






Well-Partitioned Chordal Graphs: Obstruction Set and Disjoint Paths

Jungho Ahn^{1,2} , Lars Jaffke³ , O-joung Kwon^{2,4} , and Paloma T. Lima³

¹ Department of Mathematical Sciences, KAIST, Daejeon, South Korea
junghoahn@kaist.co.kr

² Discrete Mathematics Group, IBS, Daejeon, South Korea

³ Department of Informatics, University of Bergen, Bergen, Norway
{lars.jaffke,paloma.lima}@uib.no

⁴ Department of Mathematics, Incheon National University, Incheon, South Korea
ojoungkwon@gmail.com

Abstract. We introduce a new subclass of chordal graphs that generalizes split graphs, which we call well-partitioned chordal graphs. Split graphs are graphs that admit a partition of the vertex set into cliques that can be arranged in a star structure, the leaves of which are of size one. Well-partitioned chordal graphs are a generalization of this concept in the following two ways. First, the cliques in the partition can be arranged in a tree structure, and second, each clique is of arbitrary size. We provide a characterization of well-partitioned chordal graphs by forbidden induced subgraphs, and give a polynomial-time algorithm that given any graph, either finds an obstruction, or outputs a partition of its vertex set that asserts that the graph is well-partitioned chordal. We demonstrate the algorithmic use of this graph class by showing that two variants of the problem of finding pairwise disjoint paths between k given pairs of vertices is in FPT parameterized by k on well-partitioned chordal graphs, while on chordal graphs, these problems are only known to be in XP. From the other end, we observe that there are problems that are polynomial-time solvable on split graphs, but become NP-complete on well-partitioned chordal graphs.

Keywords: Well-partitioned chordal graph · Chordal graph · Split graph · Disjoint paths · Forbidden induced subgraphs

1 Introduction

A central methodology in the study of the complexity of computationally hard graph problems is to impose additional structure on the input graphs,

J. Ahn and O. Kwon are supported by the Institute for Basic Science (IBS-R029-C1). O. Kwon is also supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Education (No. NRF-2018R1D1A1B07050294). L. Jaffke is supported by the Trond Mohn Foundation (TMS). P. T. Lima is supported by the Research Council of Norway via the project “CLASSIS”.

© Springer Nature Switzerland AG 2020

I. Adler and H. Müller (Eds.): WG 2020, LNCS 12301, pp. 148–160, 2020.

https://doi.org/10.1007/978-3-030-60440-0_12

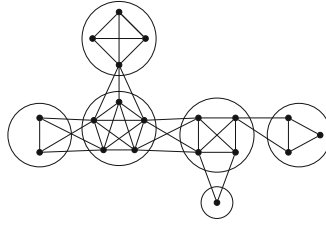


Fig. 1. A well-partitioned chordal graph.

and determine if this can be exploited in the design of an efficient algorithm. Typically, one restricts the input to be contained in a *graph class*, which is a set of graphs that share a common structural property. Following the establishment of the theory of NP-hardness, numerous problems were investigated in specific classes of graphs; either providing a polynomial-time algorithm for a problem Π on a specific graph class, while Π is NP-hard in a more general setting, or showing that Π remains NP-hard on a graph class. A key question in this field is to find for a given problem Π that is hard on a graph class \mathcal{A} , a subclass $\mathcal{B} \subsetneq \mathcal{A}$ such that Π is efficiently solvable on \mathcal{B} . Naturally, the goal is to narrow down the gap $\mathcal{A} \setminus \mathcal{B}$ as much as possible, and several notions of hardness/efficiency can be applied. For instance, we can require our target problem to be NP-hard on \mathcal{A} and polynomial-time solvable on \mathcal{B} ; or, from the viewpoint of parameterized complexity [6, 7], we require a target parameterized problem Π to be W[1]-hard on \mathcal{A} , while Π is in FPT on \mathcal{B} , or a separation in the kernelization complexity [8] of Π between \mathcal{A} and \mathcal{B} .

Chordal graphs are arguably one of the main characters in the algorithmic study of graph classes. They find applications for instance in computational biology [21] and sparse matrix computations [10]. Split graphs are an important subclass of chordal graphs. The complexities of computational problems on chordal and split graphs often coincide, however, this is not always the case. For instance, several variants of graph (vertex) coloring problems are polynomial-time solvable on split graphs and NP-hard on chordal graphs, see the works of Havet et al. [12], and of Silva [22]. Also, the SPARSEST k -SUBGRAPH [24] and DENSEST k -SUBGRAPH [5] problems are polynomial-time solvable on split graphs and NP-hard on chordal graphs. Other problems, for instance the TREE 3-SPANNER problem [3], are easy on split graphs, while their complexity on chordal graphs is still unresolved.

In this work, we introduce the class of *well-partitioned chordal graphs*, a subclass of chordal graphs that generalizes split graphs, which can be used as a tool for narrowing down complexity gaps for problems that are hard on chordal graphs, and easy on split graphs. The definition of well-partitioned chordal graphs is mainly motivated by a property of split graphs: the vertex set of a split graph can be partitioned into sets that can be viewed as a central clique of arbitrary size and cliques of size one that have neighbors only in the central clique. Thus, this partition has the structure of a star. Well-partitioned chordal

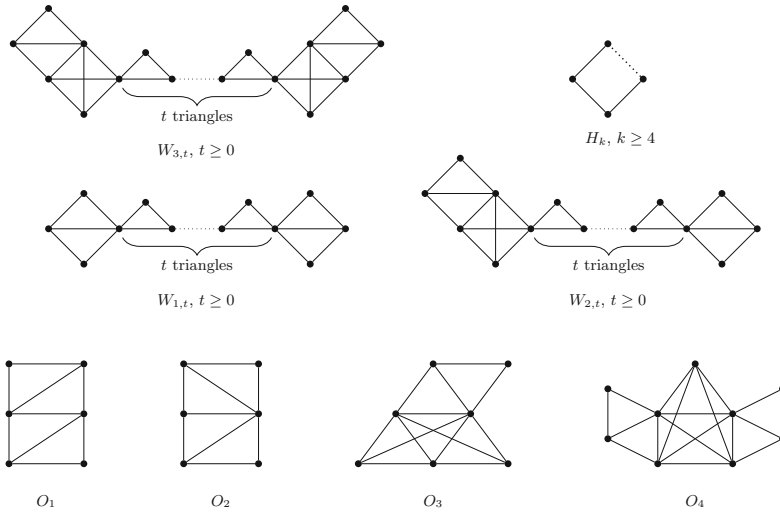


Fig. 2. The set of obstructions \mathbb{O} for well-partitioned chordal graphs.

graphs relax these ideas in two ways: by allowing the parts of the partition to be arranged in a tree structure instead of a star, and by allowing the cliques in each part to have arbitrary size. The interaction between adjacent parts P and Q remains simple: it induces a complete bipartite graph between a subset of P , and a subset of Q . Such a tree structure is called a partition tree, and we give an example of a well-partitioned chordal graph in Fig. 1. Now, it is not difficult to observe that the graphs constructed in the NP-hardness proofs in the works [12, 22] are in fact well-partitioned chordal graphs.

The main structural contribution of this work is a characterization of well-partitioned chordal graphs by forbidden induced subgraphs (see Fig. 2).

Theorem 1. *A graph is a well-partitioned chordal graph if and only if it has no induced subgraph isomorphic to a graph in \mathbb{O} . Furthermore, there is a polynomial-time algorithm that given a graph G , outputs either an induced subgraph of G isomorphic to a graph in \mathbb{O} , or a partition tree for each connected component which confirms that G is a well-partitioned chordal graph.*

Before we proceed with the discussion of the algorithmic results of this paper, we would like to briefly touch on the relationship of well-partitioned chordal graphs and width parameters. Each split graph is a well-partitioned chordal graph, and there are split graphs of whose *maximum induced matching width* (mim-width) depends linearly on the number of vertices [17]. This rules out the applicability of any algorithmic meta-theorem based on one of the common width parameters such as tree-width or clique-width, to the class of well-partitioned chordal graphs. It is known that mim-width is a lower bound for them [23].

Besides narrowing the complexity gap between the classes of chordal and split graphs, the class of well-partitioned chordal graphs can also be useful as

Table 1. Complexity of the DISJOINT PATHS and SET-RESTRICTED DISJOINT PATHS problems parameterized by the number k of terminal pairs. Size bounds for kernels are in terms of the number of *vertices* of the kernelized instances. *Given a partition tree.

Graph Class	DISJOINT PATHS	SET-RESTRICTED DISJOINT PATHS
Chordal	linear FPT [14]	XP [1]
Well-partitioned chordal	$\mathcal{O}(k^3)$ kernel [T. 5]	linear* FPT [T. 2, 3]
Split	$\mathcal{O}(k^2)$ kernel [13]	$\mathcal{O}(k^2)$ kernel [C. 1]

a step towards determining the yet unresolved complexity of a problem Π on chordal graphs when it is known that Π is easy on split graphs. This is the case in our current work. Specifically, we study the DISJOINT PATHS problem, formally defined as follows, and generalizations thereof. Two paths P_1 and P_2 are called *internally vertex-disjoint*, if for $i \in [2]$, no internal vertex of P_i is contained in P_{3-i} . (Note that this excludes the possibility that an endpoint of one path is used as an internal vertex in the other path.)

The classical DISJOINT PATHS problem takes as input a graph G and a set $\mathcal{X} = \{(s_1, t_1), \dots, (s_k, t_k)\}$ of k pairs of vertices of G , called *terminals*, and asks whether G contain k pairwise internally vertex-disjoint paths P_1, \dots, P_k such that for all $i \in [k]$, P_i is an (s_i, t_i) -path. This problem has already been shown by Karp to be NP-complete [15], and as a cornerstone result in the early days of fixed-parameter tractability theory, Robertson and Seymour showed that DISJOINT PATHS parameterized by k is in FPT [16, 19]. From the viewpoint of kernelization complexity, Bodlaender et al. showed that DISJOINT PATHS does not admit a polynomial kernel unless $\text{NP} \subseteq \text{coNP/poly}$ [2].

Restricting the problem to chordal and split graphs, Heggernes et al. showed that DISJOINT PATHS remains NP-complete on split graphs, and that it admits a polynomial kernel parameterized by k [13], and Kammer and Tholey showed that it has an FPT-algorithm with linear dependence on the size of the input chordal graph [14]. The question whether DISJOINT PATHS has a polynomial kernel on chordal graphs remains open. We go one step towards such a polynomial kernel, by showing that DISJOINT PATHS has a polynomial kernel on well-partitioned chordal graphs; generalizing the polynomial kernel on split graphs [13].

We also study a generalization of the DISJOINT PATHS problem, where in a solution, each path P_i can only use a restricted set of vertices U_i , which is specified for each terminal pair at the input. This problem was recently introduced by Belmonte et al. and given the name SET-RESTRICTED DISJOINT PATHS [1]. Since this problem contains DISJOINT PATHS as a special case (setting all domains equal to the whole vertex set), it is NP-complete. Belmonte et al. showed that SET-RESTRICTED DISJOINT PATHS parameterized by k is in XP on chordal graphs, and leave as an open question whether it is in FPT or $W[1]$ -hard on chordal graphs. Towards showing the former, we give an FPT-algorithm on well-partitioned chordal graphs. While we do not settle the kernelization complexity of SET-RESTRICTED DISJOINT PATHS on well-partitioned chordal graphs, we

observe that our FPT-algorithm implies a polynomial kernel on split graphs. We summarize these results in Table 1.

Finally, we also consider the SET-RESTRICTED DISJOINT CONNECTED SUBGRAPHS problem where we are given k terminal *sets* instead of pairs, and k domains, and the question is whether there are k pairwise disjoint connected subgraphs, each one connecting one of the terminal sets, using only vertices from the specified domain. This problem was also introduced in [1] and shown to be in XP on chordal graphs, when the parameter is the total number of vertices in all terminal sets. Extending our ideas of the above mentioned algorithms, we show that this problem is in fact FPT on well-partitioned chordal graphs.

Throughout, proofs of statements marked ‘♣’ are deferred to the full version.

2 Preliminaries

For a positive integer n , we let $[n] := \{1, 2, \dots, n\}$. All graphs considered here are simple and finite. For a graph G we denote by $V(G)$ and $E(G)$ the vertex set and edge set of G , respectively. Given $uv \in E(G)$, we call u and v its *endpoints*. Let G and H be two graphs. For a vertex v of a graph G , $N_G(v) := \{w \in V(G) \mid vw \in E(G)\}$ is the set of *neighbors* of v in G . The *degree* of v is $\deg_G(v) := |N_G(v)|$. The *subgraph induced by* X , denoted by $G[X]$, is the graph $(X, \{uv \in E(G) \mid u, v \in X\})$. We denote by $G - X$ the graph $G[V(G) \setminus X]$, and for a single vertex $x \in V(G)$, we use the shorthand ‘ $G - x$ ’ for ‘ $G - \{x\}$ ’. For two sets $X, Y \subseteq V(G)$, we denote by $G[X, Y]$ the graph $(X \cup Y, \{xy \in E(G) \mid x \in X, y \in Y\})$. We say that X is *complete to* Y if $X \cap Y = \emptyset$ and each vertex in X is adjacent to every vertex in Y . Let G be a graph. We say that G is *complete* if $uv \in E(G)$ for every $u, v \in V(G)$. A set $X \subseteq V(G)$ is a *clique* if $G[X]$ is complete. A graph G is *connected* if for each 2-partition (X, Y) of $V(G)$ with $X \neq \emptyset$ and $Y \neq \emptyset$, there is a pair $x \in X, y \in Y$ such that $xy \in E(G)$. A tree with at most one vertex of degree at least two is a *star*.

A *hole* in a graph G is an induced cycle of G of length at least 4. A graph is *chordal* if it has no hole as an induced subgraph. A vertex is *simplicial* if $N_G(v)$ is a clique. We say that a graph G has a *perfect elimination ordering* v_1, \dots, v_n if v_i is simplicial in $G[\{v_i, v_{i+1}, \dots, v_n\}]$ for each $i \in [n - 1]$. It is known that a graph is chordal if and only if it has a perfect elimination ordering [9]. A graph G is a *split graph* if there is a 2-partition (C, I) of $V(G)$ such that C is a clique and I is an independent set. For a family \mathcal{S} of subsets of some set, the *intersection graph* of \mathcal{S} is the graph on vertex set \mathcal{S} and edge set $\{ST \mid S, T \in \mathcal{S} \text{ and } S \cap T \neq \emptyset\}$.

3 Well-Partitioned Chordal Graphs

A connected graph G is a *well-partitioned chordal graph* if there exist a partition \mathcal{P} of $V(G)$ and a tree T having \mathcal{P} as a vertex set such that the following hold.

- (i) Each part $X \in \mathcal{P}$ is a clique in G .
- (ii) For each edge $XY \in E(\mathcal{T})$, there are subsets $X' \subseteq X$ and $Y' \subseteq Y$ such that $E(G[X, Y]) = \{xy \mid x \in X', y \in Y'\}$.
- (iii) For each pair of distinct $X, Y \in V(\mathcal{T})$ with $XY \notin E(\mathcal{T})$, $E(G[X, Y]) = \emptyset$.

The tree \mathcal{T} is called a *partition tree of G* , and the elements of \mathcal{P} are called its *bags*. A graph is a well-partitioned chordal graph if all of its connected components are well-partitioned chordal graphs. We remark that a well-partitioned chordal graph can have more than one partition tree. Also, observe that well-partitioned chordal graphs are closed under taking induced subgraphs.

A useful concept when considering partition trees of well-partitioned chordal graphs is that of a *boundary of a bag*. Let \mathcal{T} be a partition tree of a well-partitioned chordal graph G and let $X, Y \in V(\mathcal{T})$ be two bags that are adjacent in \mathcal{T} . The *boundary of X with respect to Y* , denoted by $\text{bd}(X, Y)$, is the set of vertices of X that have a neighbor in Y , i.e. $\text{bd}(X, Y) := \{x \in X \mid N_G(x) \cap Y \neq \emptyset\}$. By item (ii) of the definition of the class, $\text{bd}(X, Y)$ is complete to $\text{bd}(Y, X)$.

We now consider the relation between well-partitioned chordal graphs and other well-studied classes of graphs. It is easy to see that every well-partitioned chordal graph G is a chordal graph because every leaf of the partition tree of a component of G contains a simplicial vertex of G , and after removing this vertex, the remaining graph is still a well-partitioned chordal graph. Thus, we may construct a perfect elimination ordering. We show that, in fact, well-partitioned chordal graphs constitute a subclass of substar graphs. A graph is a *substar graph* [4] if it is an intersection graph of substars of a tree.

Proposition 1 (♣). *Every well-partitioned chordal graph is a substar graph.*

From the definition of well-partitioned chordal graphs, one can also see that every split graph is a well-partitioned chordal graph. Indeed, if G is a split graph with clique K and independent set S , the partition tree of G is a star, with the clique K as its central bag and each vertex of S contained in a different leaf bag. We show that, in fact, every starlike graph is a well-partitioned chordal graph. A *starlike graph* [11] is an intersection graph of substars of a star.

Proposition 2 (♣). *Every starlike graph is a well-partitioned chordal graph.*

We show that the graph O_1 in Fig. 2 is not a well-partitioned chordal graph. On the other hand, O_1 is a substar graph. Also a path graph on 5 vertices is a well-partitioned chordal graph but not a starlike graph. These observations with Propositions 1 and 2 show that we have the following hierarchy:

$$\text{split graphs} \subsetneq \text{starlike graphs} \subsetneq \text{well-partitioned chordal graphs} \subsetneq \text{substar graphs} \subsetneq \text{chordal graphs}$$

4 Characterization by Forbidden Induced Subgraphs

This section is entirely devoted to the proof of Theorem 1. That is, we show that the set \mathbb{O} of graphs depicted in Fig. 2 is the set of all forbidden induced subgraphs for well-partitioned chordal graphs, and give a polynomial-time recognition algorithm for this graph class. For convenience, we say that an induced subgraph of a graph that is isomorphic to a graph in \mathbb{O} is an *obstruction* for well-partitioned chordal graphs, or simply an obstruction.

Proposition 3 (\clubsuit). *The graphs in \mathbb{O} are not well-partitioned chordal graphs.*

In the rest, we outline the implementation of the algorithm, which also proves that the set \mathbb{O} is a complete set of forbidden induced subgraphs of well-partitioned chordal graphs.

Proposition 4. *Given a graph G , one can in polynomial time output either an obstruction in G or a partition tree of each connected component of G confirming that G is a well-partitioned chordal graph.*

We introduce the main concept in the algorithm, called a boundary-crossing path. Let G be a connected well-partitioned chordal graph with a partition tree \mathcal{T} . For a bag X of \mathcal{T} and $B \subseteq X$, a vertex $z \in V(G) \setminus X$ is said to *cross* B in X , if it has a neighbor both in B and in $X \setminus B$. In this case, we also say that B has a crossing vertex. In the following definitions, a path $X_1X_2 \dots X_\ell$ in \mathcal{T} is considered to be ordered from X_1 to X_ℓ . Let $\ell \geq 3$ be an integer. A path $X_1X_2 \dots X_\ell$ in \mathcal{T} is called a *boundary-crossing path* if for each $1 \leq i \leq \ell - 2$, there is a vertex in X_i that crosses $\text{bd}(X_{i+1}, X_{i+2})$. If for each $1 \leq i \leq \ell - 2$, there is no bag $Y \in V(\mathcal{T}) \setminus \{X_i\}$ containing a vertex that crosses $\text{bd}(X_{i+1}, X_{i+2})$, then we say the path is *exclusive*. If for each $1 \leq i \leq \ell - 2$, $\text{bd}(X_i, X_{i+1})$ is complete to X_{i+1} , then we say the path is *complete*. If a boundary-crossing path is both complete and exclusive, then we call it *good*. For convenience, we say that any path in \mathcal{T} with at most two bags is a boundary-crossing path.

The outline of the recognition algorithm is as follows. First we may assume that a given graph G is chordal, otherwise we find a hole in polynomial time [18]. We may also assume that G is connected. So, it has a simplicial vertex v , and by an inductive argument, we can assume that $G - v$ is a well-partitioned chordal graph. As v is simplicial, $G - v$ is also connected, and thus it admits a partition tree \mathcal{T} . If v has neighbors in one bag of \mathcal{T} , then we can simply put v as a new bag adjacent to that bag. Thus, we may assume that v has neighbors in two distinct bags, say C_1 and C_2 . Then our algorithm is divided into three parts:

1. We find a maximal good boundary-crossing path ending in C_2C_1 (or C_1C_2). To do this, given a good boundary-crossing path $C_iC_{i-1} \dots C_2C_1$, find a bag C_{i+1} containing a vertex crossing $\text{bd}(C_i, C_{i-1})$. If there is no such bag, then this path is maximal. Otherwise, we argue that in polynomial time either we can find an obstruction, or verify that $C_{i+1}C_i \dots C_2C_1$ is good.

2. Assume that $C_k C_{k-1} \dots C_2 C_1$ is the obtained maximal good boundary-crossing path. Then we can in polynomial time modify \mathcal{T} so that no vertex crosses $\text{bd}(C_2, C_1)$.
3. We show that if no vertex crosses $\text{bd}(C_2, C_1)$ and no vertex crosses $\text{bd}(C_1, C_2)$, then we can extend \mathcal{T} to a partition tree of G .

Steps 2 and 3 can be handled immediately. Step 1 is the most technically involved one. We first prove a handful of auxiliary lemmas that we can use to find pieces of obstructions in boundary-crossing paths that are not good, and puzzle them together. We separately deal with the following three cases, and in each case, we show that either one can in polynomial time find an obstruction or output a partition tree of G .

- (Lemma A) $C_1 \subseteq N_G(v)$.
- (Lemma B) $\text{bd}(C_1, C_2) \setminus N_G(v) \neq \emptyset$ and $C_2 \setminus N_G(v) \neq \emptyset$.
- (Lemma C) $C_1 \setminus N_G(v) \neq \emptyset$, $C_2 \setminus N_G(v) \neq \emptyset$ and $N_G(v) = \text{bd}(C_1, C_2) \cup \text{bd}(C_2, C_1)$.

In the proofs of these lemmas, we crucially use the aforementioned auxiliary lemmas that came out of our line of attack at Step 1 above. We sketch the idea of the proof of Lemma A.

Proof (Sketch of the proof of Lemma A). Since v is a simplicial vertex, we have that $\text{bd}(C_1, C_2) = C_1$. If $N_G(v) \cap C_2 = \text{bd}(C_2, C_1)$, then we can obtain a partition tree for G by adding v to C_1 . Thus, we may assume that $N_G(v) \cap C_2 \neq \text{bd}(C_2, C_1)$. Assume that $C_2 = \text{bd}(C_2, C_1)$. Since $\text{bd}(C_2, C_1)$ is complete to C_1 , we have that $C_1 \cup C_2$ is a clique. Hence, we can obtain a partition tree \mathcal{T}' for G from \mathcal{T} by removing C_1 and C_2 , adding a new bag $C^* = C_1 \cup C_2$, making all neighbors of C_1 and C_2 in \mathcal{T} adjacent to C^* , and adding a new bag $C_v := \{v\}$ and making C_v adjacent to C^* . Thus, we may assume that $C_2 \setminus \text{bd}(C_2, C_1) \neq \emptyset$.

Since $C_1 = \text{bd}(C_1, C_2)$, no vertex of $G - v$ crosses $\text{bd}(C_1, C_2)$. If no vertex of $G - v$ crosses $\text{bd}(C_2, C_1)$, then using Step 3, we can obtain a partition tree for G in polynomial time. Thus, we may assume that there is a bag C_3 having a vertex that crosses $\text{bd}(C_2, C_1)$. So, $C_3 C_2 C_1$ is a boundary-crossing path.

We find either an obstruction or a maximal good boundary-crossing path ending in $C_3 C_2 C_1$. First check whether $\text{bd}(C_3, C_2)$ is complete to C_2 . Otherwise, choose a vertex $p \in \text{bd}(C_3, C_2)$, and a non-neighbor q of p in C_2 . As p crosses $\text{bd}(C_2, C_1)$, p has a neighbor a in $C_2 \setminus \text{bd}(C_2, C_1)$ and a neighbor b in $\text{bd}(C_2, C_1)$. There are three possibilities; q is contained in one of $N_G(v) \cap C_2$, $\text{bd}(C_2, C_1) \setminus N_G(v)$, or $C_2 \setminus \text{bd}(C_2, C_1)$. In each case, we can find an obstruction. So, we may assume that $\text{bd}(C_3, C_2)$ is complete to C_2 . Next, we check if there exists another neighbor bag $D \neq C_3$ of C_2 having a vertex q that crosses $\text{bd}(C_2, C_1)$. In this case, we can find O_3 . Otherwise, $C_3 C_2 C_1$ is a good boundary-crossing path.

We now extend a given good boundary-crossing path $C_i C_{i-1} \dots C_2 C_1$ by recursively finding a bag C_{i+1} having a vertex crossing $\text{bd}(C_i, C_{i-1})$, and if the new sequence is not good, then we output an obstruction. This recursive step stops at some point, and we end up with a maximal good boundary-crossing

path $C_k C_{k-1} \cdots C_1$. Now, it is not difficult to see that replacing the sequence C_k, C_{k-1}, \dots, C_1 with $C_k \setminus \text{bd}(C_k, C_{k-1}), \text{bd}(C_k, C_{k-1}) \cup (C_{k-1} \setminus \text{bd}(C_{k-1}, C_{k-2}), \dots, \text{bd}(C_2, C_1) \cup C_1$ makes a new tree partition where no vertex crosses $\text{bd}(C'_2, C'_1)$ where C'_2 and C'_1 are the new last two bags. Then we can apply Step 3 to obtain a partition tree for the entire graph G in polynomial time. \square

Proof (of Proposition 4). We use the polynomial-time algorithm of [18] to find a hole¹ in G if one exists. We may assume that G is chordal. Since a graph is a well-partitioned chordal graph if and only if its connected components are well-partitioned chordal graphs, it is sufficient to show it for each connected component. From now on, we assume that G is connected. We can find a perfect elimination ordering (v_1, v_2, \dots, v_n) of G in polynomial time [20].

For each $i \in \{1, 2, \dots, n\}$, let $G_i := G[\{v_i, v_{i+1}, \dots, v_n\}]$. Observe that since G is connected and v_i is simplicial in G_i for all $1 \leq i \leq n-1$, each G_i is connected. From $i = n$ to 1, we recursively find either an obstruction or a partition tree of G_i . Clearly, G_n admits a partition tree. Let $1 \leq i \leq n-1$, and assume that we obtained a partition tree \mathcal{T} of G_{i+1} . Recall that v_i is simplicial in G_i .

Since v_i is simplicial in G_i , $N_{G_i}(v_i)$ is a clique. This implies that there are at most two bags in $V(\mathcal{T})$ that have a non-empty intersection with $N_{G_i}(v_i)$. If there is only one such bag in $V(\mathcal{T})$, say C , we can construct a partition tree for G_i by simply adding a bag consisting of v_i and making it adjacent to C .

Hence, from now on, we can assume that there are precisely two distinct adjacent bags $C_1, C_2 \in V(\mathcal{T})$ that have a non-empty intersection with $N_{G_i}(v_i)$. As $N_{G_i}(v_i)$ is a clique, we can observe that $N_{G_i}(v_i) \subseteq \text{bd}(C_1, C_2) \cup \text{bd}(C_2, C_1)$.

If $C_1 \subseteq N_{G_i}(v_i)$ or $C_2 \subseteq N_{G_i}(v_i)$, then by Lemma A, we can in polynomial time either output an obstruction or output a partition tree of G_i . Thus, we may assume that $C_1 \setminus N_{G_i}(v_i) \neq \emptyset$ and $C_2 \setminus N_{G_i}(v_i) \neq \emptyset$. If $\text{bd}(C_1, C_2) \setminus N_{G_i}(v_i) \neq \emptyset$ or $\text{bd}(C_2, C_1) \setminus N_{G_i}(v_i) \neq \emptyset$, then by Lemma B, we can in polynomial time either output an obstruction or output a partition tree of G_i . Thus, we may further assume that $\text{bd}(C_1, C_2) \setminus N_{G_i}(v_i) = \emptyset$ and $\text{bd}(C_2, C_1) \setminus N_{G_i}(v_i) = \emptyset$. Then by Lemma C, we can in polynomial time either output an obstruction or output a partition tree of G_i , and this concludes the proposition. \square

5 Algorithmic Applications

In this section, we give several FPT-algorithms and kernels for problems on well-partitioned chordal graphs. Specifically, we consider variants of the DISJOINT PATHS problem, called the SET-RESTRICTED DISJOINT PATHS and SET-RESTRICTED TOTALLY DISJOINT PATHS problems, where each path additionally has to be from a predefined domain. Recall that P_1 and P_2 are internally vertex-disjoint, if for $i \in [2]$, $(V(P_i) \setminus \{s_i, t_i\}) \cap V(P_{3-i}) = \emptyset$. Given a graph G , a set

¹ Note that holes in the sense of [18] are chordless cycles on at least five vertices; we can check for C_4 separately by brute force. While there are algorithms that verify chordality more directly, we use this procedure to fulfil the promise that we can always output an obstruction if there is one.

$\mathcal{X} = \{(s_1, t_1), \dots, (s_k, t_k)\}$ of k pairs of vertices of G , called *terminals*, and a set $\mathcal{U} = \{U_1, \dots, U_k\}$ of k vertex subsets of G , called *domains*, the SET-RESTRICTED DISJOINT PATHS problem asks if G contains k pairwise internally vertex-disjoint paths P_1, \dots, P_k such that for $i \in [k]$, P_i is an (s_i, t_i) -path with $V(P_i) \subseteq U_i$.

First, we can remove any adjacent terminal pair from the input, since we can always use the corresponding edge as the path in a solution. Next, we observe that finding pairwise internally vertex-disjoint paths is equivalent to finding pairwise internally vertex-disjoint *induced* paths. We call such a solution a *minimal* solution. We then use the following marking procedure. For each $i \in [k]$, we consider the path in \mathcal{T} that connects that bag containing s_i with the bag containing t_i . For each edge C_1C_2 on the path, we mark a maximal subset of $U_i \cap \text{bd}(C_1, C_2)$ of size at most $2k$, and a maximal subset of $U_i \cap \text{bd}(C_2, C_1)$ of size at most $2k$. We show that if our instance is a YES-instance, then it has some minimal solution that only uses marked vertices. We can therefore guess the intersection of such a solution with each bag, and we only have to consider its marked vertices. Formally, this is captured by the following notion.

Definition 1 (I-Feasible Bag). *Let $I \subseteq [k]$. Let $B \in V(\mathcal{T})$ be a bag and $M_i \subseteq V(G)$, $i \in I$, be sets of vertices. Then, we say that B is I -feasible w.r.t. $\{M_i \mid i \in I\}$, if there is a set $X \subseteq B$ and a labeling $\lambda: X \rightarrow [k]$ such that the following hold. For each $i \in I$ such that B lies on the path from the bag containing s_i to the bag containing t_i in \mathcal{T} , and each neighbor C of B on that path, either $\{s_i, t_i\} \cap \text{bd}(B, C) \neq \emptyset$, or there is a vertex $x_i \in X \cap M_i \cap \text{bd}(B, C)$ such that $\lambda(x_i) = i$. We use the shorthand ‘feasible’ for ‘ $[k]$ -feasible’.*

The algorithm works as follows. We apply the above marking procedure to obtain the marked sets M_1, \dots, M_k . Note that for each bag B , $|B \cap \bigcup_{i \in [k]} M_i| = \mathcal{O}(k^2)$: for each $i \in [k]$ we marked at most $4k$ vertices in B , and only if B lies on the path from the bag containing s_i to the bag containing t_i in \mathcal{T} . Then, for each bag $B \in V(\mathcal{T})$, we verify whether B is feasible w.r.t. M_1, \dots, M_k . If this is the case for all bags, then we conclude that we are dealing with a YES-instance, and otherwise, that we are dealing with a NO-instance.

Theorem 2 (♣). *There is an algorithm that solves each instance $(G, k, \mathcal{X}, \mathcal{U})$ of SET-RESTRICTED DISJOINT PATHS where G is a well-partitioned chordal graph given along with a partition tree \mathcal{T} , in time $2^{\mathcal{O}(k \log k)} \cdot n$.*

In the SET-RESTRICTED TOTALLY DISJOINT PATHS problem, we additionally require the paths in a solution to be pairwise distinct, i.e. if there is an edge xy in the graph and $\{s_i, t_i\} = \{s_j, t_j\} = \{x, y\}$, then only one of the paths P_i and P_j may consist of the edge xy . We call edges xy such that for some $w \geq 2$, $\{x, y\} = \{s_{i_1}, t_{i_1}\} = \dots = \{s_{i_w}, t_{i_w}\}$ a *heavy edge of weight w* . We call the indices i_1, \dots, i_w *heavy indices*. Instead of looking for minimal solutions, we look for *minimum* solutions, meaning that no other solution has fewer edges. In such a solution, in any chordal graph, the paths corresponding to a heavy edge of weight w are $w - 1$ paths of length two, and one path consisting only of the edge itself. For each such index i_j , either s_{i_j} and t_{i_j} are in a common bag, or they are

contained in the union of the boundaries of adjacent bags. In the former case, the middle vertex of a length two path may be from the bag itself, or from one of its neighbors, if both terminals are in the boundary to that neighbor. In the latter case, the middle vertex has to be in the union of the boundaries.

These observations allow for an adaption of the marking procedure to take into account heavy indices; the remaining indices can be treated as before. The algorithm works as follows. First, we apply the adapted marking procedure to obtain M_1, \dots, M_k . Then, we guess the part of the solution corresponding to heavy indices, again among the marked vertices. Let I be the indices that are not heavy. It then suffices to check whether for one of these guesses, all bags are I -feasible w.r.t. $\{M_i \mid i \in I\}$. If we have a successful guess, then we conclude that we have a YES-instance, and otherwise, that we have a NO-instance.

Theorem 3 (♣). *There is an algorithm that solves each instance $(G, k, \mathcal{X}, \mathcal{U})$ of SET-RESTRICTED TOTALLY DISJOINT PATHS where G is a well-partitioned chordal graph given along with a partition tree \mathcal{T} , in time $2^{\mathcal{O}(k \log k)} \cdot n$.*

We then observe that the techniques used in the previous algorithms can solve the more general SET-RESTRICTED DISJOINT CONNECTED SUBGRAPHS problem on well-partitioned chordal graphs as well. Here, the parameter s is the sum of the sizes of all terminal sets.

Theorem 4 (♣). *There is an algorithm that solves each instance $(G, k, \mathcal{X}, \mathcal{U})$ of SET-RESTRICTED DISJOINT CONNECTED SUBGRAPHS where G is a well-partitioned chordal graph given with a partition tree \mathcal{T} , in time $2^{\mathcal{O}(s \log s)} \cdot n$.*

As a consequence of the marking procedures, we have the following polynomial kernels on split graphs.

Corollary 1 (♣). *SET-RESTRICTED DISJOINT PATHS and SET-RESTRICTED TOTALLY DISJOINT PATHS on split graphs admit kernels on $\mathcal{O}(k^2)$ vertices.*

Moreover, with two more reduction rules, we obtain polynomial kernels on well-partitioned chordal graphs. This can be seen as follows. The subgraph of the partition tree that only has bags with marked vertices has at most $2k$ degree one bags, and therefore $\mathcal{O}(k)$ bags of degree at least three. In the DISJOINT PATHS and TOTALLY DISJOINT PATHS problems, where we do not need to consider the domains of the paths, we can get rid of degree two bags that do not contain terminals as follows. If in such a degree two bags, the boundaries are large enough, then we can always bypass that bag in any solution. If one of the boundaries is too small, then no solution can pass through the bag.

Theorem 5 (♣). *DISJOINT PATHS and TOTALLY DISJOINT PATHS on well-partitioned chordal graphs parameterized by k admit kernels on $\mathcal{O}(k^3)$ vertices.*

6 Conclusions

In this paper, we introduced the class of *well-partitioned chordal graphs*, a subclass of chordal graphs that generalizes split graphs. We provided a characterization by a set of forbidden induced subgraphs which also gave a polynomial-time

recognition algorithm, together with algorithmic applications in variants of the DISJOINT PATHS problem. Another typical characterization of (subclasses of) chordal graphs is via vertex orderings. For instance, chordal graphs are famously characterized as the graphs admitting perfect elimination orderings [9]. It would be interesting to see if well-partitioned chordal graphs admit a concise characterization in terms of vertex orderings as well. While the degree of the polynomial in the runtime of our recognition algorithm is moderate, our algorithm does not run in linear time. We therefore ask if it is possible to recognize well-partitioned chordal graphs in linear time; and note that a characterization in terms of vertex orderings can be a promising step in this direction.

References

1. Belmonte, R., Golovach, P.A., Heggenes, P., Hof, P.V., Kamiński, M., Paulusma, D.: Detecting fixed patterns in chordal graphs in polynomial time. *Algorithmica* **69**(3), 501–521 (2014)
2. Bodlaender, H.L., Thomassé, S., Yeo, A.: Kernel bounds for disjoint cycles and disjoint paths. *Theoret. Comput. Sci.* **412**(35), 4570–4578 (2011)
3. Brandstädt, A., Dragan, F.F., Le, H.O., Le, V.B.: Tree spanners on chordal graphs: complexity and algorithms. *Theoret. Comput. Sci.* **310**(1–3), 329–354 (2004)
4. Chang, Y.W., Jacobson, M.S., Monma, C.L., West, D.B.: Subtree and substar intersection numbers. *Discret. Appl. Math.* **44**(1–3), 205–220 (1993)
5. Corneil, D.G., Perl, Y.: Clustering and domination in perfect graphs. *Discret. Appl. Math.* **9**(1), 27–39 (1984)
6. Cygan, M.: *Parameterized Algorithms*. Springer, Cham (2015). <https://doi.org/10.1007/978-3-319-21275-3>
7. Downey, R.G., Fellows, M.R.: *Fundamentals of Parameterized Complexity*. TCS. Springer, London (2013). <https://doi.org/10.1007/978-1-4471-5559-1>
8. Fomin, F.V., Lokshtanov, D., Saurabh, S., Zehavi, M.: *Kernelization*. Cambridge University Press, Cambridge (2019)
9. Fulkerson, D., Gross, O.: Incidence matrices and interval graphs. *Pac. J. Math.* **15**(3), 835–855 (1965)
10. George, A., Gilbert, J.R., Liu, J.W.: *Graph Theory and Sparse Matrix Computation*. IMA, vol. 56. Springer, New York (2012). <https://doi.org/10.1007/978-1-4613-8369-7>
11. Gustedt, J.: On the pathwidth of chordal graphs. *Discret. Appl. Math.* **45**(3), 233–248 (1993)
12. Havet, F., Sales, C.L., Sampaio, L.: b-coloring of tight graphs. *Discret. Appl. Math.* **160**(18), 2709–2715 (2012)
13. Heggenes, P., van’t Hof, P.V., van Leeuwen, E.J., Saei, R.: Finding disjoint paths in split graphs. *Theor. Comput. Syst.* **57**(1), 140–159 (2015)
14. Kammer, F., Tholey, T.: The k-disjoint paths problem on chordal graphs. In: Paul, C., Habib, M. (eds.) *WG 2009*. LNCS, vol. 5911, pp. 190–201. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11409-0_17
15. Karp, R.M.: On the computational complexity of combinatorial problems. *Networks* **5**(1), 45–68 (1975)
16. Kawarabayashi, K.I., Kobayashi, Y., Reed, B.: The disjoint paths problem in quadratic time. *J. Combin. Theor. Ser. B* **102**(2), 424–435 (2012)

17. Mengel, S.: Lower bounds on the mim-width of some graph classes. *Discret. Appl. Math.* **248**, 28–32 (2018)
18. Nikolopoulos, S.D., Palios, L.: Detecting holes and antiholes in graphs. *Algorithmica* **47**(2), 119–138 (2007)
19. Robertson, N., Seymour, P.D.: Graph minors. XIII. The disjoint paths problem. *J. Combin. Theor. Ser. B* **63**(1), 65–110 (1995)
20. Rose, D.J., Tarjan, R.E., Lueker, G.S.: Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.* **5**(2), 266–283 (1976)
21. Semple, C., Steel, M.: *Phylogenetics*. Oxford Lecture Series in Mathematics and its Applications, vol. 24. Oxford University Press, Oxford (2003)
22. Silva, A.: Graphs with small fall-spectrum. *Discret. Appl. Math.* **254**, 183–188 (2019)
23. Vatshelle, M.: *New Width Parameters of Graphs*. Ph.D. thesis, University of Bergen, Norway (2012)
24. Watrigant, R., Bougeret, M., Giroudeau, R.: Approximating the Sparsest k -subgraph in chordal graphs. *Theor. Comput. Syst.* **58**(1), 111–132 (2016)