A Polynomial-Time Algorithm for MCS Partial Search Order on Chordal Graphs

Guozhen Rong ✉

Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation,
Changsha University of Science and Technology, China

Yongjie Yang ✉ ⬤

Chair of Economic Theory, Saarland University, Saarbrücken, Germany

Wenjun Li ✉

Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation,
Changsha University of Science and Technology, China

─── **Abstract** ───

We study the partial search order problem (PSOP) proposed recently by Scheffler [WG 2022]. Given a graph G together with a partial order on the set of vertices of G, this problem determines if there is an $\mathcal{S}$-ordering that is consistent with the given partial order, where $\mathcal{S}$ is a graph search paradigm like BFS, DFS, etc. This problem naturally generalizes the end-vertex problem which has received much attention over the past few years. It also generalizes the so-called $\mathcal{F}$-tree recognition problem which has just been studied in the literature recently. Our main contribution is a polynomial-time dynamic programming algorithm for the PSOP of the maximum cardinality search (MCS) restricted to chordal graphs. This resolves one of the most intriguing open questions left in the work of Scheffler [WG 2022]. To obtain our result, we propose the notion of layer structure and study numerous related structural properties which might be of independent interest.

1 Introduction

Graph search paradigms are pervasive in algorithms for innumerable graph problems. In addition to the most popular paradigms breadth-first search (BFS) and depth-first search (DFS), several other prevalent graph search paradigms – including, for instance, lexicographic breadth-first search (LBFS), lexicographic depth-first search (LDFS), maximum cardinality search (MCS), maximal neighborhood search (MNS) – have also been extensively studied in the literature [10, 25, 28, 29]. These graph search paradigms have proved to be exclusively useful in dealing with a variety of graph problems [6, 8, 9, 22]. For instance, MCS has been successfully used in the recognition of special graphs [29], the computation of minimal separators [22], the computation of minimal triangulation of graphs [3], determining lower bounds of treewidth [5, 21], etc. In several of these algorithmic applications, last vertices

48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023).
Editors: Jérôme Leroux, Sylvain Lombardy, and David Peleg; Article No. 77; pp. 77:1–77:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

visited in graphs are crucial for the correctness of the algorithms. Last visited vertices also exhibit some nice structural properties. For instance, for a cocomparability graph that is Hamiltonian, if a vertex is last visited by LDFS, then there is a Hamiltonian path starting from this vertex [1, 8]. For more concrete examples on this issue, we refer to [1, 11]. These specialities of last visited vertices inspired Corneil, Köhler, and Lanlignel [11] to put forward the end-vertex problem, in which we are given a graph and a particular vertex v, and are asked whether v can be the last visited one according to a certain graph search paradigm. Thenceforth, investigation on the end-vertex problem has flourished, resulting in the complexity of the problem for both general graphs and many special graphs such as chordal graphs, split graphs, interval graphs, bipartite graphs, etc., being substantially established [1, 24, 32]. For a comprehensive summary of the recent progress, we refer to [16].

A closely related problem is the search tree recognition problem which has a relatively longer history [17, 18, 20]. This problem determines if a given spanning tree of a graph can be obtained via a traversal of the graph by a certain search paradigm. This problem comes natural for some search paradigms like BFS and DFS, since they not only output an ordering but also generate a spanning tree during the search. However, it is ill-defined for some other search paradigms like MCS and MNS. Aiming at overcoming the plight, Beisegel et al. [2] introduced the notions of $\mathcal{F}$-tree and $\mathcal{L}$-tree ($\mathcal{F}$ and $\mathcal{L}$ respectively stand for "first" and "last"). Particularly, given an ordering σ of the vertices of a graph, the $\mathcal{F}$-tree (respectively, $\mathcal{L}$-tree) is a spanning tree of the graph so that every vertex v other than the first one in σ is adjacent to its first (respectively, last) neighbor appearing before v in σ. BFS-trees and DFS-trees are $\mathcal{F}$-trees and $\mathcal{L}$-trees of BFS and DFS, respectively. Having these notions, Beisegel et al. [2] studied the complexity of the $\mathcal{F}$-tree recognition problem and the $\mathcal{L}$-tree recognition problem of the above-mentioned search paradigms for both general graphs and many special graph classes. Very recently, Scheffler [27] complemented these results by showing that the $\mathcal{L}$-tree recognition problem of BFS restricted to bipartite graphs, and the $\mathcal{F}$-tree recognition problem of DFS restricted to chordal graphs and chordal bipartite graphs are NP-hard, standing in a strong contrast to the polynomial-time solvability of the $\mathcal{F}$-tree recognition problem of BFS and the $\mathcal{L}$-tree recognition problem of DFS in general [18, 20].

From the known results, Scheffler [26] discerned that the complexity of the end-vertex problem and the complexity of the $\mathcal{F}$-tree recognition problem seemed to be somewhat connected. For instance, for LBFS, MCS, and MNS, both problems are NP-hard on weakly chordal graphs and are linear-time solvable on split graphs. Additionally, for MNS and MCS, both problems are polynomial-time solvable when restricted to chordal graphs [2, 24]. Towards a comprehensive understanding of the connection, Scheffler [26] introduced the partial search order problem (PSOP) which generalizes both the end-vertex problem and the $\mathcal{F}$-tree recognition problem. Given a graph G and a partial order R on the set of vertices of G, the PSOP of a search paradigm $\mathcal{S}$ determines if G admits an $\mathcal{S}$-ordering which linearly extends R. Scheffler [26] derived a polynomial-time algorithm for the PSOP of LBFS restricted to chordal bipartite graphs, and a polynomial-time algorithm for the PSOP of MCS restricted to split graphs. However, whether the PSOP of MCS restricted to chordal graphs, arguably the most intriguing case, is polynomial-time solvable is unknown prior to our current work. We resolve this open question in the affirmative. To this end, we propose the notion of layer structure and study a number of structural properties which might be of independent interest. At a high level, based on the properties studied, we iteratively decompose the clique graph of a given chordal graph into what we call layer structures, handle the components (which we call units) of each layer structure separately, and utilize dynamic programming techniques to merge local solutions into a whole one.

2 Preliminaries

For an integer i, $[i]$ denotes the set of positive integers no greater than i.

2.1 Graphs

We follow standard notions in graph theory. For notions used but not defined in the paper, we refer to [31]. The graphs we consider are finite, undirected, and simple. Let G be a graph. The *vertex set* and *edge set* of G are denoted by $V(G)$ and $E(G)$, respectively. For a vertex $v \in V(G)$, its *neighborhood* in G, denoted $N_G(v)$, is the set of vertices adjacent to v in G. A *clique* of G is a subset of pairwise adjacent vertices in G. We call a clique of G containing a vertex $v \in V(G)$ a *v-clique*. Analogously, a clique of G containing a subset $X \subseteq V(G)$ is called an *X-clique*. The subgraph of G *induced* by $X \subseteq V(G)$ is denoted by $G[X]$.

A *path* P of length t is a graph with a sequence of $t+1$ distinct vertices $v_1, v_2, \ldots, v_{t+1}$ and with the edge set $\{v_i v_{i+1} \mid i \in [t]\}$. We say that P is a path between v_1 and v_t, or simply call it a v_1-v_t *path*. Two vertices in G are *connected* if there is a path between them in G. For $u, v \in V(G)$, a *u-v separator* is a subset $X \subseteq V(G)$ so that u and v are disconnected after deleting all vertices in X from G. A *u-v* separator X is *minimal* if there are no other *u-v* separators X' such that $X' \subsetneq X$. The *length* of a cycle is the number of edges it contains. A *hole* is an induced cycle of length greater than three. A graph is *chordal* if it does not contain any holes as induced subgraphs.

2.2 The Partial Search Order Problem

A *partial order* on a set X is a reflexive, antisymmetric, and transitive binary relation on X. For ease of exposition, for a partial order R, we sometimes use $x <_R y$ to denote $(x, y) \in R$. A *linear order* is a partial order that is complete. We usually write a linear order R in the format of $(x_1, x_2, \ldots, x_m)$ which means that $(x_i, x_j) \in R$ for all $i, j \in [m]$ such that $i < j$. A linear order R *extends* a partial order R' if for every $(x, y) \in R'$ it holds that $(x, y) \in R$. We also call R a *(linear) extension* of R'. For a binary relation R on a set X, and for $X' \subseteq X$, we use $R|_{X'}$ to denote R restricted to X'. For a graph search paradigm $\mathcal{S}$ and a graph G, an $\mathcal{S}$-*ordering* of G is an ordering of $V(G)$ that can be generated from an $\mathcal{S}$ search on G. The partial search order problem of $\mathcal{S}$, denoted PSOP-$\mathcal{S}$, is defined as follows.

PSOP-$\mathcal{S}$	
Given:	A connected graph G and a partial order R on $V(G)$.
Question:	Is there an $\mathcal{S}$-ordering of G that extends R?

This paper focuses on PSOP-MCS restricted to connected chordal graphs.

2.3 Clique Graphs

It has long been known that chordal graphs admit a characterization in terms of their clique trees. Precisely, a connected graph G is chordal if and only if there exists a tree T whose vertices one-to-one correspond to maximal cliques of G so that for every vertex $v \in V(G)$ the vertices of T corresponding to all maximal v-cliques of G induce a subtree of T [4, 7, 15, 29]. Such a tree T is referred to as a *clique tree* of G [7, 15, 30]. It is a folklore that every chordal graph G has at most $|V(G)|$ maximal cliques [13], and hence every clique tree of G contains at most $|V(G)|$ vertices.

Another relevant notion is *clique graph*, first introduced by Galinier, Habib, and Paul [14]. Precisely, the clique graph of a connected chordal graph G, denoted $C(G)$, is the graph whose vertex set is exactly the set of all maximal cliques of G, and two vertices K and K' in
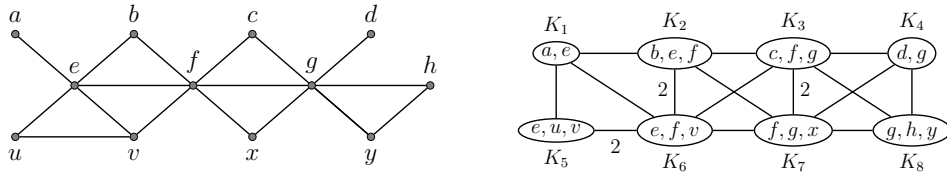
Figure 1 A connected chordal graph (left) and its clique graph (right). In the clique graph, all omitted edge weights are 1.

the clique graph are adjacent if and only if $K \cap K'$ is a minimal u-v separator of G for all $u \in K \setminus K'$ and all $v \in K' \setminus K$. Each edge KK' in the clique graph $C(G)$ is associated with the *label* $K \cap K'$ and with the *weight* $|K \cap K'|$.

For clarity, hereinafter we call vertices in a clique tree or a clique graph *nodes*.

▶ **Lemma 1** ([14]). *Let G be a connected chordal graph G, and let K and K' be two maximal cliques in G. Then, K and K' are adjacent in $C(G)$ if and only if they are adjacent in some clique tree of G.*

In effect, Lemma 1 asserts that the clique graph of a connected chordal graph is the union of all clique trees of the same graph.

For a label S of some edge in $C(G)$, we use $C(G) \ominus S$ to denote the graph obtained from $C(G)$ by deleting all edges with the label S. For example, for the chordal graph G in Figure 1, $C(G) \ominus \{f\}$ is the graph shown in Figure 2.
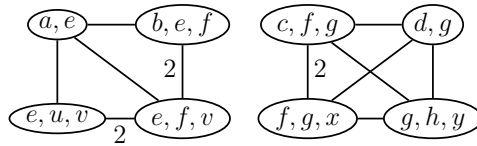
Figure 2 $C(G) \ominus \{f\}$ where G is the connected chordal graph in Figure 1.

▶ **Lemma 2** ([24]). *Let G be a connected chordal graph, and let S be the label of an edge in $C(G)$ with the minimum weight. Then,*

(i) *for every $v \in V(G) \setminus S$, all maximal v-cliques of G are in the same connected component of $C(G) \ominus S$; and*

(ii) *for every $u, v \in V(G) \setminus S$, S is a u-v separator in G if and only if any maximal u-clique and any maximal v-clique of G are in different connected components of $C(G) \ominus S$.*

2.4 Graph Search Paradigms

Now we give definitions of three graph search paradigms, namely, MCS, generic search, and Prim search. Each search paradigm regulates how the vertices of a graph are visited one after another without repeating. Our focus is PSOP-MCS, but our algorithm for PSOP-MCS resorts to Prim search of clique graphs and generic search of layer structures (defined in Section 3) of clique graphs.

Both MCS and generic search arbitrarily select one vertex to visit first. Then, MCS picks an unvisited vertex having the maximum number of visited neighbors as the next to visit [29], while the generic search picks an unvisited vertex having at least one visited neighbor as the next to visit [28].

Prim's algorithm is one of the most famous algorithms for finding minimum spanning trees [12, 19, 23]. In a nutshell, starting from a tree consisting of an edge with the minimum weight, the algorithm grows the tree by adding edges, one-by-one, of minimum possible weights without destroying the tree structure, until the tree becomes a spanning tree. By turning "minimum" to "maximum" in the algorithm, it instead returns a maximum spanning tree. Algorithm 1 delineates the Prim search modified from Prim's algorithm [24]. Generally speaking, starting from any arbitrary vertex, it picks as the next one to be visited a so far unvisited vertex incident to an edge with the maximum weight among all edges between visited vertices and unvisited vertices.

Algorithm 1 Prim Search.

Input: A connected graph G of n vertices where every edge has a weight.

Output: An ordering π of $V(G)$.

1: $\pi(v) \leftarrow 1$, where v is an arbitrary vertex of G;   /* the first visited vertex */

2: $S \leftarrow \{v\}$;

3: **for** $i = 2, 3, \ldots, n$ **do**

4:      let v be a vertex in $V(G) \setminus S$ incident to an edge with the maximum weight among all edges between S and $V(G) \setminus S$;

5:      $\pi(v) \leftarrow i$;   /* visit v */

6:      $S \leftarrow S \cup \{v\}$;

7: **return** π;

Following [24], we call an ordering obtained from applying Algorithm 1 to a graph G a *Prim ordering* of G. Prim orderings of the clique graph of a chordal graph have an appealing property in respect of their clique graphs, as stated in the following lemma.

▶ **Lemma 3** ([24]). *Let G be a connected chordal graph, and let $(K_1, K_2, \ldots, K_t)$ be a Prim ordering of $C(G)$. For every $i \in [t]$, the subgraph of $C(G)$ induced by $\{K_1, K_2, \ldots, K_i\}$ is the clique graph of the subgraph of G induced by $\bigcup_{j \in [i]} K_j$.*

Let σ be an ordering of $V(G)$, and let π be an ordering of the maximal cliques of G. For $v \in V(G)$, we use K_π^v to denote the first v-clique in π. We say that σ is a *generation* of π (or π *generates* σ) if for all $x, y \in V(G)$ it holds that $K_\pi^x <_\pi K_\pi^y$ implies $x <_\sigma y$. Precisely, for an ordering $\pi = (K_1, K_2, \ldots, K_t)$ of maximal cliques of G and $i \in [t-1]$, let $V(\pi, i) = \bigcup_{j \in [i]} K_j$ be the set of vertices of G contained in at least one of the first i cliques from π. For $i \in [t] \setminus \{1\}$, let $K(\pi, i) = K_i \setminus V(\pi, i-1)$ be the set of vertices of G contained in K_i but not in any other cliques before K_i in π. Besides, let $K(\pi, 1) = K_1$. Then, σ is a generation of π if and only if it is of the form $(\overrightarrow{K(\pi, 1)}, \overrightarrow{K(\pi, 2)}, \ldots, \overrightarrow{K(\pi, t)})$, where for a set X, $\overrightarrow{X}$ can be any ordering of X.

▶ **Lemma 4** ([24]). *Let G be a connected chordal graph. Then, an ordering of $V(G)$ is an MCS ordering of G if and only if it is a generation of some Prim ordering of $C(G)$.*

3 Layer Structures of Clique Graphs

In this section, we introduce the notion of layer structure, and explore a number of structural properties pertinent to this notion. Throughout this section, let G be a connected chordal graph, and let $K^\star$ be a maximal clique of G.

Let $\mathcal{U}$ be the set of connected components of $C(G)$ after the deletion of all edges with the minimum weight. For the sake of readability, let us call each $U \in \mathcal{U}$ a *unit*. If a maximal clique K of G is a node in a unit, we say that K is *contained* in this unit. We use U^K to

denote the unit containing K. Besides, we use $\mathcal{K}(U)$ to denote the set of maximal cliques of G contained in a unit U, and use $\mathcal{V}(U)$ to denote the set of vertices of G contained in nodes of U, i.e., $\mathcal{V}(U) = \bigcup_{K \in \mathcal{K}(U)} K$. We say that an edge in $C(G)$ *crosses* two units if the two endpoints of the edge are respectively from the two units.

The layer structure we shall study is a refinement of the clique graph $C(G)$. The following two lemmas pinpoint where the refinement lies.

▶ **Lemma 5.** *The weights of edges of $C(G)$ whose both endpoints are contained in the same unit are greater than the minimum weight of edges of $C(G)$.*

Proof. Let KK' be an edge with the minimum weight in $C(G)$, and let $S = K \cap K'$. Therefore, S is a u-v separator for all $u \in K \setminus S$ and $v \in K' \setminus S$. Then, by Lemma 2 (ii), K and K' are in different connected components of $C(G) \ominus S$, implying that K and K' are contained in different units. ◀

Lemma 5 equivalently asserts that every unit U is exactly the subgraph of $C(G)$ induced by $\mathcal{K}(U)$. To put it another way, an edge in $C(G)$ crosses two units if and only if it has the minimum weight in $C(G)$.

▶ **Lemma 6.** *Let U and U' be two units from $\mathcal{U}$ so that there are edges in $C(G)$ crossing U and U'. Then, all edges in $C(G)$ crossing U and U' have the same label.*

Proof. Towards a contradiction, assume that $C(G)$ contains two distinct edges $K_1 K_2$ and $K_3 K_4$ crossing U and U' with different labels. Let $S = K_1 \cap K_2$ and let $S' = K_3 \cap K_4$. So, $S \neq S'$. Obviously, U and U' both remain connected in $C(G) \ominus S$. Moreover, since the edge $K_3 K_4$ is present in $C(G) \ominus S$, U and U' are in the same connected component of $C(G) \ominus S$. This means that K_1 and K_2 are in the same connected component of $C(G) \ominus S$. However, this contradicts Lemma 2 (ii). ◀

Now we are ready to define the layer structure.

▶ **Definition 7** (Layer Structure). *The layer structure of $C(G)$ rooted by $K^\star$ is a graph with the vertex set $\mathcal{U}$ so that there is an edge between two units in $\mathcal{U}$ if and only if there exists at least one edge in $C(G)$ crossing the two units. The label and the weight of an edge UU' in the layer structure are respectively $K \cap K'$ and $|K \cap K'|$, where KK' can be any edge in $C(G)$ crossing U and U'. The unit $U^{K^\star}$ is called the root of the layer structure. A unit is in the i-th layer if it is at a distance i from the root, where the distance between two units is defined as the length of a shortest path between them in the layer structure.*

See Figure 3 for an illustration of Definition 7. By Lemma 6, the labels and the weights of edges in the layer structure are well-defined. Let $\mathcal{L}_i$ be the set of all units in the i-th layer, and let $\mathcal{L}_{\leq i} = \bigcup_{j \in [i] \cup \{0\}} \mathcal{L}_j$. Obviously, $\mathcal{L}_0 = \{U^{K^\star}\}$. In addition, if two units from respectively two layers $\mathcal{L}_i$ and $\mathcal{L}_j$ are adjacent, it holds that $|i - j| \leq 1$. Recall that G is connected. Hence, $C(G)$ and the layer structure are connected too. As a result, every unit in $\mathcal{U}$ is in some layer.

In the following, we explore numerous properties of the layer structure.

▶ **Property 1.** *Let UU' be an edge in the layer structure. Then, every path between U and U' in the layer structure contains an edge with the same label as UU'.*

Proof. Let KK' be an edge in $C(G)$ crossing the two units U and U'. Let $S = K \cap K'$. It is clear that S is both the label of KK' and the label of UU'. By Lemma 2 (ii), K and K' are disconnected in $C(G) \ominus S$. It follows that every K-K' path in $C(G)$ contains at least one edge with the label S. Then, by Lemma 5, we know that every U-U' path in the layer structure contains at least one edge with the label S. ◀
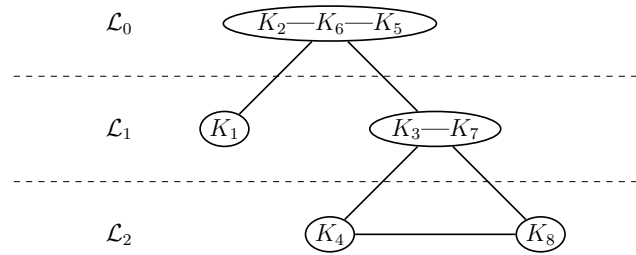
Figure 3 The layer structure of the chordal graph in Figure 1 rooted by K_2 (or K_5, K_6).

▶ **Property 2.** *Let U_1U_2 and U_3U_4 be two distinct edges of the layer structure with the same label S. Then, the units in the set $\{U_1, U_2, U_3, U_4\}$ are pairwise adjacent in the layer structure, and all edges among them have the same label S.*

Proof. Let K_1K_2 and K_3K_4 be two distinct edges of $C(G)$ both with the label S, where $K_i \in \mathcal{K}(U_i)$ for every $i \in [4]$. We first show that U_1, U_2, U_3, and U_4 are pairwise adjacent in the layer structure. Assume, for the sake of contradiction, that one from $\{U_1, U_2\}$ is not adjacent to one from $\{U_3, U_4\}$. By symmetry, suppose that $U_1 \neq U_3$, and U_1 is not adjacent to U_3 in the layer structure.

We claim that K_1 and K_3 are disconnected in $C(G) \ominus S$. Assume for contradiction that K_1 and K_3 are connected in $C(G) \ominus S$. Let C' be the connected component of $C(G) \ominus S$ containing K_1 and K_3, and let V' be the set of vertices of G contained in nodes of C'. By the minimality of $|S|$, there exists a Prim ordering of $C(G)$ so that all maximal cliques of G contained in C' are visited before all the other maximal cliques of G. Additionally, in light of Lemma 2 (ii), the subgraph of $C(G)$ induced by the nodes in C' does not contain any edge with the label S. Then, by Lemma 3, C' is the clique graph of $G[V']$. Obviously, $S \subsetneq V'$. By Lemma 1, C' contains all clique trees of $G[V']$ as subgraphs. We fix a clique tree of $G[V']$. There is a unique K_1-K_3 path in the clique tree, and by the definition of clique trees, all nodes on this path are S-cliques. Obviously, this path is also present in $C(G) \ominus S$. The label of each edge on this path cannot be S, since such edges are absent in $C(G) \ominus S$. So, the labels of all edges on this path properly contain S. This indicates that K_1 and K_3 are contained in the same unit in the layer structure. However, this contradicts that $U_1 \neq U_3$.

So, it holds that K_1 and K_3 are disconnected in $C(G) \ominus S$. Then, according to Lemma 2 (ii), S is a u-v separator for all $u \in K_1 \setminus S$ and all $v \in K_3 \setminus S$. By the minimality of $|S|$ and the definition of clique graphs, K_1K_3 is an edge of $C(G)$ with the label S. As a result, U_1U_3 is an edge of the layer structure, contradicting that U_1 and U_3 are nonadjacent in the layer structure.

Now we can conclude that the units in $\{U_1, U_2, U_3, U_4\}$ are pairwise adjacent in the layer structure. Then, from Property 1 and the fact that both U_1U_2 and U_3U_4 have the label S, it follows that the labels of the edges among U_1, U_2, U_3, and U_4 are all S. ◄

Note that in Property 2 it may be that $\{U_1, U_2\} \cap \{U_3, U_4\} \neq \emptyset$.

▶ **Property 3.** *Every unit U in the i-th layer $\mathcal{L}_i$ where $i \geq 1$ is adjacent to exactly one unit from the layer $\mathcal{L}_{i-1}$.*

Proof. Let U be a unit from the i-th layer. The statement holds trivially for $i = 1$. Consider now the case where $i \geq 2$. Assume, for the sake of contradiction, that there are two distinct units U_1 and U_2 in the layer $\mathcal{L}_{i-1}$ both adjacent to U in the layer structure. Let S_1 and S_2 be the labels of UU_1 and UU_2, respectively. By Definition 7, there is a path from the root

to U_1, and a path from the root to U_2 in the layer structure. Then, as $i \geq 2$, there exists a U_1-U_2 path P of length at least two in the layer structure such that all inner units of P are from $\mathcal{L}_{\leq i-2}$. Our proof proceeds by distinguishing between the following two cases.

Case 1: $S_1 = S_2$.
    By Property 2, $U_1 U_2$ is an edge with the label S_1 in the layer structure. By Property 1, there exists an edge in P with the label S_1. By Property 2, U is adjacent to the two endpoints of this edge, which is impossible since U is from the i-th layer but at least one of the two endpoints of the edge is from $\mathcal{L}_{\leq i-2}$.

Case 2: $S_1 \neq S_2$.
    By Property 1, there exists an edge in $E(P) \cup \{UU_2\}$ with the label S_1. As $S_1 \neq S_2$, this edge belongs to P. By Property 2, U is adjacent to the two endpoints of this edge. However, analogous to the discussion in Case 1, we know that this is impossible.

As both cases lead to some contradictions, we know that U is adjacent to exactly one unit from $\mathcal{L}_{i-1}$. ◀

Now for each unit U from a layer $\mathcal{L}_i$ where $i \geq 1$, we call the only unit from the layer $\mathcal{L}_{i-1}$ adjacent to U the *parent* of U, and use $\mathsf{parent}(U)$ to denote it. Correspondingly, we say that U is a *child* of $\mathsf{parent}(U)$. Furthermore, for a unit U from a layer $\mathcal{L}_i$ and a unit U' from a layer $\mathcal{L}_j$ such that $i < j$, we say that U' is a *descendant* of U if there is a path from the root to U' through U (i.e., U is also on the path) in the layer structure. For a unit U, let $\mathsf{dst}(U)$ be the set of all descendants of U, and let $\mathsf{dst}[U] = \mathsf{dst}(U) \cup \{U\}$.

▶ **Property 4.** *Let U_1 and U_2 be two units in the same layer $\mathcal{L}_i$ where $i \geq 1$. Then, the following statements are equivalent:*
(1) *U_1 and U_2 are adjacent.*
(2) *$\mathsf{parent}(U_1) = \mathsf{parent}(U_2)$, and the label of the edge between U_1 and its parent equals that between U_2 and its parent.*

Proof. From Property 2, we know that Statement (2) implies Statement (1). In the following, we show that Statement (1) implies Statement (2).

    Assume that U_1 and U_2 are adjacent, and let S be the label of the edge between them. Let $U_3 = \mathsf{parent}(U_1)$ and let $U_4 = \mathsf{parent}(U_2)$. We first prove that U_1 and U_2 have the same parent, i.e., $U_3 = U_4$. Assume, for contradiction, that $U_3 \neq U_4$. By Properties 2 and 3, this implies that neither the label of $U_1 U_3$ nor the label of $U_2 U_4$ is S. In addition, it also holds that $i > 1$, implying that there exists a U_3-U_4 path P of length at least two in the layer structure whose inner units are all from $\mathcal{L}_{\leq i-2}$ (cf. the proof of Property 3). Then, by Property 1, there exists an edge in P with the label S. By Property 2, the two endpoints of this edge are adjacent to U_1 in the layer structure. However, this is impossible since at least one endpoint of this edge is from $\mathcal{L}_{\leq i-2}$ but U_1 is from the layer $\mathcal{L}_i$. This completes the proof for that $U_3 = U_4$. Having $U_3 = U_4$, from Properties 1 and 2, it follows that the labels of $U_1 U_3$ and $U_2 U_4$ are both S. ◀

    Property 4 implies that if a subset of units in the same layer $\mathcal{L}_i$ are connected in the layer structure restricted to $\mathcal{L}_i$, then they are pairwise adjacent. For ease of exposition, we group units in the same layer into *bags* so that two units are in the same bag if they are adjacent. By Property 3, if we ignore edges inside all bags in the layer structure, we obtain a tree rooted at $U^{K^\star}$. As an important consequence, every path connecting two units from the same layer $\mathcal{L}_i$ is completely contained in $\mathcal{L}_{\leq i}$.

    In addition, Property 3 indicates that every nonroot unit U in a layer $\mathcal{L}_i$ is adjacent to exactly one unit in $\mathcal{L}_{\leq i-1}$, and this unit is its parent in $\mathcal{L}_{i-1}$. Property 4 further strengthens that $\mathsf{parent}(U)$ indeed separates all descendants of $\mathsf{parent}(U)$ from all the other units. In other

words, $\{\mathsf{parent}(U)\}$ is a U'-U'' separator in the layer structure for all $U' \in \mathsf{dst}(\mathsf{parent}(U))$ and all $U'' \in \mathcal{U} \setminus \mathsf{dst}[\mathsf{parent}(U)]$. An important consequence is that every generic search ordering of the layer structure starting from the root visits every nonroot unit after (not necessarily consecutive) its parent.

The next property identifies labels of edges between adjacent units.

▶ **Property 5.** *The label of every edge UU' in the layer structure is $\mathcal{V}(U) \cap \mathcal{V}(U')$.*

Proof. Let U and U' be two adjacent units in the layer structure. Let $S = \mathcal{V}(U) \cap \mathcal{V}(U')$, and let S' be the label of the edge between U and U' in the layer structure. Clearly, $S' \subseteq S$. Therefore, to complete the proof, it needs only to show that every $v \in S$ is contained in S'. In light of Properties 3 and 4, only the two cases described below may occur.

Case 1: one of U and U' is the parent of the other.

For the sake of contradiction, assume that there exists $v \in S \setminus S'$. By Lemma 6, all edges between $\mathcal{K}(U)$ and $\mathcal{K}(U')$ in $C(G)$ have the same label S'. By the definition of the layer structure, $|S'|$ is a minimum edge weight in $C(G)$. Then, by Properties 3 and 4, U and U' are disconnected in the layer structure after removing all edges with the label S'. This is equivalent to every $K \in \mathcal{K}(U)$ and every $K' \in \mathcal{K}(U')$ being disconnected in $C(G) \ominus S'$. However, as $v \in \mathcal{V}(U) \cap \mathcal{V}(U')$, this violates Lemma 2 (i).

Case 2: U and U' are in the same bag.

By Property 4, U and U' have the same parent, say $\hat{U}$, and the labels of edges among units in the same bag as U and their parent $\hat{U}$ are S'. Then, similar to Case 1, it is easy to see that U and U' are disconnected in the layer structure after removing all edges with the label S', which violates Lemma 2 (i) too.

As both cases violate Lemma 2, we know that Property 5 holds. ◀

▶ **Property 6.** *Let v be a vertex in G. Let i be the minimum integer such that $\mathcal{L}_i$ contains a maximal v-clique. Then, all maximal v-cliques of G contained in $\mathcal{L}_i$ are in one unit.*

Proof. Let v and i be as stipulated in the statement of Property 6. Assume for contradiction that there exist two maximal v-cliques K and K' respectively from two different units U and U' in the layer $\mathcal{L}_i$. Our proof is completed by considering the following two cases.

Case 1: U and U' are adjacent in the layer structure.

By Property 5, v is in the label of the edge UU'. By Property 4, U and U' have the same parent, say, $\hat{U}$. Moreover, by Properties 2 and 4, the edges $U\hat{U}$ and $U'\hat{U}$ have the same label as the edge UU'. It follows that $\hat{U}$ contains some v-clique. However, this contradicts that i is the minimum integer with $\mathcal{L}_i$ containing a maximal v-clique.

Case 2: U and U' are not adjacent in the layer structure.

Note that in this case $i \geq 1$. As the subgraph of $C(G)$ induced by the set of maximal v-cliques is connected, there is a K-K' path P in $C(G)$ so that v is in the label of every edge on the path. By Properties 3 and 4, this path contains at least one maximal clique of G contained in $\mathsf{parent}(U)$. However, similar to Case 1, this is in contradiction with the definition of i.

As each of the above two cases leads to a certain contradiction, Property 6 holds. ◀

Property 6 shows that for every vertex v in G, there is a unique unit that contains a maximal v-clique and is at the least distance to the root in the layer structure. Let U^v be such a unique unit for v.

▶ **Property 7.** *For each $v \in V(G)$, every unit containing a maximal v-clique of G is from $\mathsf{dst}[U^v]$.*

Proof. Assume, for the sake of contradiction, there exists $v \in V(G)$ and a unit U such that $v \in \mathcal{V}(U)$ and $U \notin \mathsf{dst}[U^v]$. By Property 6, U and U^v cannot be in the same layer. Then, as $U \notin \mathsf{dst}[U^v]$, every U^v-U path in the layer structure contains the parent of U^v. As all maximal v-cliques are connected in $C(G)$, and $v \in \mathcal{V}(U) \cap \mathcal{V}(U^v)$, we know that $v \in \mathcal{V}(\mathsf{parent}(U^v))$. However, this contradicts the definition of U^v.   ◀

▶ **Property 8.** *Let U and U' be two units from the same bag, and let $\hat{U}$ be their parent. Then, for every edge KK' in $C(G)$ so that $K \in \mathcal{K}(U)$ and $K' \in \mathcal{K}(U')$, there exists $\hat{K} \in \mathcal{K}(\hat{U})$ which is adjacent to both K and K' in $C(G)$.*

Proof. Let $K \in \mathcal{K}(U)$ and $K' \in \mathcal{K}(U')$ be as stipulated in Property 8. Let $S = K \cap K'$. As U and U' are in the same bag, they are adjacent in the layer structure. By Lemma 6 and Definition 7, the label of the edge UU' is S. This means that $|S|$ is the minimum weight of edges in $C(G)$. By Properties 2 and 4, the labels of the edges $U\hat{U}$ and $U'\hat{U}$ are also S. As a result, there exists $\hat{K} \in \mathcal{K}(\hat{U})$ so that $S \subseteq \hat{K}$. By Properties 3 and 4, $\hat{K}$ is disconnected from K in $C(G) \ominus S$. Then, according to Lemma 2 (ii) and the minimality of $|S|$, S is a minimal u-v separator in G for all $u \in K \setminus \hat{K}$ and all $v \in \hat{K} \setminus K$. Therefore, $\hat{K}$ and K are adjacent in $C(G)$. Similarly, $\hat{K}$ is also adjacent to K' in $C(G)$.   ◀

Now we study two lemmas which provide insight into connections among MCS orderings of G, Prim orderings of $C(G)$, and generic search orderings of the layer structure. We say that a Prim ordering π of the clique graph $C(G)$ *respects* a partial order R on $V(G)$ if for every $(x, y) \in R$ it holds that $K_\pi^x <_\pi K_\pi^y$ or $K_\pi^x = K_\pi^y$. By saying that an ordering starts with elements in a subset, we mean the elements in the subset are before all the other elements in the ordering.

▶ **Lemma 8.** *Let R be a partial order on $V(G)$. There is an MCS ordering of G extending R if and only if there is a Prim ordering of $C(G)$ respecting R. Moreover, given a Prim ordering of $C(G)$ which starts with some node K and respects R, we can construct an MCS ordering of G starting with the vertices in K and extending R in polynomial time.*

Proof. We start the proof with the first statement. For the forward direction, let σ be an MCS ordering of G extending R. By Lemma 4, there exists a Prim ordering π of $C(G)$ that generates σ. For every $(x, y) \in R$, since σ extends R, it holds that $x <_\sigma y$. As π generates σ, it holds that $K_\pi^x <_\pi K_\pi^y$ or $K_\pi^x = K_\pi^y$. For the backward direction, let $\pi = (K_1, K_2, \ldots, K_t)$ be a Prim ordering of $C(G)$ so that for all $(x, y) \in R$ it holds that either $K_\pi^x <_\pi K_\pi^y$ or $K_\pi^x = K_\pi^y$. Let σ be a generation of π so that for every $i \in [t]$ it holds that σ restricted to $K(\pi, i)$ extends R restricted to $K(\pi, i)$, i.e., $\sigma|_{K(\pi,i)}$ is a linear extension of $R|_{K(\pi,i)}$. As for every $i, j \in [t]$ such that $i \neq j$, $K(\pi, i)$ and $K(\pi, j)$ are disjoint, σ is well-defined. By Lemma 4, σ is an MCS ordering of G. To complete the proof for the first statement, it suffices to show that σ is a linear extension of R. Let $(x, y) \in R$. If $K_\pi^x = K_\pi^y$, i.e., x and y are contained in some $K(\pi, i)$ where $i \in [t]$, as $\sigma|_{K(\pi,i)}$ extends $R|_{K(\pi,i)}$, it holds that $x <_\sigma y$. Otherwise, $K_\pi^x <_\pi K_\pi^y$ holds. Then, as σ is a generation of π, $x <_\sigma y$ holds too. This completes the proof that σ extends R.

Concerning the second statement, observe that the above proof for the backward direction is constructive, and the polynomial-time solvability follows from the fact that computing a linear extension of a partial order can be done in polynomial time.   ◀

By Property 6, each partial order R on $V(G)$ specifies a partial order on the units: $\mathcal{Q}^R = \{(U^x, U^y) \mid (x, y) \in R\}$.

▶ **Lemma 9.** *Let R be a partial order on $V(G)$. Then, if there is an MCS ordering of G that starts with vertices from $K^\star$ and extends R, there is a generic search ordering of the layer structure rooted by $K^\star$ that starts with $U^{K^\star}$ and extends $\mathcal{Q}^R$.*

Proof. Let σ be an MCS ordering of G that starts with the vertices from $K^\star$ and extends R. By Lemma 4, there is a Prim ordering π of $C(G)$ that generates σ. Obviously, $K^\star$ is the first node in π. Then, by Algorithm 1, Lemma 5, and Definition 7, we know that for every unit U, the nodes in U are consecutive in π. Moreover, in view of Properties 3 and 4, for every nonroot unit U, all nodes from parent(U) are before all nodes from U in π. Consequently, there is a linear order $\pi' = (U_1, U_2, \ldots, U_t)$ of the units in the layer structure so that

(1) $U_1 = U^{K^\star}$;

(2) for every nonroot unit U, parent(U) is before U in π'; and

(3) for every $i \in [t]$, all nodes of U_i are consecutive in the Prim ordering π.

Conditions (1) and (2) mean that π' is a generic search ordering of the layer structure starting with the root. It remains to show that π' extends $\mathcal{Q}^R$. For this purpose, let $(x, y) \in R$ such that $U^x \neq U^y$. As σ extends R, x is before y in σ. As π generates σ, either $K_\pi^x = K_\pi^y$ holds or $K_\pi^x <_\pi K_\pi^y$ holds. Condition (2) given above and Property 7 imply that K_π^v is contained in U^v for all $v \in V(G)$. Then, as $U^x \neq U^y$, it must be that $K_\pi^x <_\pi K_\pi^y$. Finally, by Condition (3) given above, we know that U^x is before U^y in π'. This completes the proof that π' extends $\mathcal{Q}^R$. ◀

4 A Dynamic Programming Algorithm for PSOP

In this section, we present a polynomial-time dynamic programming algorithm for PSOP-MCS restricted to chordal graphs.

For an ordering π of units in a layer structure, and an ordering σ of nodes contained in a unit U, *realizing* π by σ is the operation of replacing U in π with σ. For instance, for $\pi = (U_1, U_2, U_3, U_4)$ and $\sigma = (K_1, K_2, K_3)$ where $\{K_1, K_2, K_3\}$ is the set of nodes contained in U_2, realizing π by σ results in the linear order $(U_1, K_1, K_2, K_3, U_3, U_4)$.

▶ **Theorem 10.** *PSOP-MCS restricted to chordal graphs is polynomial-time solvable.*

Proof. Let $I = (G, R)$ be an instance of PSOP-MCS where G is a connected chordal graph of n vertices. Our algorithm consists of the following steps.

Step 1 We sort the weights of edges in the clique graph $C(G)$ of G in increasing order. Let $(w(1), w(2), \ldots, w(t))$ be this order, where t is the number of different weights of edges in $C(G)$. Hence $w(1) < w(2) < \cdots < w(t)$ holds. Notice that $t = O(n)$ since edges of $C(G)$ may have at most $n - 2$ different weights.

Step 2 For each $i \in [t]$, let $C^i(G)$ be the graph obtained from $C(G)$ by removing edges whose weights are from $\{w(1), w(2), \ldots, w(i)\}$. Let $C^0(G) = C(G)$. Clearly, every $C^i(G)$, where $i \in [t]$, is obtained from $C^{i-1}(G)$ by deleting all edges with the minimum weight.

Step 3 We maintain a binary dynamic programming table $D(i, H, K)$, where $i \in [t] \cup \{0\}$, H is a connected component of $C^i(G)$, and K is a node in H. (Note that for $i = 0$, we have that $H = C(G)$.) As $t = O(n)$, every chordal graph of n vertices has at most n maximal cliques, and K is a node in H, the table has $O(n^2)$ entries.

For each connected component H of some $C^i(G)$, let $\mathcal{V}(H)$ be the subset of vertices of G contained in nodes of H. We define $D(i, H, K)$ to be 1 if there is a Prim ordering of H which starts with K and respects $R|_{\mathcal{V}(H)}$, and define $D(i, H, K)$ to be 0 otherwise. (We elaborate on how to compute the table later.)

Step 4 After the table is computed, if $D(0, C(G), K) = 0$ for all maximal cliques K of G, by the definition of the table there is no Prim ordering of $C(G)$ respecting R, and by Lemma 8, the given instance I is a No-instance. Otherwise, there exists a maximal clique K of G such that $D(0, C(G), K) = 1$. By the definition of the table and by Lemma 8, there is an MCS ordering of G extending R. Therefore, in this case, we conclude that I is a Yes-instance.

Computing the table defined above lies at the core of the algorithm, which is the main focus of the remainder of the proof. To this end, we first show that each graph H used in Step 3 is a clique graph of a connected chordal graph.

$\triangleright$ **Claim 11.** For every $i \in [t] \cup \{0\}$, every connected component H of $C^i(G)$ is the clique graph of the subgraph of G induced by $\mathcal{V}(H)$.

Proof of Claim 11. We prove the claim by induction on i. The statement is clearly true for $i = 0$. Now let $i \in [t]$, and let H be a connected component of $C^i(G)$. Let H' be the connected component of $C^{i-1}(G)$ containing H. By induction, H' is the clique graph of $G[\mathcal{V}(H')]$ which is a connected chordal graph. Obviously, H is a connected component of H' after removing all edges with the minimum weight in H'. As a consequence, there is a Prim ordering of H' which starts with a node from H and consecutively visits nodes in H. Then, by Lemma 3, H is the clique graph of the subgraph of $G[\mathcal{V}(H')]$ induced by $\mathcal{V}(H)$ which is exactly $G[\mathcal{V}(H)]$. This completes the proof for the claim. $\triangleleft$

By Claim 11, each H in Step 3 is the clique graph of a connected chordal graph. Therefore, all properties and lemmas studied in the previous section apply to H and each of its layer structures.

Now we show how to compute the table defined in Step 3. We fill the entries $D(i, H, K)$ in a decreasing order of the values of i.

- filling the base entries
  As $C^t(G)$ consists of pairwise nonadjacent nodes corresponding to maximal cliques of G, by the definition of the table, we set $D(t, H, K) = 1$ for all base entries.
- updating the table
  Now we delineate how to update other entries $D(i, H, K)$, assuming all entries $D(i + 1, H', K')$ have been computed. To compute the entry $D(i, H, K)$, we first compute the layer structure of H rooted by K. Let $\mathcal{LS}$ be the layer structure. By Definition 7, $\mathcal{LS}$ can be computed in polynomial time. Note that each unit in $\mathcal{LS}$ is a connected component of $C^{i+1}(G)$. For each vertex x in $\mathcal{V}(H)$, let U^x be the unit in $\mathcal{LS}$ which contains x and is at the least distance from the root of $\mathcal{LS}$. By Property 6, such a unit is unique. Besides, for each node K' of H, let $U^{K'}$ be the unit in $\mathcal{LS}$ containing K'. Let $R' = R|_{\mathcal{V}(H)}$ be R restricted to $\mathcal{V}(H)$. Let $\mathcal{Q}^{R'} = \{(U^x, U^y) \mid (x, y) \in R'\}$. Now we determine if there is a generic search ordering of $\mathcal{LS}$ which starts with the root and extends $\mathcal{Q}^{R'}$. This can be done in polynomial time [26, Theorem 6][1]. If this is not the case, by Claim 11 and Lemma 9, there is no MCS ordering of $G[\mathcal{V}(H)]$ that starts with vertices from K and extends R', and by Claim 11 and Lemma 8 there is no Prim ordering of H which starts with K and respects R'. So in this case, we set $D(i, H, K) = 0$. Otherwise, let $\pi = (U_0, U_1, \ldots, U_p)$ be a generic search ordering of $\mathcal{LS}$ extending $\mathcal{Q}^{R'}$ so that $U_0 = U^K$. Then, we let $D(i, H, K) = 1$ if and only if

---

[1] Scheffler [26] showed that the rooted version of the PSOP for the generic search can be solved in polynomial time. In this version, we are given a graph G, a partial order on $V(G)$, and a vertex $v \in V(G)$, and the question is whether G admits a generic search ordering which starts with v and extends the partial order.

(1) $D(i+1, U^K, K) = 1$, and

(2) for every U_j, $j \in [p]$, there exists a node K_j contained in U_j such that in H the node K_j is adjacent to at least one node from the parent of U_j in $\mathcal{LS}$ and, moreover, $D(i+1, U_j, K_j) = 1$.

We show the correctness of this step as follows. Observe that in every generic search ordering of $\mathcal{LS}$ starting from the root, every unit is visited before all its children.

We first prove the "if" direction. Assuming Conditions (1) and (2), let π' be the ordering obtained from π by

(a) realizing the first unit U^K by a Prim ordering of U^K which starts with K and respects R' restricted to U^K (guaranteed by $D(i+1, U^K, K) = 1$), and

(b) realizing every U_j, where $j \in [p]$, by a Prim ordering of U_j which starts with K_j and respects R' restricted to U_j (guaranteed by $D(i+1, U_j, K_j) = 1$).

The remainder of the proof for the "if" direction comprises the two claims below.

▷ **Claim 12.** π' is a Prim ordering of H with the first node being K.

Proof of Claim 12. By Condition (1) and Operation (a), we know that the first node in π' is K. Besides, from Condition (2) and Operation (b), for every U_j where $j \in [p]$ the first node of π' restricted to U_j is K_j. By the definition of Prim ordering (Algorithm 1) and the definition of π', it suffices now to show that for every U_j, $j \in [p]$, the following condition holds: K_j is adjacent to at least one node in H which is before K_j in π' and is from a different unit adjacent to U_j. This is the case as by Condition (2), K_j is adjacent to at least one node from the parent of U_j in $\mathcal{LS}$, and as π is a generic search ordering of $\mathcal{LS}$ with the root being the first unit, by Properties 3 and 4, the parent of U_j is before U_j in π, implying that all nodes in the parent of U_j are before all nodes of U_j in π'.   ◁

▷ **Claim 13.** π' respects R'.

Proof of Claim 13. To verify that π' respects R', let $(x, y) \in R'$. Due to Properties 3, 4, 7, and that π is a generic search ordering of the layer structure starting with the root, we know that for every $v \in V(G)$ the first node in π' containing v is from U^v, i.e., $K^v_{\pi'} \in \mathcal{K}(U^v)$. Our proof proceeds by distinguishing between the following two cases. If $U^x = U^y = U$, then as π has been realized by a Prim ordering of U respecting R' restricted to U in π', it holds that $K^x_{\pi'} <_{\pi'} K^y_{\pi'}$ or $K^x_{\pi'} = K^y_{\pi'}$. Otherwise, as π extends $\mathcal{Q}^{R'}$, U^x is before U^y in π. By the definition of π', maximal x-cliques in U^x are before maximal y-cliques in U^y. By Properties 3, 4, and 7, none of any units containing a maximal y-clique is before U^y in π. Then, from $K^x_{\pi'} \in \mathcal{K}(U^x)$ and $K^y_{\pi'} \in \mathcal{K}(U^y)$, it follows $K^x_{\pi'} <_{\pi'} K^y_{\pi'}$.   ◁

Now we give the proof for the "only if" direction. To this end, assume that $D(i, H, K) = 1$, i.e., H admits at least one Prim ordering, say π', which starts with K and respects R'. As π' respects R', for each unit U in the layer structure, π' restricted to U, i.e., $\pi'|_U$, is a Prim ordering of U respecting R' restricted to U. Consequently, $D(i+1, U, K') = 1$ where K' is the first node in $\pi'|_U$. This immediately implies that Condition (1) holds. We show below that Condition (2) also holds. Let U_j, $j \in [p]$, be a unit in $\mathcal{LS}$. Let K' be the first node in $\pi'|_{U_j}$. We claim that K' is adjacent in H to some node from the parent of U_j in $\mathcal{LS}$. As π' is a Prim ordering of H and K' is not the first node in π', K' is adjacent to at least one node, say $\hat{K}$, before K' in π' and, moreover, as K' is the first node in $\pi'|_{U_j}$, $\hat{K}$ is from a different unit, say $\hat{U}$. If $\hat{U}$ is the parent of U_j in $\mathcal{LS}$, we are done. Otherwise, by Lemma 5

and Definition 7, we know that nodes in each unit are consecutive in π'. As K is the first node of π' and K is contained in the root of $\mathcal{LS}$, by Properties 3 and 4, none of the nodes contained in any descendant of U_j is visited before K' in π'. It follows that $\hat{U}$ is in the same bag as U_j. Then, by Property 8, there is a node from the parent of U_j adjacent to K' in H.

The algorithm runs in polynomial time since the table has at most $O(n^2)$ entries, and computing the value of each entry can be done in polynomial time as described above.   ◄

References

1   Jesse Beisegel, Carolin Denkert, Ekkehard Köhler, Matjaž Krnc, Nevena Pivač, Robert Scheffler, and Martin Strehler. On the end-vertex problem of graph searches. *Discrete Mathematics & Theoretical Computer Science*, 21(1):Nr. 13, 2019. `doi:10.23638/DMTCS-21-1-13`.

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

3   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`.

4   Jean R. S. Blair and Barry W. Peyton. An introduction to chordal graphs and clique trees. In *Graph Theory and Sparse Matrix Computation*, pages 1–29. Springer, 1993. `doi:10.1007/978-1-4613-8369-7_1`.

5   Hans L. Bodlaender and Arie M. C. A. Koster. On the maximum cardinality search lower bound for treewidth. *Discrete Applied Mathematics*, 155(11):1348–1372, 2007. `doi:10.1016/j.dam.2007.02.004`.

6   Anna Bretscher, Derek Gordon 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`.

7   Peter Buneman. A characterisation of rigid circuit graphs. *Discrete Mathematics*, 9(3):205–212, 1974. `doi:10.1016/0012-365X(74)90002-8`.

8   Derek Gordon 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`.

9   Derek Gordon Corneil, Jérémie Dusart, Michel Habib, and Ekkehard Köhler. On the power of graph searching for cocomparability graphs. *SIAM Journal on Discrete Mathematics*, 30(1):569–591, 2016. `doi:10.1137/15M1012396`.

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

11   Derek Gordon 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`.

12   Edsger Wybe Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959. `doi:10.1007/BF01386390`.

13   Gabriel Andrew Dirac. On rigid circuit graphs. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 25:71–76, 1961. `doi:10.1007/BF02992776`.

14   Philippe Galinier, Michel Habib, and Christophe Paul. Chordal graphs and their clique graphs. In Manfred Nagl, editor, *WG*, volume 1017 of *Lecture Notes in Computer Science*, pages 358–371. Springer, 1995. `doi:10.1007/3-540-60618-1_88`.

15   Fǎnicǎ Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47–56, 1974. `doi:10.1016/0095-8956(74)90094-X`.

16   Jan Gorzny. *Related Orderings of AT-Free Graphs*. PhD thesis, University of Waterloo, 2022. URL: `https://uwspace.uwaterloo.ca/handle/10012/18079`.

17   Torben Hagerup. Biconnected graph assembly and recognition of DFS trees. Technical report, Universität des Saarlandes, 1985. `doi:10.22028/D291-26437`.

18   Torben Hagerup and Manfred Nowak. Recognition of spanning trees defined by graph searches. Technical report, Universität des Saarlandes, 1985.

19   Vojtěch Jarník. O jistém problému minimálním. *Práce Moravské Přírodovědecké Společnosti*, 6(4):57–63, 1930. URL: `https://dml.cz/bitstream/handle/10338.dmlcz/500726/Jarnik_01-0000-31_1.pdf`.

20   Ephraim Korach and Zvi Ostfeld. DFS tree construction: Algorithms and characterizations. In Jan van Leeuwen, editor, *WG*, volume 344 of *Lecture Notes in Computer Science*, pages 87–106. Springer, 1988. `doi:10.1007/3-540-50728-0_37`.

21   Arie M. C. A. Koster, Hans L. Bodlaender, and Stan P. M. van Hoesel. Treewidth: Computational experiments. *Electronic Notes in Discrete Mathematics*, 8:54–57, 2001. `doi:10.1016/S1571-0653(05)80078-2`.

22   P. Sreenivasa Kumar and C. E. Veni Madhavan. Minimal vertex separators of chordal graphs. *Discrete Applied Mathematics*, 89(1-3):155–168, 1998. `doi:10.1016/S0166-218X(98)00123-1`.

23   Robert Clay Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36(6):1389–1401, 1957. `doi:10.1002/j.1538-7305.1957.tb01515.x`.

24   Guozhen Rong, Yixin Cao, Jianxin Wang, and Zhifeng Wang. Graph searches and their end vertices. *Algorithmica*, 84(9):2642–2666, 2022. `doi:10.1007/s00453-022-00981-5`.

25   Donald J. Rose, Robert 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`.

26   Robert Scheffler. Linearizing partial search orders. In Michael A. Bekos and Michael Kaufmann, editors, *WG*, volume 13453 of *Lecture Notes in Computer Science*, pages 425–438. Springer, 2022. `doi:10.1007/978-3-031-15914-5_31`.

27   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`.

28   Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972. `doi:10.1137/0201010`.

29   Robert Endre 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`.

30   James Richard Walte. *Representations of Rigid Cycle Graphs*. PhD thesis, Wayne State University, 1972.

31   Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, second edition, 2001. URL: `https://openlibrary.org/books/OL6785828M/Introduction_to_graph_theory`.

32   Meibiao Zou, Zhifeng Wang, Jianxin Wang, and Yixin Cao. End vertices of graph searches on bipartite graphs. *Information Processing Letters*, 173:Nr. 106176, 2022. `doi:10.1016/j.ipl.2021.106176`.