Kernels for the Disjoint Paths Problem on Subclasses of Chordal Graphs

Juhi Chaudhary
Ben-Gurion University of the Negev, Beersheba, Israel

Harmender Gahlawat
Ben-Gurion University of the Negev, Beersheba, Israel

Michal Włodarczyk
University of Warsaw, Poland

Meirav Zehavi
Ben-Gurion University of the Negev, Beersheba, Israel

Abstract

Given an undirected graph G and a multiset of k terminal pairs $\mathcal{X}$, the Vertex-Disjoint Paths (VDP) and Edge-Disjoint Paths (EDP) problems ask whether G has k pairwise internally vertex-disjoint paths and k pairwise edge-disjoint paths, respectively, connecting every terminal pair in $\mathcal{X}$. In this paper, we study the kernelization complexity of VDP and EDP on subclasses of chordal graphs. For VDP, we design a $4k$ vertex kernel on split graphs and an $\mathcal{O}(k^2)$ vertex kernel on well-partitioned chordal graphs. We also show that the problem becomes polynomial-time solvable on threshold graphs. For EDP, we first prove that the problem is NP-complete on complete graphs. Then, we design an $\mathcal{O}(k^{2.75})$ vertex kernel for EDP on split graphs, and improve it to a $7k+1$ vertex kernel on threshold graphs. Lastly, we provide an $\mathcal{O}(k^2)$ vertex kernel for EDP on block graphs and a $2k+1$ vertex kernel for clique paths. Our contributions improve upon several results in the literature, as well as resolve an open question by Heggernes et al. [Theory Comput. Syst., 2015].

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Mathematics of computing → Graph algorithms

Keywords and phrases Kernelization, Parameterized Complexity, Vertex-Disjoint Paths Problem, Edge-Disjoint Paths Problem

Digital Object Identifier 10.4230/LIPIcs.IPEC.2023.10

Related Version *Full Version*: `https://doi.org/10.48550/arXiv.2309.16892` [5]

Funding This research was supported by the European Research Council (ERC) grant titled PARAPATH.

1 Introduction

The Vertex-Disjoint Paths (VDP) and Edge-Disjoint Paths (EDP) problems are fundamental routing problems, having applications in VLSI design and virtual circuit routing [20, 39, 44, 45]. Notably, they have been a cornerstone of the groundbreaking Graph Minors project of Robertson and Seymour [42], and several important techniques, including the *irrelevant vertex technique*, originated in the process of solving disjoint paths [42]. In VDP (respectively, EDP), the input is an undirected graph G and a multiset of terminal pairs $\mathcal{X} = \{(s_1, t_1), \dots, (s_k, t_k)\}$, and the goal is to find k pairwise internally vertex-disjoint (respectively, edge-disjoint) paths $P_1, \dots, P_k$ such that P_i is a path with endpoints s_i and t_i.

Both VDP and EDP are studied extensively, and have been at the center of numerous results in algorithmic graph theory [6, 15, 18, 22, 31, 34, 36, 48]. Karp [30] proved that VDP is NP-complete (attributing the result to Knuth), and a year later, Even, Itai, and

Table 1 Summary of the kernelization results of VDP and EDP parameterized by the number of occurrences of terminal pairs (k) on the subclasses of chordal graphs studied in this paper.

Graph Class	VDP	EDP
Well-partitioned Chordal	$\mathcal{O}(k^2)$ vertex kernel [Theorem 7]	OPEN
Split	$4k$ vertex kernel [Theorem 6]	$\mathcal{O}(k^{2.75})$ vertex kernel [Theorem 2]
Threshold	P [Theorem 9]	$7k+1$ vertex kernel [Theorem 3]
Block	P [Observation 8]	$4k^2-2k$ vertex kernel [Theorem 4]
Clique Path	P [Observation 8]	$2k+1$ vertex kernel [Theorem 5]

Shamir [16] proved the same for EDP. When k is *fixed* (i.e., treated as a constant), Robertson and Seymour [37, 42] gave an $\mathcal{O}(|V(G)|^3)$ time algorithm as a part of their famous Graph Minors project. This algorithm is a *fixed-parameter tractable* (FPT) algorithm parameterized by k. Later, the power 3 was reduced to 2 by Kawarabayashi, Kobayashi, and Reed [32].

In Parameterized Complexity, each problem instance is associated with an integer parameter k. We study both VDP and EDP through the lens of *kernelization* under the parameterization by k. A *kernelization algorithm* is a polynomial-time algorithm that takes as input an instance (I, k) of a problem and outputs an *equivalent instance* (I', k') of the same problem such that the size of (I', k') is bounded by some computable function $f(k)$. The problem is said to admit an $f(k)$ sized kernel, and if $f(k)$ is polynomial, then the problem is said to admit a polynomial kernel. It is known that a problem is FPT if and only if it admits a kernel. Due to its profound impact, kernelization has been termed "*the lost continent of polynomial time*" [17]. For more details on kernelization, we refer to books [12, 19].

Bodlaender et al. [4] proved that, unless NP $\subseteq$ coNP/poly, VDP does not admit a polynomial kernel (on general graphs). On the positive side, Heggernes et al. [27] extended this study to show that VDP and EDP admit polynomial kernels on split graphs with $\mathcal{O}(k^2)$ and $\mathcal{O}(k^3)$ vertices, respectively. Yang et al. [47] further showed that a restricted version of VDP, where each vertex can appear in at most one terminal pair, admits a $4k$ vertex kernel. Recently, Ahn et al. [2] introduced so-called *well-partitioned chordal graphs* (being a generalization of split graphs), and showed that VDP on these graphs admits an $\mathcal{O}(k^3)$ vertex kernel. In this paper, we extend the study of kernelization of EDP and VDP on these (and other) subclasses of chordal graphs. We provide an extended survey in the Appendix.

1.1 Our Contribution

Proofs of the results marked with ($*$) are deferred to the full version [5] to respect the space constraints. An overview of our results is given in Table 1. We begin by discussing the results about EDP. First, we observe that the problem remains NP-hard even on inputs with a trivial graph structure given by a clique, unlike VDP. This extends the known hardness results for split graphs [27] and graphs of cliquewidth at most 6 [25]. We prove this theorem in the Appendix.

▶ **Theorem 1.** *EDP is* NP*-hard on complete graphs.*

Every graph class treated in this paper includes cliques, so EDP is NP-hard on each of them. This motivates the study of kernelization algorithms. From now on, we always use k to denote the number of occurrences of terminal pairs in an instance.

We present an $\mathcal{O}(k^{2.75})$ vertex kernel for EDP on split graphs, improving upon the $\mathcal{O}(k^3)$ vertex kernel given by Heggernes et al. [27]. Our main technical contribution is a lemma stating that the length of each path in a minimum-size solution is bounded by $\mathcal{O}(\sqrt{k})$. This allows us to obtain the following.

▶ **Theorem 2.** *EDP on split graphs admits a kernel with at most $\mathcal{O}(k^{2.75})$ vertices.*

In the quest to achieve better bounds, we consider a subclass of split graphs. Specifically, we prove that EDP on threshold graphs admits a kernel with at most $7k+1$ vertices. Here, we exploit the known vertex ordering of threshold graphs that exhibits an inclusion relation concerning the neighborhoods of the vertices.

▶ **Theorem 3** (∗)**.** *EDP on threshold graphs admits a kernel with at most $7k+1$ vertices.*

Another important subclass of chordal graphs is the class of block graphs. For this case, we present a kernel with at most $4k^2-2k$ vertices. Our kernelization algorithm constructs an equivalent instance where the number of blocks can be at most $4k-2$, and each block contains at most k vertices. Thus, we have the following theorem.

▶ **Theorem 4.** *EDP on block graphs admits a kernel with at most $4k^2-2k$ vertices.*

Whenever a block has more than two cut vertices, decreasing the size of that block below $\mathcal{O}(k)$ becomes trickier. However, if we restrict our block graph to have at most two cut vertices per block – i.e., if we deal with *clique paths* – then this can be done. The key point in designing our linear kernel in clique paths is that, in the reduced instance, for each block B, the number of vertices in B is dictated by a linear function of the number of terminal pairs having at least one terminal vertex in B. So, we obtain a $2k+1$ vertex kernel for this class.

▶ **Theorem 5.** *EDP on clique paths admits a kernel with at most $2k+1$ vertices.*

Now, we switch our attention to kernelization algorithms for VDP. First, we give a $4k$ vertex kernel for VDP on split graphs. This resolves an open question by Heggernes et al. [27], who asked whether this problem admits a linear vertex kernel. For this purpose, we use the result by Yang et al. [47], who gave a $4k$ vertex kernel for a restricted variant of VDP, called VDP-Unique by us, where each vertex can participate in at most one terminal pair.

In order to obtain a linear vertex kernel for VDP, we give a parameter-preserving reduction to VDP-Unique. Our reduction relies on a non-trivial matching-based argument.

In this way, we improve upon the $4k^2$ vertex kernel given by Heggernes et al. [27] as well as generalize the result given by Yang et al. [47]. Specifically, we have the following theorem.

▶ **Theorem 6.** *VDP on split graphs admits a kernel with at most $4k$ vertices.*

Next, we give an $\mathcal{O}(k^2)$ vertex kernel for VDP on well-partitioned chordal graphs (see Definition 64). Ahn et al. [2] showed that VDP admits an $\mathcal{O}(k^3)$ vertex kernel on this class. We improve their bound by giving a marking procedure that marks a set of at most $\mathcal{O}(k^2)$ vertices in G, which “covers” some solution (if it exists). As a result, we arrive at the following theorem.

▶ **Theorem 7** (∗)**.** *VDP on well-partitioned chordal graphs admits a kernel with $\mathcal{O}(k^2)$ vertices.*

Unlike EDP, the VDP problem turns out easier on the remaining graph classes. In block graphs, for every terminal pair with terminals in different blocks, there is a unique induced path connecting these terminals (all internal vertices of this path are the cut vertices). After adding these paths to the solution, we end up with a union of clique instances, where VDP is solvable in polynomial time. This leads to the following observation about block graphs and its subclass, clique paths.

▶ **Observation 8.** *VDP on block graphs (and, in particular, on clique paths) is solvable in polynomial time.*

Finally, we identify a less restricted graph class on which VDP is polynomial-time solvable, namely the class of threshold graphs. This yields a sharp separation between split graphs and its subclass – threshold graphs – in terms of VDP.

▶ **Theorem 9** (∗). *VDP on thresholds graphs is solvable in polynomial time.*

1.2 Organization of the paper

We begin with formal preliminaries, where we gather information about the studied graph classes and the basic algorithmic tools. In Section 3, we prove the kernelization theorems for EDP, which are followed by the NP-hardness proof for EDP on cliques in Appendix C. Next, we cover the kernelization results for VDP in Section 4. We conclude in Section 5.

2 Preliminaries

For a positive integer ℓ, let $[\ell]$ denote the set $\{1, \ldots, \ell\}$. We provide the preliminaries concerning parameterized complexity and graph classes considered in the Appendix.

Graph Notations. All graphs considered in this paper are simple, undirected, and connected unless stated otherwise. Standard graph-theoretic terms not explicitly defined here can be found in [13]. For a graph G, let $V(G)$ denote its vertex set, and $E(G)$ denote its edge set. For a graph G, the subgraph of G induced by $S \subseteq V(G)$ is denoted by $G[S]$, where $G[S] = (S, E_S)$ and $E_S = \{xy \in E(G) \mid x, y \in S\}$. For two sets $X, Y \subseteq V(G)$, we denote by $G[X, Y]$ the subgraph of G with vertex set $X \cup Y$ and edge set $\{xy \in E(G) : x \in X, y \in Y\}$. The *open neighborhood* of a vertex v in G is $N_G(v) = \{u \in V(G) : uv \in E(G)\}$. The *degree* of a vertex v is $|N_G(v)|$, and it is denoted by $d_G(v)$. When there is no ambiguity, we do not use the subscript G in $N_G(v)$ and $d_G(v)$. A vertex v with $d(v) = 1$ is a *pendant vertex*. The *distance* between two vertices in a graph G is the number of edges in the shortest path between them. We use the notation $d(u, v)$ to represent the distance between two vertices u and v in a graph G (when G is clear from the context). For a graph G and a set $X \subseteq V(G)$, we use $G - X$ to denote $G[V(G) \setminus X]$, that is, the graph obtained from G by deleting X. In a graph G, two vertices u and v are *twins* if $N_G[u] = N_G[v]$.

An *independent set* of a graph G is a subset of $V(G)$ such that no two vertices in the subset have an edge between them in G. A *clique* is a subset of $V(G)$ such that every two distinct vertices in the subset are adjacent in G. Given a graph G, a *matching* M is a subset of edges of G that do not share an endpoint. The edges in M are called *matched edges*, and the remaining edges are called *unmatched edges*. Given a matching M, a vertex $v \in V(G)$ is *saturated* by M if v is incident on an edge of M, that is, v is an end vertex of some edge of M. Given a graph G, Max Matching is to find a matching of maximum cardinality in G.

▶ **Proposition 10** ([28]). *For a bipartite graph G, Max Matching can be solved in $\mathcal{O}(\sqrt{|V(G)|} \cdot |E(G)|)$ time.*

A path $P = (v_1, \ldots, v_n)$ is an *M-alternating path* if the edges in P are matched and unmatched alternatively with respect to M. If both the end vertices of an alternating path are unsaturated, then it is an *M-augmenting path.*

▶ **Proposition 11** ([3]). *A matching M is maximum if and only if there is no M-augmenting path in G.*

A path $P = (v_1, v_2, \ldots, v_n)$ on n vertices is a (v_1, v_n)-*path* and $\{v_2, \ldots, v_{n-1}\}$ are the *internal vertices* of P. Moreover, for a path $P = (v_1, v_2, \ldots, v_n)$, we say that P *visits* the vertices $\{v_1, v_2, \ldots, v_n\}$. Throughout this paper, let P_{uv} denote the path containing only the edge uv. Let P_1 be an (s_1, t_1)-path and P_2 be an (s_2, t_2)-path. Then, P_1 and P_2 are *vertex-disjoint* if $V(P_1) \cap V(P_2) = \emptyset$. Moreover, P_1 and P_2 are *internally vertex-disjoint* if $(V(P_1) \setminus \{s_1, t_1\}) \cap V(P_2) = \emptyset$ and $(V(P_2) \setminus \{s_2, t_2\}) \cap V(P_1) = \emptyset$, that is, no internal vertex of one path is used as a vertex on the other path, and vice versa. Two paths are said to be *edge-disjoint* if they do not have any edge in common. Note that two internally vertex-disjoint paths are edge-disjoint, but the converse may not be true. A path P is *induced* if $G[V(P)]$ is the same as P. For a path $P = (v_1, \ldots, v_n)$ on n vertices and vertices $v_i, v_j \in V(P)$, let (v_i, v_j)-*subpath of* P denote the subpath of P with endpoints v_i and v_j.

Problem Statements. Given a graph G and a set (or, more generally, an ordered multiset) $\mathcal{X}$ of pairs of distinct vertices in G, we refer to the pairs in $\mathcal{X}$ as *terminal pairs*. A vertex in G is a *terminal vertex* if it appears in at least one terminal pair in $\mathcal{X}$ (when $\mathcal{X}$ is clear from context); else, it is a *non-terminal vertex*. For example, if G is a graph with $V(G) = \{v_1, v_2, \ldots, v_6\}$, and $\mathcal{X} = \{(v_1, v_3), (v_2, v_3), (v_3, v_6), (v_1, v_6)\}$ is a set of terminal pairs, then $\{v_1, v_2, v_3, v_6\}$ are terminal vertices in G and $\{v_4, v_5\}$ are non-terminal vertices in G. Formally, the definitions of VDP and EDP are given below.

VERTEX-DISJOINT PATHS (VDP):
Input: A graph G and an ordered multiset $\mathcal{X} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of k terminal pairs.
Question: Does G contain k distinct and pairwise internally vertex-disjoint paths $P_1, \ldots, P_k$ such that for all $i \in [k]$, P_i is an (s_i, t_i)-path?

EDGE-DISJOINT PATHS (EDP):
Input: A graph G and an ordered multiset $\mathcal{X} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of k terminal pairs.
Question: Does G contain k pairwise edge-disjoint paths $P_1, \ldots, P_k$ such that for all $i \in [k]$, P_i is an (s_i, t_i)-path?

▶ Remark 12. Note that in both problems (VDP and EDP), we allow different terminal pairs to intersect, that is, it may happen that for $i \neq j, \{s_i, t_i\} \cap \{s_j, t_j\} \neq \emptyset$.

If there are two identical pairs $\{s_i, t_i\} = \{s_j, t_j\} = \{x, y\}$ in $\mathcal{X}$ and the edge xy is present in G then only one of the paths P_i, P_j can use the edge xy if we require them to be edge-disjoint. However, setting $P_i = P_j = (x, y)$ does not violate the condition of being internally vertex-disjoint. It is natural though and also consistent with the existing literature to impose the additional condition that all paths in a solution have to be pairwise distinct.

▶ Remark 13. Throughout this paper, we assume that the degree of every terminal vertex is at least the number of terminal pairs in which it appears. Else, it is trivially a No-instance.

Following the notation introduced in [2] and [27], we have the following definitions.

▶ **Definition 14.** *An edge $xy \in E(G)$ is* heavy *if for some $w \geq 2$, there exist pairwise distinct indices $i_1, \ldots, i_w$ such that for each $j \in [w]$, $\{x, y\} = \{s_{i_j}, t_{i_j}\}$. We call a terminal pair (s_i, t_i)* heavy *if $s_i t_i$ is a heavy edge; else, we call it* light*. Note that calling a terminal pair heavy or light only makes sense when the terminals in the pair have an edge between them.*

▶ **Definition 15** (Minimum Solution). *Let $(G, \mathcal{X}, k)$ be a Yes-instance of VDP or EDP. A solution $\mathcal{P} = \{P_1, \ldots, P_k\}$ for the instance $(G, \mathcal{X}, k)$ is* minimum *if there is no solution $\mathcal{Q} = \{Q_1, \ldots, Q_k\}$ for $(G, \mathcal{X}, k)$ such that $\sum_{i=1}^{k} |E(Q_i)| < \sum_{i=1}^{k} |E(P_i)|$.*

Since we deal with subclasses of chordal graphs, the following proposition is crucial for us.

▶ **Proposition 16** ([27]). *Let $(G, \mathcal{X}, k)$ be a Yes-instance of VDP such that G is a chordal graph, and let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be a minimum solution of $(G, \mathcal{X}, k)$. Then, for every path $P_i \in \mathcal{P}$, either P_i is an induced path or P_i is a path of length 2, and there exists a path $P_j \in \mathcal{P}$ of length 1 whose endpoints are the same as the endpoints of P_i.*

The next observation follows from Proposition 16.

▶ **Observation 17.** *Let $(G, \mathcal{X}, k)$ be a Yes-instance of VDP. If there is a terminal pair $(s, t) \in \mathcal{X}$ such that $st \in E(G)$, then P_{st} belongs to every minimum solution of $(G, \mathcal{X}, k)$.*

3 Kernelization Results on EDP

We begin with the analysis of the simplest scenario where the input graph is a clique. In this setting, EDP is still NP-hard (see Appendix C), but we show below that whenever the size of the clique is larger than the parameter k, then we always obtain a Yes-instance. This improves the bound in [27, Lemma 7] by a factor of 2, which will play a role in optimizing the constants in our kernels (particularly, the linear ones).

▶ **Lemma 18** ($*$). *Let $(G, \mathcal{X}, k)$ be an instance of EDP such that G is a clique. If $|V(G)| > k$, then $(G, \mathcal{X}, k)$ is a Yes-instance.*

The bound above is tight as one can construct a No-instance $(G, \mathcal{X}, k)$ where G is a clique and $|V(G)| = k$. Consider $\mathcal{X}$ comprising just k copies of some pair $\{u, v\}$. Since the degree of u is $k - 1$, there cannot be k edge-disjoint paths having u as their common endpoint.

If G is a split graph with more than k vertices in the clique and the degree of each terminal vertex is at least the number of terminals on it, then we can reduce such an instance to the setting of Lemma 18 by replacing each terminal v in the independent set with an arbitrary neighbor of v. As a consequence, we obtain the following corollary, being a quantitative improvement over [27, Lemma 8].

▶ **Corollary 19.** *Let $(G, \mathcal{X}, k)$ be an instance of EDP such that G is a split graph with split partition (C, I). If $|C| > k$ and the degree of each terminal vertex is at least the number of terminals on it, then $(G, \mathcal{X}, k)$ is a Yes-instance.*

3.1 A Subcubic Vertex Kernel for Split Graphs

In this section, we show that EDP on split graphs admits a kernel with $\mathcal{O}(k^{2.75})$ vertices. Let $(G, \mathcal{X}, k)$ be an instance of EDP where G is a split graph. Note that given a split graph G, we can compute (in linear time) a partition (C, I) of $V(G)$ such that C is a clique and I is an independent set [26]. We partition the set I into two sets, say, I_T and I_N, where I_T and I_N denote the set of terminal vertices and the set of non-terminal vertices in I, respectively.

To ease the presentation of mathematical calculations, for this section (Section 3.1), we assume that $k^{\frac{1}{4}}$ is a natural number. If this is not the case, then we can easily get a new equivalent instance that satisfies this condition in the following manner. Let $d = (\lceil k^{\frac{1}{4}} \rceil)^4 - k$ and $v \in C$. Now, we add d terminal pairs $\{(s_{i_1}, t_{i_1}), \ldots, (s_{i_d}, t_{i_d})\}$ and attach each of these terminals to v. Observe that this does not affect the size of our kernel ($\mathcal{O}(k^{2.75})$ vertices) since $(\lceil k^{\frac{1}{4}} \rceil)^4 = \mathcal{O}(k)$. Moreover, we assume that $k > 8$, as otherwise, we can use the FPT algorithm for EDP [42] to solve it in polynomial time.

Overview. Heggernes et al. [27] gave an $\mathcal{O}(k^3)$ vertex kernel for EDP on split graphs. In our kernelization algorithm (in this section), we use their algorithm as a preprocessing step. After the prepossessing step, the size of C and I_T gets bounded by $2k$ each, and the size of I_N gets bounded by $\mathcal{O}(k^3)$. Therefore, we know that the real challenge in designing an improved kernel for EDP on split graphs lies in giving a better upper bound on $|I_N|$.

Our kernelization algorithm makes a non-trivial use of a lemma (Lemma 25), which establishes that the length of each path in any minimum solution (of EDP on a split graph G) is bounded by $\mathcal{O}(\sqrt{k})$. This, in turn, implies that a minimum solution of EDP for split graphs contains $\mathcal{O}(k^{1.5})$ edges. Note that during the preprocessing step (i.e., the kernelization algorithm by Heggernes et al. [27]), for every pair of vertices in C, at most $4k+1$ vertices are reserved in I_N, giving a cubic vertex kernel. In our algorithm, we characterized those vertices (called *rich* by us) in C for which we need to reserve only $\mathcal{O}(k^{1.5})$ vertices in I_N. Informally speaking, a vertex $v \in C$ is *rich* if there are $\Omega(k^{0.75})$ vertices in C that are "reachable" from v, even if we delete all the edges used by a "small" solution (containing $\mathcal{O}(k^{1.5})$ edges). Then, we show that if two vertices are rich, then even if they do not have any common neighbors in I_N, there exist "many" ($\Omega(k^{1.5})$) edge-disjoint paths between them even after removing any $\mathcal{O}(k^{1.5})$ edges of G. Hence, for every rich vertex, we keep only those vertices in I_N that are necessary to make the vertex rich, that is, we keep $\mathcal{O}(k^{1.5})$ vertices in I_N for every rich vertex. Thus, all rich vertices in C contribute a total of $\mathcal{O}(k^{2.5})$ vertices in I_N. The vertices in C that are not rich are termed as *poor*. Finally, we establish that a poor vertex cannot have too many neighbors in I_N. More specifically, a poor vertex can have only $\mathcal{O}(k^{1.75})$ neighbors in I_N. So, even if we keep all their neighbors in I_N, we store a total of $\mathcal{O}(k^{2.75})$ vertices in I_N for the poor vertices. This leads us to the desired kernel.

3.1.1 A Bound on the Length of the Paths in a Minimum Solution

In this section, we prove that for a minimum solution $\mathcal{P}$ of an instance $(G, \mathcal{X}, k)$ of EDP where G is a split graph, each path $P \in \mathcal{P}$ has length at most $4\sqrt{k}+3$. We prove this bound by establishing that if there is a path of length $4\sqrt{k}+4$ in $\mathcal{P}$, then $\mathcal{P}$ contains at least $k+1$ paths, a contradiction. To this end, we need the concept of *intersecting edges* (see Definition 21) and *non-compatible edges* (see Definition 22). Now, consider the following remark.

▶ Remark 20. For ease of exposition, throughout this section (Section 3.1.1), we assume (without mentioning explicitly it every time) that $(G, \mathcal{X}, k)$ is a Yes-instance of EDP, where G is a split graph. Moreover, $\mathcal{P}$ denotes a (arbitrary but fixed) minimum solution of $(G, \mathcal{X}, k)$, and $P \in \mathcal{P}$ is a path such that P contains ℓ vertices, say, $v_1, \ldots, v_\ell$, from clique C. Moreover, without loss of generality, let $v_1, \ldots, v_\ell$ be the order in which these vertices appear in the path P from some terminal to the other. Note that if a path, say, P', in a split graph has length p (i.e. $|E(P')| = p$), then it contains at least $\lceil \frac{p}{2} \rceil$ vertices from C. Therefore, to bound the length of P by $\mathcal{O}(\sqrt{k})$, it suffices to bound the number of vertices of C in P by $\mathcal{O}(\sqrt{k})$.

Assuming the ordering $v_1, \ldots, v_\ell$ of the vertices in $V(P) \cap C$ along the path P, we have the following definitions.

▶ **Definition 21** (Intersecting Edges). *Consider two edges $e_i = v_i v_{i'}$ and $e_j = v_j v_{j'}$ such that $i, i', j, j' \in [\ell]$, and without loss of generality, assume that $i < i'$, $j < j'$, and $i \leq j$. Then, e_i and e_j are* non-intersecting *if $j \geq i'$; otherwise, they are* intersecting.

▶ **Definition 22** (Non-compatible Edges). *Two edges $e_1, e_2 \in E(G)$ are* non-compatible *if there does not exist a path $P' \in \mathcal{P} \setminus \{P\}$ (given $P \in \mathcal{P}$) such that $\{e_1, e_2\} \subseteq E(P')$. Moreover, a set of edges $\mathcal{S} = \{e_1, \ldots, e_p\} \subseteq E(G)$ is* non-compatible *if every distinct $e_i, e_j \in \mathcal{S}$ are non-compatible.*

Next, we show that, since P is a path in a minimum solution $\mathcal{P}$, most of the edges with both endpoints in $\{v_1, \dots, v_\ell\}$ are used by paths in $\mathcal{P}$ (otherwise, we get a contradiction to the fact that $\mathcal{P}$ is a minimum solution). In particular, we have the following lemma.

▶ **Lemma 23** (∗). *Each edge of the form v_iv_j, where $i, j \in [\ell]$ and $j \geq i+2$, is used by some path in $\mathcal{P} \setminus \{P\}$.*

Now, we show that if two edges are intersecting edges, then they are non-compatible.

▶ **Lemma 24** (∗). *Let $e_i = v_iv_{i'}$ and $e_j = v_jv_{j'}$ be two (distinct) intersecting edges. Then, e_i and e_j are non-compatible.*

Now, we present the main lemma of this section.

▶ **Lemma 25** (∗). *Let $(G, \mathcal{X}, k)$ be a Yes-instance of EDP where G is a split graph. Moreover, let $\mathcal{P}$ be a minimum solution of $(G, \mathcal{X}, k)$. Then, for every path $P \in \mathcal{P}$, $|E(P)| < 4\sqrt{k} + 4$.*

We have the following corollary as a consequence of Lemma 25 (since $k \geq 9$).

▶ **Corollary 26.** *Let $\mathcal{P}$ be a minimum solution of an instance $(G, \mathcal{X}, k)$ of EDP where G is a split graph. Then, $\sum_{P \in \mathcal{P}} |E(P)| \leq 5k^{1.5}$.*

3.1.2 An $\mathcal{O}(k^{2.75})$ Vertex Kernel for Split Graphs

In this section, we use Corollary 26 stating that there can be at most $5k^{1.5}$ edges in any minimum solution to design a subcubic ($\mathcal{O}(k^{2.75})$) vertex kernel for EDP on split graphs. We start with the following preprocessing step, which we apply only once to our input instance.

Preprocessing Step. First, we use the kernelization for EDP on split graphs provided by Heggernes et al. [27] as a preprocessing step. In their kernel, if $|C| \geq 2k$, then they report a Yes-instance (due to [27, Lemma 8]), and hence, assume that $|C| < 2k$. Due to Corollary 19, if $|C| > k$, then we have a Yes-instance, and hence we assume that $|C| \leq k$. Moreover, in their kernel, for any two vertices $u, w \in C$, $|N(u) \cap N(w) \cap I_N| \leq 4k + 1$ (i.e., u and w have at most $4k + 1$ common neighbors in I_N). Furthermore, there are no pendant vertices in I_N.

Next, we define a Marking Procedure, where we label the vertices in C as *rich* or *poor*. Furthermore, we partition the vertices in I_N into two sets, denoted U (read *unmarked*) and M (read *marked*), in the following manner.

Marking Procedure. Let $(G, \mathcal{X}, k)$ be an instance of EDP where G is a split graph.

```
1. $M \Leftarrow \emptyset$ and $U \Leftarrow I_N$. (Initially, all vertices in $I_N$ is unmarked.) Moreover, fix an ordering $v_1, \dots, v_{|C|}$ of the vertices of $C$.
2. For $1 \leq i \leq |C|$:
  2.1. $A_{v_i} \Leftarrow \emptyset$, $M_{v_i} \Leftarrow \emptyset$ (read marked for $v_i$), and $U_T = U$ (read unmarked temporary).
  2.2. For $1 \leq j \leq |C|$ such that $i \neq j$ and $|A_{v_i}| < 100k^{0.75}$:
    2.2.1. If $|N(v_i) \cap N(v_j) \cap U_T| \geq k^{0.75}$, then $A_{v_i} \Leftarrow A_{v_i} \cup \{v_j\}$. Moreover, select some (arbitrary) subset $M_{v_i,v_j} \subseteq N(v_i) \cap N(v_j) \cap U_T$ such that $|M_{v_i,v_j}| = k^{0.75}$. Then, $M_{v_i} \Leftarrow M_{v_i} \cup M_{v_i,v_j}$ and $U_T \Leftarrow U_T \setminus M_{v_i,v_j}$.
  2.3. If $|A_{v_i}| = 100k^{0.75}$, then label $v_i$ as rich. Moreover, $M \Leftarrow M \cup M_{v_i}$ and $U \Leftarrow U_T$.
  2.4. If $|A_{v_i}| < 100k^{0.75}$, then label $v_i$ as poor.
```

This completes our Marking Procedure.

▶ Remark 27. Note that the definition of *rich* and *poor* depends on the order in which our Marking Procedure picks and marks the vertices (i.e., being rich or poor is not an intrinsic property of the vertex itself). A different execution of the above procedure can label different vertices as rich and poor. Moreover, note that if $M_{v,x}$ exists (i.e., v is rich and $x \in A_v$) and $M_{x,v}$ exists (i.e., x is rich and $v \in A_x$), then $M_{x,v} \cap M_{v,x} = \emptyset$.

We have the following observation regarding the vertices marked rich by an execution of Marking Procedure on an instance $(G, \mathcal{X}, k)$ of EDP where G is a split graph.

▶ **Observation 28** (∗). *Consider an execution of Marking Procedure on an instance $(G, \mathcal{X}, k)$ of EDP where G is a split graph. Then for a rich vertex v, $|M_v| = 100k^{1.5}$ (i.e., the number of vertices marked in I_N for v are $100k^{1.5}$).*

▶ **Definition 29** (Reachable Vertices). *Consider an execution of Marking Procedure on an instance $(G, \mathcal{X}, k)$ of EDP where G is a split graph. Moreover, let $\mathcal{P}$ be a solution of $(G, \mathcal{X}, k)$. Then, for a rich vertex $v \in C$, let $R_v \subseteq A_v$ (read* reachable from v*) denote the set of vertices such that for each vertex $x \in R_v$, there is a vertex $u \in M_{v,x}$ such that u is not used by any path in $\mathcal{P}$.*

Notice that, in Definition 29, a path of the form (v, u, x) is edge-disjoint from every path in $\mathcal{P}$. Furthermore, we briefly remark that R_v is defined with respect to the execution of Marking Procedure and a solution $\mathcal{P}$ of $(G, \mathcal{X}, k)$, which we will always fix before we use R_v. Let $\mathcal{P}$ be a solution to an instance $(G, \mathcal{X}, k)$ of EDP. Informally speaking, in the following lemma, we show that if $\mathcal{P}$ uses at most $6k^{1.5}$ edges, then for a rich vertex v, $R_v = \Omega(k^{0.75})$ (i.e., there are "many reachable" vertices in A_v from v using paths that are edge-disjoint from every path in $\mathcal{P}$). In particular, we have the following lemma.

▶ **Lemma 30** (∗). *Consider an execution of Marking Procedure on an instance $(G, \mathcal{X}, k)$ of EDP where G is a split graph. Moreover, let $\mathcal{P}$ be a solution of $(G, \mathcal{X}, k)$ (not necessarily minimum) such that the total number of edges used in $\mathcal{P}$ is at most $6k^{1.5}$. Then, for any rich vertex $v \in C$, $|R_v| \geq 94k^{0.75}$.*

Next, we provide the following reduction rule.

▶ **Reduction Rule 1** (RR1). *Let $(G, \mathcal{X}, k)$ be an instance of EDP where G is a split graph. Let U be the set of unmarked vertices we get after an execution of Marking Procedure on $(G, \mathcal{X}, k)$. Moreover, let $U' \subseteq U$ be the set of vertices in U that do not have a poor neighbor. If $U' \neq \emptyset$, then $G' \Leftarrow G - U'$ and $\mathcal{X}' \Leftarrow \mathcal{X}$.*

The following two lemmas (Lemmas 31 and 32) are essential to prove the safeness of RR1.

▶ **Lemma 31** (∗). *Let $(G, \mathcal{X}, k)$ be an instance of EDP where G is a split graph. Consider an execution of Marking Procedure on $(G, \mathcal{X}, k)$. Moreover, let $\mathcal{P}$ be a solution of $(G, \mathcal{X}, k)$ (not necessarily minimum) such that the total number of edges used in $\mathcal{P}$ is ℓ, where $\ell \leq 6k^{1.5}$. Furthermore, let $u \in U$ be an unmarked vertex such that u does not have any poor neighbor. If there is a path $P \in \mathcal{P}$ such that $u \in V(P)$, then there exists a solution $\mathcal{P}' = (\mathcal{P} \setminus \{P\}) \cup \{P'\}$ of $(G, \mathcal{X}, k)$ such that $u \notin V(P')$, and the total number of edges in $\mathcal{P}'$ is at most $\ell + 3$.*

▶ **Lemma 32** (∗). *Let $(G, \mathcal{X}, k)$ be an instance of EDP where G is a split graph. Let U be the set of unmarked vertices we get after an execution of Marking Procedure on $(G, \mathcal{X}, k)$, and let $u \in U$ be a vertex such that u does not have any poor neighbor. Let $G' = G - \{u\}$. Then, $(G, \mathcal{X}, k)$ is a Yes-instance if and only if $(G', \mathcal{X}, k)$ is a Yes-instance.*

Since G' is a split graph (due to Remark 67), Lemma 32 implies the following.

▶ **Lemma 33.** *RR1 is safe.*

Next, we show that an exhaustive application of RR1 provides a subcubic vertex kernel.

▶ **Lemma 34** (∗). *Let $(G, \mathcal{X}, k)$ be an instance of EDP where G is a split graph. If we cannot apply RR1 on $(G, \mathcal{X}, k)$, then $|V(G)| = \mathcal{O}(k^{2.75})$.*

Finally, due to Preprocessing Step, RR1, Lemmas 33 and 34, and the observation that RR1, Marking Procedure, and Preprocessing Step can be executed in polynomial time (and do not increase our parameter k) we have the following theorem.

▶ **Theorem 2.** *EDP on split graphs admits a kernel with at most $\mathcal{O}(k^{2.75})$ vertices.*

3.2 A Quadratic Vertex Kernel for Block Graphs

In this section, we show that EDP on block graphs admits a kernel with at most $4k^2 - 2k$ vertices. Let us first discuss the overall idea leading us to this result.

Overview. Let $(G, \mathcal{X}, k)$ be an instance of EDP where G is a block graph. First, we aim to construct a reduced instance where the number of blocks can be at most $4k - 2$. We begin by showing that if there is an end block that does not contain any terminal, then we can delete this block from the graph (in RR2), while preserving all solutions. Next, we argue that if there is a block, say, B, with at most two cut vertices that do not contain any terminal, then we can either *contract* (defined in Definition 37) B to a single vertex, or answer negatively (in RR4). Thus, each block with at most two cut vertices in the (reduced) graph contains at least one terminal. This bounds the number of blocks with at most two cut vertices to be at most $2k$ (as k terminal pairs yield at most $2k$ terminals). First, we observe the following.

▶ **Observation 35** (∗). *Let ℓ be the number of end blocks in a block graph G. Then, the number of blocks with at least three cut vertices is at most $\ell - 2$.*

Observation 35, along with the fact that the number of end blocks is at most $2k$, establishes that the number of blocks with at least three cut vertices in the (reduced) graph is at most $2k - 2$. Therefore, we have at most $4k - 2$ blocks in the (reduced) graph. Finally, due to Lemma 18 and the properties of block graphs, we show that if a block, say, B, is big enough (i.e., $|V(B)| > k$), then we can contract B to a single vertex while preserving the solutions (in RR3). Hence, each block contains at most k vertices, and thus, the total number of vertices in the (reduced) graph is at most $4k^2 - 2k$.

Our kernelization algorithm is based on the application of three reduction rules (RR2-RR4), discussed below, to an instance $(G, \mathcal{X}, k)$ of EDP where G is a block graph.

▶ **Reduction Rule 2** (RR2). *If B is an end block of G with cut vertex v such that B does not contain any terminal, then $G' \Leftarrow G[V(G) \setminus (V(B) \setminus \{v\})]$ and $\mathcal{X}' \Leftarrow \mathcal{X}$.*

We have the following lemma to establish that RR2 is safe.

▶ **Lemma 36** (∗). *RR2 is safe.*

The following definitions (Definitions 37 and 38) are crucial to proceed further in this section. Informally speaking, we contract a block B by replacing it with a (new) vertex v such that "edge relations" are preserved. We have the following formal definition.

▶ **Definition 37** (Contraction of a Block). *Let $(G, \mathcal{X}, k)$ be an instance of EDP where G is a block graph, and let B be a block of G. The* contraction *of B in $(G, \mathcal{X}, k)$ yields another instance $(G', \mathcal{X}', k')$ of EDP as follows. First, $V(G') = (V(G) \setminus V(B)) \cup \{v\}$ (i.e., delete $V(B)$ and add a new vertex v). Moreover, define $f : V(G) \to V(G')$ such that $f(x) = x$ if $x \in V(G) \setminus V(B)$, and $f(x) = v$ if $x \in V(B)$. Second, $E(G') = \{f(x)f(y) : xy \in E(G),\ f(x) \neq f(y)\}$. Similarly, $\mathcal{X}' = \{(f(s), f(t)) : (s,t) \in \mathcal{X},\ f(s) \neq f(t)\}$. Finally, let $k' = |\mathcal{X}'|$. Note that k' might be smaller than k (in case B contains a terminal pair). Moreover, $\bigcup_{u \in V(B)}(N_G(u) \setminus B) \subseteq N_{G'}(v)$.*

We will exploit the properties of block graphs to show that if a block B has at least $k+1$ vertices, then we can contract B to a single vertex "safely". To this end, we define the instance $(G, \mathcal{X}, k)$ of EDP restricted to a block B of the block graph G as follows.

▶ **Definition 38** (Restriction of an Instance $(G, \mathcal{X}, k)$ to a Block). *Consider a block B whose set of cut vertices is U. For each cut vertex $u \in U$, let $C_{B,u}$ denote the (connected) component of $G[V(G) \setminus (V(B) \setminus \{u\})]$ containing u. Now, define $h : V(G) \to V(B)$ such that $h(x) = x$ if $x \in V(B)$, and $h(x) = u$ if $x \in V(C_{B,u})$. Then, the* restriction of $(G, \mathcal{X}, k)$ to B, *denoted by $(B, \mathcal{X}_B, |\mathcal{X}_B|)$, is defined as follows: $\mathcal{X}_B = \{(h(s), h(t)) : (s,t) \in \mathcal{X},\ h(s) \neq h(t)\}$.*

We have the following trivial observation that we will use for our proofs.

▶ **Observation 39.** *Let $\mathcal{P}$ be a set of edge-disjoint paths. Consider a set of paths $\mathcal{P}^*$ constructed in the following manner. For each path $P \in \mathcal{P}$, add a path P^* to $\mathcal{P}^*$ such that $E(P^*) \subseteq E(P)$. Then, $\mathcal{P}^*$ is also a set of edge-disjoint paths.*

We have the following lemma.

▶ **Lemma 40** (∗). *Let $(G, \mathcal{X}, k)$ be an instance of EDP on a block graph G, and let B be a block of G. If $(G, \mathcal{X}, k)$ is a Yes-instance, then $(B, \mathcal{X}_B, |\mathcal{X}_B|)$ is a Yes-instance.*

Next, we will show that if for a block B, $(B, \mathcal{X}_B, |\mathcal{X}_B|)$ is a Yes-instance, then we can contract B "safely". In particular, we have the following lemma.

▶ **Lemma 41** (∗). *Let $(G, \mathcal{X}, k)$ be an instance of EDP on a block graph G, and let B be a block of G whose set of cut vertices is U. Moreover, let the contraction of B in $(G, \mathcal{X}, k)$ yield $(G', \mathcal{X}', k')$. Given that $(B, \mathcal{X}_B, |\mathcal{X}_B|)$ is a Yes-instance, $(G, \mathcal{X}, k)$ is a Yes-instance if and only if $(G', \mathcal{X}', k')$ is a Yes-instance.*

Next, we have the following reduction rule.

▶ **Reduction Rule 3** (RR3). *If G has a block B such that $|V(B)| > k$, then contract B in $(G, \mathcal{X}, k)$ to get $(G', \mathcal{X}', k')$.*

The safeness of RR3 is implied by Lemma 41 and Lemma 18.

▶ **Corollary 42.** *RR3 is safe.*

Finally, we have the following reduction rule.

▶ **Reduction Rule 4** (RR4). *Let B be a block of G that has exactly two cut vertices, say, u and w, and there is no terminal vertex in $V(B) \setminus \{u, w\}$. Consider the instance $(B, \mathcal{X}_B, |\mathcal{X}_B|)$ restricted to the block B. If $|V(B)| > |\mathcal{X}_B|$, then contract B in $(G, \mathcal{X}, k)$ to get the instance $(G', \mathcal{X}', k')$. Else, answer negatively.*

We have the following lemma to establish that RR4 is safe.

▶ **Lemma 43** (∗). *RR4 is safe.*

Next, we establish that once we cannot apply RR2-RR4 anymore, the size of the reduced graph is bounded by a quadratic function of k. In particular, we have the following lemma.

▶ **Lemma 44** (∗). *Let $(G, \mathcal{X}, k)$ be an instance of EDP where we cannot apply reduction rules RR2-RR4 anymore. Then, G contains at most $4k - 2$ blocks and $4k^2 - 2k$ vertices.*

Observe that RR2-RR4 can be implemented in polynomial time and do not increase our initial parameter k. Hence, Lemmas 36, 43, 44, and Corollary 42 imply the following.

▶ **Theorem 4.** *EDP on block graphs admits a kernel with at most $4k^2 - 2k$ vertices.*

3.3 A Linear Vertex Kernel for Clique Paths

A *clique path* is a block graph where each block has at most two cut vertices. (In an informal manner, we can think that the blocks are arranged in the form of a path.) In this section, we present a linear vertex kernel for EDP on clique paths. First, we present an overview of the overall idea leading to this result.

Overview. Let $(G, \mathcal{X}, k)$ be an instance of EDP where G is a clique path. Our kernelization algorithm is based on the application of RR2-RR4 (defined in Section 3.2) along with three new reduction rules (RR5-RR7). In our kernel for block graph in Section 3.2, we established that for a block B, if $|V(B)| \geq k + 1$, then we can contract B (see RR3). Moreover, we showed that the total number of blocks in a reduced instance can be at most $4k - 2$, thus giving us an $\mathcal{O}(k^2)$ vertex kernel.

Here, we use the property of clique paths that each block can have at most two cut vertices to improve the kernel size. Since there is no block with more than two cut vertices, each block must contain a terminal after an exhaustive application of RR2-RR4. Let B be a block of G with cut vertices u and w. Consider the instance $(B, \mathcal{X}_B, |\mathcal{X}_B|)$, that is, the instance $(G, \mathcal{X}, k)$ restricted to the block B (see Definition 38). Any terminal pair $(s, t) \in \mathcal{X}_B$ is of one of the following types: (i) Type-A: $s, t \in \{u, w\}$, (ii) Type-B: $s = u$ and $t \in V(B) \setminus \{u, w\}$, or vice versa, (iii) Type-C: $s = w$ and $t \in V(B) \setminus \{u, w\}$, or vice versa, and (iv) Type-D: $s, t \in V(B) \setminus \{u, w\}$. Let a, b, c, and d denote the cardinality of Type-A, Type-B, Type-C, and Type-D occurrences of terminal pairs in $\mathcal{X}_B$, respectively. We show that if $|V(B)| > d + 2 + \max\{b + c - 1, 0\}$, then we can either contract B to a single vertex "safely" (when $|V(B)| > \max\{a + b, a + c\}$) or report a No-instance. Summing these numbers over all blocks yields an upper bound on the total size of the reduced instance. The Type-A pairs are irrelevant now, each pair of Type-B or Type-C contributes to two blocks, while each Type-D pair appears in only a single block. By grouping the summands in an appropriate way, we are able to attain a bound of $2k + 1$. We have the following reduction rules.

▶ **Reduction Rule 5** (RR5). *Let $(G, \mathcal{X}, k)$ be an instance of EDP where G is a clique path. Moreover, let B be a block of G such that B has two cut vertices, say, u and w. Consider the instance $(B, \mathcal{X}_B, |\mathcal{X}_B|)$. If $|V(B)| \leq \max\{a + b, a + c\}$, then report a No-instance.*

We have the following lemma to prove that RR5 is safe.

▶ **Lemma 45** (∗). *RR5 is safe.*

Next, we have the following reduction rule.

▶ **Reduction Rule 6** (RR6). *Let $(G, \mathcal{X}, k)$ be an instance of EDP where G is a clique path. Moreover, let B be a block of G that has two cut vertices, say, u and w. Consider the instance $(B, \mathcal{X}_B, |\mathcal{X}_B|)$. If $|V(B)| > d + 2 + \max\{b + c - 1, 0\}$, then contract B in $(G, \mathcal{X}, k)$ to obtain the instance $(G', \mathcal{X}', k')$.*

We have the following lemma to establish that RR6 is safe.

▶ **Lemma 46** ($*$). *RR6 is safe.*

In RR6, we showed that in a reduced instance, the size of each block with two cut vertices is bounded by a linear function of the number of times its vertices appear in some occurrences of terminal pairs. In the next reduction rule (RR7), we consider the end blocks (i.e., blocks with exactly one cut vertex). So, consider an end block B of G with cut vertex u, and let $(B, \mathcal{X}_B, |\mathcal{X}_B|)$ be the restriction of $(G, \mathcal{X}, k)$ to B. Any terminal pair $(s, t) \in \mathcal{X}_B$ is one of the following types: (i) Type-B$'$: $s = u$ and $t \in V(B) \setminus \{u\}$, or vice versa, and (ii) Type-D$'$: $s, t \in V(B) \setminus \{u\}$. Let b' and d' denote the number of occurrences of Type-B$'$ and Type-D$'$ terminal pairs in $\mathcal{X}_B$, respectively. We have the following reduction rule.

▶ **Reduction Rule 7** (RR7). *Let $(G, \mathcal{X}, k)$ be an instance of EDP where G is a clique path. Let B be an end block of G with cut vertex u. Consider the instance $(B, \mathcal{X}_B, |\mathcal{X}_B|)$. If $|V(B)| > b' + d'$, then contract B in $(G, \mathcal{X}, k)$ to obtain $(G', \mathcal{X}', k')$.*

We have the following lemma to prove that RR7 is safe.

▶ **Lemma 47** ($*$). *RR7 is safe.*

Next, we establish that once we cannot apply RR2-RR7, the number of vertices of the reduced graph is bounded by a linear function of k in the following lemma.

▶ **Lemma 48** ($*$). *Let $(G, \mathcal{X}, k)$ be an instance of EDP where G is a clique path. If we cannot apply RR2-RR7 on this instance, then G contains at most $2k + 1$ vertices.*

Now, we have the following observation.

▶ **Observation 49.** *RR5-RR7 can be applied in polynomial time. Moreover, during the application of RR5-RR7, we never increase the initial k.*

Finally, due to RR2-RR7, along with Lemmas 36, 43, 46, 47, and 48, Corollary 42, and Observation 49, we have the following theorem.

▶ **Theorem 5.** *EDP on clique paths admits a kernel with at most $2k + 1$ vertices.*

4 Kernelization Results on VDP on Split Graphs

Let $(G, \mathcal{X}, k)$ be an instance of VDP where G is a split graph. Note that given a split graph G, we can compute (in linear time) a partition (C, I) of $V(G)$ such that C is a clique and I is an independent set [26]. We partition the set I into two sets, say, I_T and I_N, where I_T denotes the set of terminal vertices in I and I_N denotes the set of non-terminal vertices in I. Observe that a terminal pair in a split graph can only be of one of the following types: (i) Type-I: One of the terminal vertices belongs to C, and the other belongs to I, (ii) Type-II: Both terminal vertices belong to C, and Type-III: Both terminal vertices belong to I.

Our kernelization algorithm is based on a preprocessing step and the application of three reduction rules (RR8-RR10). We begin by defining the Clean-Up operation, which is a preprocessing step of our kernelization algorithm. Therefore, given an instance $(G, \mathcal{X}, k)$ of VDP where G is a split graph, we apply this operation before applying RR8-RR10.

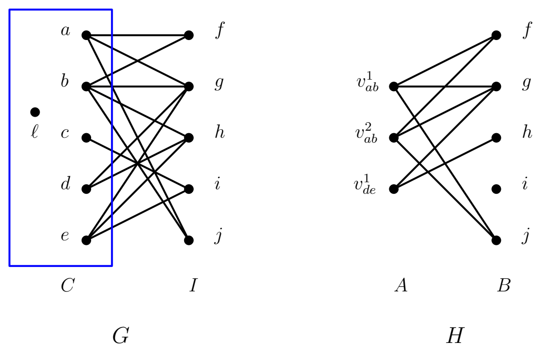

Figure 1 Let $(G, \mathcal{X}, k)$ be an instance of VDP where G is a split graph with a partition (C, I) and $\mathcal{X} = \{(a, b), (a, b), (a, b), (d, e), (d, e), (c, d), (b, d)\}$. The vertices in C form a clique; to keep the picture clean, we do not show edges in C. Observe that the terminal pairs (a, b) and (d, e) are heavy.

Clean-Up. First, consider the following construction (Construction $\mathcal{A}$), which is crucial to define the CLEAN-UP operation. This construction was also used by Heggernes et al. [27] who used it to remove a subset of I_N (safely). However, in the CLEAN-UP operation, we will remove the entire set I_N (safely). In order to do so, we need a few more technical arguments than the ones present in [27].

Construction $\mathcal{A}$. Given an instance $(G, \mathcal{X}, k)$ of VDP where G is a split graph, we construct a bipartite graph, say, H, with bipartition (A, B) as follows: For every terminal pair (s, t) of Type-II such that st is a heavy edge with weight $w \geq 2$, we introduce $w - 1$ vertices, say, $v_{st}^1, \dots, v_{st}^{w-1}$, to A. The set B consists of all the vertices from set I_N. For each $v \in I_N$, introduce an edge from v to vertices $v_{st}^1, \dots, v_{st}^{w-1}$ if and only if v is adjacent to both s and t in G. See Figure 1 for an illustration of the construction of H from $(G, \mathcal{X}, k)$.

Due to Proposition 10, we compute a maximum matching, say, M, in H in polynomial time. Let $\widehat{\mathcal{X}} \subseteq \mathcal{X}$ be the multiset of all terminal pairs whose corresponding vertices in H are saturated by M. For example, in Figure 1, if $M = \{v_{ab}^1 f, v_{ab}^2 g, v_{de}^1 h\}$ is a maximum matching in H, then $\widehat{\mathcal{X}} = \{(a, b), (a, b), (d, e)\}$. This ends Construction $\mathcal{A}$.

▶ **Definition 50** (CLEAN-UP). *Given an instance $(G, \mathcal{X}, k)$ of VDP where G is a split graph, construct the bipartite graph H and find a maximum matching M in H, as described in Construction $\mathcal{A}$. If $I_N \neq \emptyset$ or $\widehat{\mathcal{X}} \neq \emptyset$, then $G' \Leftarrow G - I_N$, $\mathcal{X}' \Leftarrow \mathcal{X} \setminus \widehat{\mathcal{X}}$, $k' \Leftarrow k - |\widehat{\mathcal{X}}|$.*

Note that due to Remark 67, G' is a split graph. Now, with the help of Definition 52, Observation 53, we will establish that $(G, \mathcal{X}, k)$ and $(G', \mathcal{X}', k')$, as described in Definition 50, are equivalent (see Lemma 54).

Before that, consider the next observation which follows trivially from Proposition 16.

▶ **Observation 51.** *Let $(G, \mathcal{X}, k)$ be a Yes-instance of VDP where G is a split graph. Moreover, let $\mathcal{P}$ be a minimum solution of $(G, \mathcal{X}, k)$. If there exists a path $P \in \mathcal{P}$ that visits a vertex $x \in I_N$, then P must be of the form (u', x, v'), where $u', v' \in C$ are terminal vertices such that $P_{u'v'} \in \mathcal{P}$.*

▶ **Definition 52.** *Let $(G, \mathcal{X}, k)$ be a Yes-instance of VDP where G is a split graph. Moreover, let M and H be as described in Construction $\mathcal{A}$. Let $\mathcal{P}$ be a minimum solution of $(G, \mathcal{X}, k)$. Then, $M' \subseteq E(H)$ is said to be* induced *by $\mathcal{P}$ in H if it is constructed as follows:*

1. *Initialize $M' \Leftarrow \emptyset$.*
2. *For every path $P \in \mathcal{P}$ that visits a vertex $x \in I_N$: By Observation 51, P must be of the form (u', x, v'), where $u', v' \in C$ are terminal vertices such that $P_{u'v'} \in \mathcal{P}$. This further implies that $u'v'$ is a heavy edge. Let $w \geq 2$ be the weight of $u'v'$, and consider the vertices $v^1_{u'v'}, \dots, v^{w-1}_{u'v'}$ in graph H. By Construction $\mathcal{A}$, x is adjacent to every vertex in the set $\{v^1_{u'v'}, \dots, v^{w-1}_{u'v'}\}$. So, we can arbitrarily choose an edge $xv^j_{u'v'}$ of H, for some $j \in [w-1]$ such that $xv^j_{u',v'}$ does not appear in M'*[1]*, and add it to M'.*

▶ **Observation 53** (∗). *Let $(G, \mathcal{X}, k)$ be a Yes-instance of VDP where G is a split graph. Moreover, let M, A, B, and H be as described in Construction $\mathcal{A}$. Let $\mathcal{P}$ be a minimum solution of $(G, \mathcal{X}, k)$. Let M' be induced by $\mathcal{P}$ in H as described in Definition 52. Then, M' is a matching.*

For an illustrative example, consider G, H, and $\mathcal{X}$ given in Figure 1. Let $\mathcal{P} = \{(a, f, b), (a, \ell, b), P_{ab}, (d, g, e), P_{de}\ , P_{cd}, P_{bd}\}$ be a solution of $(G, \mathcal{X}, 7)$. Then, $\{v^1_{ab}f, v^1_{de}g\}$ and $\{v^2_{ab}f, v^1_{de}g\}$ are two possible choices for M'.

▶ **Lemma 54** (∗). *Let $(G, \mathcal{X}, k)$ be an instance of VDP where G is a split graph. Moreover, let $\widehat{\mathcal{X}}$ be as described in Construction $\mathcal{A}$. Furthermore, let $(G', \mathcal{X}', k')$ be the output of* Clean-Up *on $(G, \mathcal{X}, k)$. Then, if $(G', \mathcal{X}', k')$ is a Yes-instance of VDP, then $(G, \mathcal{X}, k)$ is also a Yes-instance of VDP.*

Now, consider the following lemma.

▶ **Lemma 55** (∗). *Let $(G, \mathcal{X}, k)$ be an instance of VDP obtained by applying* Clean-Up*. Let $(s, t) \in \mathcal{X}$ be a heavy terminal pair of Type-II in G of weight $w \geq 2$. Then, any minimum solution of $(G, \mathcal{X}, k)$ must contain the following internally vertex-disjoint paths: $\{P_{st}\} \cup \{(s, u^1, t), \dots, (s, u^{w-1}, t)\}$, where $\{u^1, \dots, u^{w-1}\}$ is a set of non-terminals in C.*

Reduction Rules. Let us start by defining our first reduction rule (RR8).

▶ **Reduction Rule 8** (RR8). *If there is a terminal pair $(s, t) \in \mathcal{X}$ of Type-I such that $st \in E(G)$, then $V(G') \Leftarrow V(G)$, $E(G') \Leftarrow E(G) \setminus \{st\}$, $\mathcal{X}' \Leftarrow \mathcal{X} \setminus \{(s, t)\}$, $k' \Leftarrow k - 1$. Furthermore, for every $x \in \{s, t\}$ that does not appear as a terminal in any terminal pair in $\mathcal{X}'$, update $V(G') \Leftarrow V(G') \setminus \{x\}$.*

We have the following lemma to establish that RR8 is safe.

▶ **Lemma 56** (∗). *RR8 is safe.*

▶ **Observation 57.** *After applying RR8 exhaustively on G, no Type-I terminal pair in G has an edge between its terminals. Moreover, RR8 can be applied in polynomial time.*

Let $\{(s, t) \times (w)\}$ denote w copies of (s, t).

▶ **Reduction Rule 9** (RR9). *If there is a heavy terminal pair $(s, t) \in \mathcal{X}$ of Type-II of weight $w \geq 2$, then $V(G') \Leftarrow V(G) \cup \{s^1, \dots, s^{w-1}, t^1, \dots, t^{w-1}\}$, $E(G') \Leftarrow E(G) \cup \{s^i v, t^i v : v \in C, i \in [w-1]\}$, $\mathcal{X}' \Leftarrow (\mathcal{X} \setminus \overline{\mathcal{X}}) \cup \{(s^1, t^1), \dots, (s^{w-1}, t^{w-1})\}$, where $\overline{\mathcal{X}} = \{(s, t) \times (w-1)\} \subseteq \mathcal{X}$.*

We have the following lemma to establish that RR9 is safe.

[1] The existence of such a j follows from the choice of w, and since $P_{u'v'} \in \mathcal{P}$

▶ **Lemma 58** (∗). *After* Clean-Up *and the exhaustive application of RR8, RR9 is safe.*

▶ **Observation 59.** *After applying RR9 exhaustively, there do not exist any heavy Type-II terminal pairs. Moreover, RR9 can be applied in polynomial time.*

The next reduction rule is applied for every terminal participating in more than one terminal pair.

▶ **Reduction Rule 10** (RR10). *If $v \in V(G)$ belongs to $x \geq 2$ terminal pairs $(v,a_1),\dots,(v,a_x)$, then $V(G') \Leftarrow (V(G) \setminus \{v\}) \cup \{v_1,\dots,v_x\}$, $E(G') \Leftarrow E(G) \cup \{v_iu : u \in N(v), i \in [x]\}$, $\mathcal{X}' \Leftarrow (\mathcal{X} \setminus \{(v,a_1),\dots,(v,a_x)\}) \cup \{(v_1,a_1),\dots,(v_x,a_x)\}$. Moreover, if $v \in C$, then $E(G') = E(G') \cup \{v_iv_j : i \neq j \in [x]\}$.*

We have the following lemma to establish that RR10 is safe.

▶ **Lemma 60** (∗). *After* Clean-Up *and an exhaustive application of RR8 and RR9, RR10 is safe.*

By Lemma 55 and Observations 57 and 59, we have the following lemma.

▶ **Observation 61.** *After applying reduction rules RR8-RR10 exhaustively, no terminal participates in more than one terminal pair.*

Before concluding this section, we need the following result by Yang et al. [47].

▶ **Proposition 62** ([47]). *VDP-*Unique *on split graphs admits a kernel with at most $4k$ vertices, where k is the number of occurrences of terminal pairs.*

By Observations 61, we note that every instance $(G,\mathcal{X},k)$ of VDP where G is a split graph, can be converted to an instance $(G',\mathcal{X}',k')$ of VDP-Unique in polynomial time. Due to Observations 57 and 59 and Lemmas 54, 56, 58, and 60, $(G,\mathcal{X},k)$ and $(G',\mathcal{X}',k)$ are equivalent. Furthermore, our initial parameter k does not increase during the application of the Clean-Up operation and rules RR8-RR10. Therefore, using Proposition 62, we have the following theorem.

▶ **Theorem 6.** *VDP on split graphs admits a kernel with at most $4k$ vertices.*

5 Conclusion

In this paper, we studied VDP and EDP, two disjoint paths problems in the realm of Parameterized Complexity. We analyzed these problems with respect to the natural parameter "the number of (occurrences of) terminal pairs". We gave several improved kernelization results as well as new kernelization results for these problems on subclasses of chordal graphs.

For VDP, we provided a $4k$ vertex kernel on split graphs and an $\mathcal{O}(k^2)$ vertex kernel on well-partitioned chordal graphs. We also show that VDP becomes polynomial-time solvable on threshold graphs. For EDP, we first proved NP-hardness on complete graphs. Second, we provided an $\mathcal{O}(k^{2.75})$ vertex kernel on split graphs, a $7k+1$ vertex kernel on threshold graphs, an $\mathcal{O}(k^2)$ vertex kernel for EDP on block graphs, and a $2k+1$ vertex kernel on clique paths.

Apart from the obvious open questions to improve the sizes of the kernels we designed, it is interesting to determine if VDP or/and EDP admit polynomial kernels for chordal graphs. It is worth noting here that Golovach et al. [24] proved that it is unlikely for Set-Restricted Disjoint Paths, a generalization of VDP where each terminal pair has to find its path from a predefined set of vertices, to admit a polynomial kernel even on interval graphs. However, as noted by them, their reduction is heavily dependent on the sets designed for terminal pairs and thus cannot be directly generalized to VDP. Moreover, recently Włodarczyk and Zehavi [46] established that both VDP and EDP are unlikely to admit polynomial compression even when restricted to planar graphs. They also suggested to investigate the existence of polynomial kernels for chordal graphs for VDP and EDP.

Another interesting open problem is to study the kernelization complexity of EDP on well-partitioned chordal graphs. Note that EDP on well-partitioned chordal graphs is more subtle than VDP on the same. The reason is that, in VDP, a path between a non-adjacent terminal pair must be induced due to Proposition 16. This paves the way to define valid paths in the partition tree of the given well-partitioned chordal graph. However, the concept of valid paths (and, thus, the used techniques) fails for EDP, as a path in the solution can visit the bags of the partition tree in any weird manner. Furthermore, the approach for EDP on block graphs does not generalize to well-partitioned chordal graphs because the intersection of adjacent bags can be large (in well-partitioned chordal graphs), whereas, for a block graph G, there always exists a partition tree such that for any two consecutive bags, the corresponding boundary of one of the bags has size one.

References

1 I. Adler, S. G. Kolliopoulos, P. K. Krause, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Irrelevant vertices for the planar disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 122:815–843, 2017.

2 J. Ahn, L. Jaffke, O. J. Kwon, and P. T. Lima. Well-partitioned chordal graphs. *Discrete Mathematics*, 345(10):112985, 2022.

3 Claude Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences*, 43(9):842–844, 1957.

4 H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theoretical Computer Science*, 412(35):4570–4578, 2011.

5 J. Chaudhary, H. Gahlawat, M. Włodarczyk, and M. Zehavi. Kernels for the disjoint paths problem on subclasses of chordal graphs. *arXiv:2309.16892*, 2023.

6 C. Chekuri, S. Khanna, and F. B. Shepherd. An $\mathcal{O}(\sqrt{n})$ approximation and integrality gap for disjoint paths and unsplittable flow. *Theory of Computing*, 2(7):137–146, 2006.

7 J. Chuzhoy and D. H. K. Kim. On approximating node-disjoint paths in grids. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015*, pages 187–211. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

8 J. Chuzhoy, D. H. K. Kim, and S. Li. Improved approximation for node-disjoint paths in planar graphs. In *Proceedings of the 48th annual ACM symposium on Theory of Computing, STOC 2016*, pages 556–569, 2016.

9 J. Chuzhoy, D. H. K. Kim, and R. Nimavat. Improved approximation for node-disjoint paths in grids with sources on the boundary. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, pages 38:1–38:14. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

10 J. Chuzhoy, D. H. K. Kim, and R. Nimavat. Almost polynomial hardness of node-disjoint paths in grids. *Theory of Computing*, 17(6):1–57, 2021. `doi:10.4086/toc.2021.v017a006`.

11 J. Chuzhoy, D. H. K. Kim, and R. Nimavat. New hardness results for routing on disjoint paths. *SIAM Journal on Computing*, 51(2):189–237, 2022.

12 M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015.

13 R. Diestel. Graph theory 3rd ed. *Graduate texts in mathematics*, 173(33):12, 2005.

14 Z. Dvořák, D. Král', and R. Thomas. Coloring triangle-free graphs on surfaces. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 120–129. SIAM, 2009.

15 A. Ene, M. Mnich, M. Pilipczuk, and A. Risteski. On routing disjoint paths in bounded treewidth graphs. In *Proceedings of the 15th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2016*, pages 1–15. Schloss Dagstuhl, 2016.

16 S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5(4):691–703, 1976.

17 Michael R Fellows. The lost continent of polynomial time: Preprocessing and kernelization. In *International Workshop on Parameterized and Exact Computation*, pages 276–277. Springer, 2006.

18 K. Fleszar, M. Mnich, and J. Spoerhase. New algorithms for maximum disjoint paths based on tree-likeness. In *Proceedings of the European Symposium on Algorithms, ESA 2016*, pages 1–17. Schloss Dagstuhl, 2016.

19 F. V. Fomin, D. Lokshtanov, S. Saurabh, and M. Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. doi:10.1017/9781107415157.

20 A. Frank. Packing paths, circuits, and cuts-a survey. *Paths, flows, and VLSI-layout*, 49:100, 1990.

21 R. Ganian and S. Ordyniak. The power of cut-based parameters for computing edge-disjoint paths. *Algorithmica*, 83:726–752, 2021.

22 N. Garg, V. V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.

23 P. A. Golovach, F. Panolan, A. Rai, and S. Saurabh. New algorithms for maximum disjoint paths based on tree-likeness. In *Proceedings of the European Symposium on Algorithms, ESA 2016*, pages 1–17. Schloss Dagstuhl, 2016.

24 P. A. Golovach, F. Panolan, A. Rai, and S. Saurabh. Parameterized complexity of set-restricted disjoint paths on chordal graphs. In *Proceedings of the 17th International Computer Science Symposium in Russia, CSR 2022, Virtual Event, June 29–July 1, 2022, Proceedings*, pages 152–169. Springer, 2022.

25 F. Gurski and E. Wanke. Vertex disjoint paths on clique-width bounded graphs. *Theoretical Computer Science*, 359(1–3):188–199, 2006.

26 P. L. Hammer and B. Simeone. The splittance of a graph. *Combinatorica*, 1:275–284, 1981.

27 P. Heggernes, P. V. Hof, E. J. V. Leeuwen, and R. Saei. Finding disjoint paths in split graphs. *Theory of Computing Systems*, 57(1):140–159, 2015.

28 J. E. Hopcroft and R. M. Karp. An $n^{\frac{5}{2}}$ algorithm for maximum matching in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

29 F. Kammer, T. Tholey, O. J. Kwon, and P. T. Lima. The k-disjoint paths problem on chordal graphs. In *Proceedings of the 35th International Workshop on Graph-Theoretic Concepts in Computer Science*, WG 2009, pages 190–201. Springer, Berlin, 2009. doi:10.1007/978-3-642-11409-0_17.

30 R. M. Karp. On the computational complexity of combinatorial problems. *Networks*, 5:45–68, 1975.

31 K. Kawarabayashi, Y. Kobayashi, and S. Kreutzer. An excluded half-integral grid theorem for digraphs and the directed disjoint paths problem. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC 2014, pages 70–78, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2591796.2591876.

32 K. Kawarabayashi, Y. Kobayashi, and B. Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, 2012.

33 Y. Kobayashi and K. Kawarabayashi. Algorithms for finding an induced cycle in planar graphs and bounded genus graphs. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1146–1155. SIAM, 2009.

34 S. G. Kolliopoulos and C. Stein. Approximating disjoint-path problems using packing integer programs. *Mathematical Programming*, 99(1):63–87, 2004.

35 M. Kramer and J. V. Leeuwen. The complexity of wirerouting and finding minimum area layouts for arbitrary vlsi circuits. *Advances in Computing Research*, 2:129–146, 1984.

36 D. Lokshtanov, P. Misra, M. Pilipczuk, S. Saurabh, and M. Zehavi. An exponential time parameterized algorithm for planar disjoint paths. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 1307–1316, New York, NY, USA, 2020. Association for Computing Machinery. `doi:10.1145/3357713.3384250`.

37 D. Lokshtanov, S. Saurabh, and M. Zehavi. Efficient graph minors theory and parameterized algorithms for (planar) disjoint paths. In *Treewidth, Kernels, and Algorithms*, pages 112–128. Springer, 2020.

38 S. Natarajan and A. P. Sprague. Disjoint paths in circular arc graphs. *Nordic Journal of Computing*, 3:256–270, 1996.

39 T. Nishizeki, J. Vygen, and X. Zhou. The edge-disjoint paths problem is NP-complete for series-parallel graphs. *Discrete Applied Mathematics*, 115(1–3):177–186, 2001.

40 B. Reed. Rooted routing in the plane. *Discrete Applied Mathematics*, 57(2-3):213–227, 1995.

41 Bruce A Reed. Tree width and tangles: A new connectivity measure and some applications. *Surveys in combinatorics*, pages 87–162, 1997.

42 N. Robertson and P. D. Seymour. Graph minors XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.

43 N. Robertson and P. D. Seymour. Graph minors XXII. Irrelevant vertices in linkage problems. *Journal of Combinatorial Theory, Series B*, 102(2):530–563, 2012.

44 A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

45 A. Srinivas and E. Modiano. Finding minimum energy disjoint paths in wireless ad-hoc networks. *Wireless Networks*, 11:401–417, 2005.

46 M. Włodarczyk and M. Zehavi. Planar disjoint paths, treewidth, and kernels. In *Proceedings of the 64th IEEE Symposium on Foundations of Computer Science (FOCS) 2023*, 2023.

47 Y. Yang, Y. R. Shrestha, W. Li, and J. Guo. On the kernelization of split graph problems. *Theoretical Computer Science*, 734:72–82, 2018.

48 X. Zhou, S. Tamura, and T. Nishizeki. Finding edge-disjoint paths in partial k-trees. *Algorithmica*, 26(1):3–30, 2000.

A Brief Survey of Related Works

Both VDP and EDP are known to be NP-complete for planar graphs [35], line graphs [27], and split graphs [27]. Moreover, VDP is known to be NP-complete on interval graphs [38] and grids [35] as well. Both VDP and EDP were studied from the viewpoint of structural parameterization. While VDP is FPT parameterized by treewidth [41], EDP remains NP-complete even on graphs with treewidth at most 2 [39]. Gurski and Wange [25] showed that VDP is solvable in linear time on co-graphs but becomes NP-complete for graphs with clique-width at most 6. As noted by Heggernes et al. [27], there is a reduction from VDP on general graphs to EDP on line graphs, and from EDP on general graphs to VDP on line graphs. Since a graph with treewidth ℓ has clique-width at most $2\ell + 2$ [25], EDP can also be shown to be NP-complete on graphs with clique-width at most 6 [27].

The Graph Minors theory of Robertson and Seymour provides some of the most important algorithmic results of the modern graph theory. Unfortunately, these algorithms, along with the $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$ algorithms for VDP and EDP(when k is fixed), respectively [42, 32], hide such big constants that they have earned a name for themselves: "*galactic algorithms*". Since then, finding efficient FPT algorithms for VDP and EDP has been a tantalizing question for researchers. Several improvements have been made for the class of planar graphs [1, 36, 40, 43], chordal graphs [29], and bounded-genus graphs [40, 14, 33].

Concerning kernelization (in addition to the works surveyed in the introduction), we note that Ganian and Ordyniak [21] proved that EDP admits a linear kernel parameterized by the feedback edge set number. Recently, Golovach et al. [23] proved that Set-Restricted Disjoint Paths, a variant of VDP where each terminal pair has to find its path from a predefined set of vertices, does not admit a polynomial compression on interval graphs unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

The optimization variants of VDP and EDP– MaxVDP and MaxEDP – are well-studied in the realm of approximation algorithms [6, 15, 18, 22, 34]. Chekuri et al. [6] gave an $\mathcal{O}(\sqrt{n})$-approximation algorithm for MaxEDP on general graphs, matching the $\Omega(\sqrt{n})$ lower bound provided by Garg et al. [22]. Recently, MaxVDP has gained much attention on planar graphs [7, 8, 10, 11, 9]. Highlights of this line of works (MaxVDP on planar graphs) include an approximation algorithm with approximation ratio $\mathcal{O}(n^{\frac{9}{19}} \log^{\mathcal{O}(1)} n)$ [8], and hardness of approximating within a factor of $n^{\Omega(1/(\log\log n)^2)}$ [10].

B Preliminaries

B.1 Parameterized Complexity

Standard notions in Parameterized Complexity not explicitly defined here can be found in [12]. Let Π be an NP-hard problem. In the framework of Parameterized Complexity, each instance of Π is associated with a *parameter* k. We say that Π is *fixed-parameter tractable* (FPT) if any instance (I, k) of Π is solvable in time $f(k) \cdot |I|^{\mathcal{O}(1)}$, where f is some computable function of k. Two instances (I, k) and (I', k') (possibly of different problems) are *equivalent* if: (I, k) is a Yes-instance if and only if (I', k') is a Yes-instance.

A parameterized (decision) problem Π admits a *kernel* of size $f(k)$ for some computable function f that depends only on k if the following is true: There exists an algorithm (called a *kernelization algorithm*) that runs in $(|I| + k)^{\mathcal{O}(1)}$ time and translates any input instance (I, k) of Π into an equivalent instance (I', k') of Π such that the size of (I', k') is bounded by $f(k)$. If the function f is polynomial (resp., linear) in k, then the problem is said to admit a *polynomial kernel* (resp., *linear kernel*). It is well known that a decidable parameterized problem is FPT if and only if it admits a kernel [12].

To design kernelization algorithms, we rely on the notion of *reduction rule*, defined below.

▶ **Definition 63** (Reduction Rule). *A* reduction rule *is a polynomial-time procedure that consists of a condition and an operation, and its input is an instance (I, k) of a parameterized problem Π. If the condition is true, then the rule outputs a new instance (I', k') of Π such that $k' \leq k$.*

A reduction rule is *safe*, when the condition is true, (I, k) and (I', k') are equivalent. Throughout this paper, the reduction rules will be numbered, and the reduction rules will be applied exhaustively in the increasing order of their indices. So, if reduction rules i and j, where $i < j$, are defined for a problem, then i will be applied exhaustively before j. Notice that after the application of rule j, the condition of rule i might become true. In this situation, we will apply rule i again (exhaustively). In other words, when we apply rule j, we always assume that the condition of rule i is false.

B.2 Graph Classes

A graph G is a *chordal graph* if every cycle in G of length at least four has a *chord*, that is, an edge joining two non-consecutive vertices of the cycle. In what follows, we define several subclasses of the class of chordal graphs, namely, *complete graphs*, *block graphs*, *split*

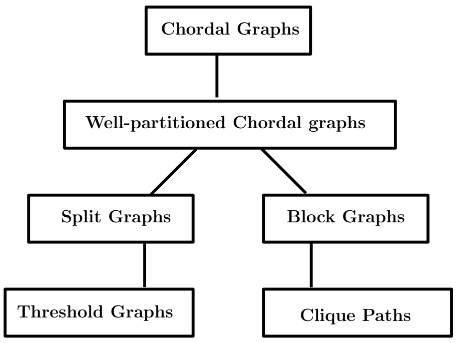

Figure 2 The inclusive relationship among the subclasses of chordal graphs studied in this paper.

graphs, *threshold graphs*, and *well-partitioned chordal graphs*. A graph whose vertex set is a clique is a *complete graph*. A vertex is a *cut vertex* in a graph G if removing it increases the total number of connected components in G. A *block* of a graph G is a maximal connected subgraph of G that does not contain any cut vertex. A graph G is a *block graph* if the vertex set of every block in G is a clique. A block of a block graph G is an *end block* if it contains exactly one cut vertex. Note that a block graph that is not a complete graph has at least two end blocks. A graph G is a *split graph* if there is a partition (C, I) of $V(G)$ such that C is a clique and I is an independent set. A split graph G is a *threshold graph* if there exists a linear ordering of the vertices in I, say, $(v_1, v_2, \ldots, v_{|I|})$, such that $N(v_1) \subseteq N(v_2) \subseteq \cdots \subseteq N(v_{|I|})$.

An undirected graph in which any two vertices are connected by exactly one path is a *tree*. A tree with at most two vertices or exactly one non-pendant vertex is a *star*. The vertices of degree one in a tree are called *leaves*. Note that a split graph admits a partition of its vertex set into cliques that can be arranged in a star structure, where the leaves are cliques of size one. Motivated by this definition of split graphs, Ahn et al. [2] introduced *well-partitioned chordal graphs*, which are defined by relaxing the definition of split graphs in the following two ways: (i) by allowing the parts of the partition to be arranged in a tree structure instead of a star structure, and (ii) by allowing the cliques in each part to have arbitrary size instead of one. A more formal definition of a well-partitioned chordal graph is given below.

▶ **Definition 64** (Well-Partitioned Chordal Graph). *A connected graph G is a* well-partitioned chordal graph *if there exists a partition $\mathcal{B}$ of $V(G)$ and a tree $\mathcal{T}$ having $\mathcal{B}$ as its vertex set such that the following hold.*

(i) *Each part $X \in \mathcal{B}$ 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 $E(G[X, Y]) = X' \times Y'$.*
(iii) *For each pair of distinct $X, Y \in \mathcal{B}$ with $XY \notin E(\mathcal{T})$, $E(G[X, Y]) = \emptyset$.*

The tree $\mathcal{T}$ in Definition 64 is called a *partition tree* of G, and the elements of $\mathcal{B}$ are called its *bags*. Notice that a well-partitioned chordal graph can have multiple partition trees.

▶ **Proposition 65** ([2]). *Given a well-partitioned chordal graph G, a partition tree of G can be found in polynomial time.*

▶ **Definition 66** (Boundary of a Well-Partitioned Chordal Graph). *Let $\mathcal{T}$ be a partition tree of a well-partitioned chordal graph G and let $XY \in E(\mathcal{T})$. The* boundary *of X with respect to Y, denoted by $\mathsf{bd}(X, Y)$, is the set of vertices of X that have a neighbor in Y, i.e., $\mathsf{bd}(X, Y) = \{x \in X : N_G(x) \cap Y \neq \emptyset\}$.*

▶ Remark 67. Note that all the graph classes considered in this paper are closed under vertex deletion. Therefore, throughout this paper, if G belongs to class $\mathcal{Z}$ and G' is obtained from G after deleting some vertices, then we assume that G' also belongs to $\mathcal{Z}$ without mentioning it explicitly.

The inclusion relationship among various subclasses of chordal graphs discussed in this paper is shown in Figure 2.

C NP-hardness for Complete Graphs

In this section, we prove that EDP is NP-hard on complete graphs by giving a polynomial-time reduction from EDP on general graphs, which is known to be NP-hard [35]. Our reduction is based on the standard technique of adding missing edges and placing a terminal pair on the endpoints of the added edge. This technique was also used to prove the NP-hardness of EDP for split graphs [27].

▶ **Theorem 1.** *EDP is* NP*-hard on complete graphs.*

Proof. Let $(G, \mathcal{X}, k)$ be an instance of EDP, where $\mathcal{X} = \{(s_1, t_1), \dots, (s_k, t_k)\}$. Define the graph G' as follows: Let $V(G') = V(G)$ and $E(G') = E(G) \cup \{uv : uv \notin E(G)\}$. Furthermore, let $\mathcal{T} = \{(s_{uv}, t_{uv}) : uv \in E(G') \setminus E(G), s_{uv} = u, t_{uv} = v\}$. Note that for ease of notation, we denote the terminal pairs in $\mathcal{T}$ by (s_{uv}, t_{uv}) rather than (u, v). Also, for an edge $uv \in E(G') \setminus E(G)$, we introduce either (s_{uv}, t_{uv}) or (s_{vu}, t_{vu}) in $\mathcal{T}$, not both. Let $\mathcal{X}' = \mathcal{X} \cup \mathcal{T}$. Now, we claim that $(G, \mathcal{X}, k)$ and $(G', \mathcal{X}', k')$ are equivalent instances of EDP, where $k' = k + |\mathcal{T}|$. Let $\mathcal{P}_{\mathcal{T}} = \{P_{uv} : (s_{uv}, t_{uv}) \in \mathcal{T}\}$.

In one direction, let $\mathcal{P} = \{P_1, \dots, P_k\}$ be a solution of $(G, \mathcal{X}, k)$. Since for every $(s_{uv}, t_{uv}) \in \mathcal{T}$, the edge uv does not belong to any path in $\mathcal{P}$, $\mathcal{P} \cup \mathcal{P}_{\mathcal{T}}$ is a set of edge-disjoint paths in G'. As $\mathcal{X}' = \mathcal{X} \cup \mathcal{T}$, $\mathcal{P} \cup \mathcal{P}_{\mathcal{T}}$ is a solution of $(G', \mathcal{X}', k')$.

In the other direction, let $\mathcal{P}'$ be a solution of $(G', \mathcal{X}', k')$ that contains as many paths from $\mathcal{P}_{\mathcal{T}}$ as possible. Next, we claim that $\mathcal{P}_{\mathcal{T}} \subseteq \mathcal{P}'$. Targeting a contradiction, suppose that there exists a terminal pair, say, $(s_{uv}, t_{uv}) \in \mathcal{T}$, such that $P_{uv} \notin \mathcal{P}'$. Let Q denote the path in $\mathcal{P}'$ connecting s_{uv} and t_{uv}. If none of the paths in $\mathcal{P}'$ uses the edge uv, then the set $(\mathcal{P}' \setminus Q) \cup \{P_{uv}\}$ is a solution of $(G', \mathcal{X}', k')$ containing more paths from $\mathcal{P}_{\mathcal{T}}$ than $\mathcal{P}'$, contradicting the choice of $\mathcal{P}'$. Hence, there must be a unique path $P^* \in \mathcal{P}'$ that uses the edge uv. Let s^* and t^* be the two terminals that are connected by the path P^*. Let W denote the walk between s^* and t^* obtained from P^* by replacing the edge uv with the path Q (there may be some vertices that are repeated in W). Since $E(W) = (E(P^*) \cup E(Q)) \setminus \{uv\}$, W is edge-disjoint from every path in $\mathcal{P}' \setminus \{P^*, Q\}$ (as $\mathcal{P}'$ is a set of edge-disjoint paths). Let Q^* be a path between s^* and t^* that uses a subset of edges of W, which again is edge-disjoint from every path in $\mathcal{P}' \setminus \{P^*, Q\}$. Hence, $\mathcal{P} = (\mathcal{P}' \setminus \{P^*, Q\}) \cup \{P_{uv}, Q^*\}$ is a solution of $(G', \mathcal{X}', k')$ that contains one more path from $\mathcal{P}_{\mathcal{T}}$ than $\mathcal{P}'$. This contradicts the choice of $\mathcal{P}'$, and thus implies that $\mathcal{P}_{\mathcal{T}} \subseteq \mathcal{P}'$. Since $\mathcal{X}' = \mathcal{X} \cup \mathcal{T}$ and $\mathcal{P}_{\mathcal{T}}$ contains the edge-disjoint paths between the terminal pairs present in $\mathcal{T}$, $\mathcal{P}' \setminus \mathcal{P}_{\mathcal{T}}$ must contain the edge-disjoint paths between the terminal pairs present in $\mathcal{X}$. Thus, $\mathcal{P}' \setminus \mathcal{P}_{\mathcal{T}}$ is a solution of $(G, \mathcal{X}, k)$. ◀