs (see Section 4.1). It remains open how the three problems behave for BFS, LBFS, and MNS, i.e., is there a graph class where the PSOP of one of these searches is NP-hard while the End-Vertex Problem and the $\mathcal{F}$-Tree Recognition Problem are easy.

Besides this, there are several interesting combinations of graph class and graph search where the complexity of the End-Vertex Problem, the $\mathcal{F}$-Tree Recognition Problem, or the PSOP is still open. The presumably most prominent case is the complexity of the End-Vertex Problem (and of the PSOP) of LBFS on chordal graphs. In essence, LBFS was originally designed exactly because of the properties of its end-vertices on chordal graphs. Thus, it is quite surprising that LBFS is the only prominent subsearch of MNS whose End-Vertex Problem is still open for chordal graphs.

Most results presented here focus on the complexity of the PSOP for restricted graph classes. In contrast, Section 4.2 considers the complexity of the PSOP when parameterized by the partial order's width. As shown, the PSOP of forgetful searches then lies in the complexity class XP. This result raises several interesting questions. First, is the problem also W[1]-hard – like other ordering problems parameterized by the width [5, 6] – or does it lie in the class FPT? What is the complexity of the PSOP of non-forgetful searches for partial orders of bounded width? And lastly, how does the restriction to other classes of partial orders influence the complexity of the PSOP? A class that might be interesting are *cs-trees* that have been considered in the context of another ordering problem [5].

The algorithms given in this paper use the complete partial order as input. Using a Hasse diagram, it is possible to encode a partial order more efficiently. Since there are partial orders of quadratic size where the Hasse diagram has only linear size (e.g. total orders), it could be fruitful to study the running time of the algorithms for instances of the PSOP where the partial order is given as Hasse diagram.

As mentioned in the introduction, the graph searches considered in this paper are used to solve several problems on graphs efficiently. This leads to the question whether the construction of a search ordering that extends a special partial order can be used in efficient algorithms for problems besides the End-Vertex Problem and the $\mathcal{F}$-Tree Recognition Problem.

References

[1] Jesse Beisegel, Carolin Denkert, Ekkehard Köhler, Matjaž Krnc, Nevena Pivač, Robert Scheffler, and Martin Strehler. Recognizing graph search trees. arXiv:1811.09249, 2018.

[2] Jesse Beisegel, Carolin Denkert, Ekkehard Köhler, Matjaž Krnc, Nevena Pivač, Robert Scheffler, and Martin Strehler. On the end-vertex problem of graph

searches. *Discrete Mathematics and Theoretical Computer Science*, 21(1), 2019. doi:10.23638/DMTCS-21-1-13.

[3] Jesse Beisegel, Carolin Denkert, Ekkehard Köhler, Matjaž Krnc, Nevena Pivač, Robert Scheffler, and Martin Strehler. Recognizing graph search trees. In *Proceedings of Lagos 2019, the tenth Latin and American Algorithms, Graphs and Optimization Symposium*, volume 346 of *ENTCS*, pages 99–110. Elsevier, 2019. doi:10.1016/j.entcs.2019.08.010.

[4] Jesse Beisegel, Carolin Denkert, Ekkehard Köhler, Matjaž Krnc, Nevena Pivač, Robert Scheffler, and Martin Strehler. The recognition problem of graph search trees. *SIAM Journal on Discrete Mathematics*, 35(2):1418–1446, 2021. doi:10.1137/20M1313301.

[5] Jesse Beisegel, Ekkehard Köhler, Fabienne Ratajczak, Robert Scheffler, and Martin Strehler. Graph search trees and the intermezzo problem. In Rastislav Královic and Antonín Kucera, editors, *49th International Symposium on Mathematical Foundations of Computer Science, MFCS 2024*, volume 306 of *LIPIcs*, pages 22:1–22:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.MFCS.2024.22.

[6] Jesse Beisegel, Fabienne Ratajczak, and Robert Scheffler. Computing Hamiltonian paths with partial order restrictions. *ACM Transactions on Computation Theory*, 17(1), 2025. doi:10.1145/3711844.

[7] Anne Berry, Jean R. S. Blair, Pinar Heggernes, and Barry W. Peyton. Maximum cardinality search for computing minimal triangulations of graphs. *Algorithmica*, 39(4):287–298, 2004. doi:10.1007/s00453-004-1084-3.

[8] Andreas Brandstädt, Feodor F. Dragan, and Falk Nicolai. LexBFS-orderings and powers of chordal graphs. *Discrete Mathematics*, 171(1):27–42, 1997. doi:10.1016/S0012-365X(96)00070-2.

[9] Anna Bretscher, Derek Corneil, Michel Habib, and Christophe Paul. A simple linear time LexBFS cograph recognition algorithm. *SIAM Journal on Discrete Mathematics*, 22(4):1277–1296, 2008. doi:10.1137/060664690.

[10] Yixin Cao, Zhifeng Wang, Guozhen Rong, and Jianxin Wang. Graph searches and their end vertices. In Pinyan Lu and Guochuan Zhang, editors, *30th International Symposium on Algorithms and Computation, ISAAC 2019*, volume 149 of *LIPIcs*, pages 1:1–1:18, Dagstuhl, 2019. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ISAAC.2019.1.

[11] Pierre Charbit, Michel Habib, and Antoine Mamcarz. Influence of the tie-break rule on the end-vertex problem. *Discrete Mathematics and Theoretical Computer Science*, 16(2):57–72, 2014. doi:10.46298/dmtcs.2081.

[12] Charles J. Colbourn and William R. Pulleyblank. Minimizing setups in ordered sets of fixed width. *Order*, 1:225–229, 1985. doi:10.1007/BF00383598.

[13] Derek G. Corneil, Barnaby Dalton, and Michel Habib. LDFS-based certifying algorithm for the minimum path cover problem on cocomparability graphs. *SIAM Journal on Computing*, 42(3):792–807, 2013. doi:10.1137/11083856X.

[14] Derek G. Corneil, Jérémie Dusart, Michel Habib, Antoine Mamcarz, and Fabien De Montgolfier. A tie-break model for graph search. *Discrete Applied Mathematics*, 199:89–100, 2016. doi:10.1016/j.dam.2015.06.011.

[15] Derek G. Corneil, Ekkehard Köhler, and Jean-Marc Lanlignel. On end-vertices of Lexicographic Breadth First Searches. *Discrete Applied Mathematics*, 158(5):434–443, 2010. doi:10.1016/j.dam.2009.10.001.

[16] Derek G. Corneil and Richard M. Krueger. A unified view of graph searching. *SIAM Journal on Discrete Mathematics*, 22(4):1259–1276, 2008. doi:10.1137/050623498.

[17] Derek G. Corneil, Stephan Olariu, and Lorna Stewart. The LBFS structure and recognition of interval graphs. *SIAM Journal on Discrete Mathematics*, 23(4):1905–1953, 2009. doi:10.1137/S0895480100373455.

[18] Robert P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51(1):161–166, 1950. doi:10.2307/1969503.

[19] Jérémie Dusart and Michel Habib. A new LBFS-based algorithm for cocomparability graph recognition. *Discrete Applied Mathematics*, 216:149–161, 2017. doi:10.1016/j.dam.2015.07.016.

[20] Jan Gorzny and Jing Huang. End-vertices of LBFS of (AT-free) bigraphs. *Discrete Applied Mathematics*, 225:87–94, 2017. doi:10.1016/j.dam.2017.02.027.

[21] Torben Hagerup. Biconnected graph assembly and recognition of DFS trees. Technical Report A 85/03, Universität des Saarlandes, 1985. doi:10.22028/D291-26437.

[22] Torben Hagerup and Manfred Nowak. Recognition of spanning trees defined by graph searches. Technical Report A 85/08, Universität des Saarlandes, 1985.

[23] Peter L. Hammer and Bruno Simeone. The splittance of a graph. *Combinatorica*, 1(3):275–284, 1981. doi:10.1007/BF02579333.

[24] John Hopcroft and Robert E. Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974. doi:10.1145/321850.321852.

[25] Ephraim Korach and Zvi Ostfeld. DFS tree construction: Algorithms and characterizations. In Jan van Leeuwen, editor, *Graph-Theoretic Concepts in Computer Science – 14th International Workshop, WG '88*, volume 344 of *LNCS*, pages 87–106, Berlin, Heidelberg, 1989. Springer. doi:10.1007/3-540-50728-0_37.

[26] Dieter Kratsch, Mathieu Liedloff, and Daniel Meister. End-vertices of graph search algorithms. In Peter Widmayer Vangelis Th. Paschos, editor, *Algorithms and Complexity – 9th International Conference, CIAC 2015*, volume 9079 of *LNCS*, pages 300–312, Cham, 2015. Springer. doi:10.1007/978-3-319-18173-8_22.

[27] Richard Krueger, Geneviève Simonet, and Anne Berry. A general label search to investigate classical graph search algorithms. *Discrete Applied Mathematics*, 159(2–3):128–142, 2011. doi:10.1016/j.dam.2010.02.011.

[28] Udi Manber. Recognizing breadth-first search trees in linear time. *Information Processing Letters*, 34(4):167–171, 1990. doi:10.1016/0020-0190(90)90155-Q.

[29] Chen-Hsing Peng, Jia-Shung Wang, and Richard C. T. Lee. Recognizing shortest-path trees in linear time. *Information Processing Letters*, 52(2):77–85, 1994. doi:10.1016/0020-0190(94)00126-X.

[30] Guozhen Rong, Yixin Cao, Jianxin Wang, and Zhifeng Wang. Graph searches and their end vertices. *Algorithmica*, 84:2642–2666, 2022. doi:10.1007/s00453-022-00981-5.

[31] Guozhen Rong, Yongjie Yang, and Wenjun Li. A polynomial-time algorithm for MCS partial search order on chordal graphs. arXiv:2212.04880, 2022.

[32] Guozhen Rong, Yongjie Yang, and Wenjun Li. A polynomial-time algorithm for MCS partial search order on chordal graphs. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science, MFCS 2023*, volume 272 of *LIPIcs*, pages 77:1–77:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2023.77.

[33] Donald J. Rose, R. Endre Tarjan, and George S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976. doi:10.1137/0205021.

[34] Robert Scheffler. Linearizing partial search orders. In Michael A. Bekos and Michael Kaufmann, editors, *Graph-Theoretic Concepts in Computer Science – 48th International Workshop, WG 2022*, volume 13453 of *LNCS*, pages 425–438, Cham, 2022. Springer. doi:10.1007/978-3-031-15914-5_31.

[35] Robert Scheffler. On the recognition of search trees generated by BFS and DFS. *Theoretical Computer Science*, 936:116–128, 2022. doi:10.1016/j.tcs.2022.09.018.

[36] Robert Scheffler. *Ready to Order? On Vertex and Edge Orderings of Graphs.* Doctoral thesis, BTU Cottbus-Senftenberg, 2023. doi:10.26127/BTUOpen-6301.

[37] Robert Scheffler. Recognizing LBFS trees of bipartite graphs. *Information Processing Letters*, 186:106483, 2024. doi:10.1016/j.ipl.2024.106483.

[38] Robert E. Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13(3):566–579, 1984. doi:10.1137/0213035.