# Finding Disjoint Paths in Split Graphs*

Pinar Heggernes[1], Pim van 't Hof[1], Erik Jan van Leeuwen[2], and Reza Saei[1]

[1] Department of Informatics, University of Bergen, Norway
{pinar.heggernes,pim.vanthof,reza.saeidinvar}@ii.uib.no
[2] MPI für Informatik, Saarbrücken, Germany
erikjan@mpi-inf.mpg.de

**Abstract.** The well-known Disjoint Paths problem takes as input a graph $G$ and a set of $k$ pairs of terminals in $G$, and the task is to decide whether there exists a collection of $k$ pairwise vertex-disjoint paths in $G$ such that the vertices in each terminal pair are connected to each other by one of the paths. This problem is known to NP-complete, even when restricted to planar graphs or interval graphs. Moreover, although the problem is fixed-parameter tractable when parameterized by $k$ due to a celebrated result by Robertson and Seymour, it is known not to admit a polynomial kernel unless NP $\subseteq$ coNP/poly. We prove that Disjoint Paths remains NP-complete on split graphs, and show that the problem admits a kernel with $O(k^2)$ vertices when restricted to this graph class. We furthermore prove that, on split graphs, the edge-disjoint variant of the problem is also NP-complete and admits a kernel with $O(k^3)$ vertices. To the best of our knowledge, our kernelization results are the first non-trivial kernelization results for these problems on graph classes.

## 1 Introduction

Finding vertex-disjoint or edge-disjoint paths with specified endpoints is one of the most studied classical and fundamental problems in algorithmic graph theory and combinatorial optimization, with many applications in such areas as VLSI layout, transportation networks, and network reliability; see, for example, the surveys by Frank [9] and by Vygen [24]. An instance of the Vertex-Disjoint Paths problem consists of a graph $G$ with $n$ vertices and $m$ edges, and a set $\mathcal{X} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of $k$ pairs of vertices in $G$, called the *terminals*. The question is whether there exists a collection $\mathcal{P} = \{P_1, \ldots, P_k\}$ of $k$ pairwise vertex-disjoint paths in $G$ such that $P_i$ connects $s_i$ to $t_i$ for every $i \in \{1, \ldots, k\}$. The Edge-Disjoint Paths problem is defined analagously, but here the task is to decide whether there exist $k$ pairwise edge-disjoint paths instead of vertex-disjoint paths.

The Vertex-Disjoint Paths problem was shown to be NP-complete by Karp [13], one year before Even et al. [8] proved that the same holds for Edge-Disjoint Paths. A celebrated result by Robertson and Seymour [22], obtained

* This research is supported by the Research Council of Norway.

V. Geffert et al. (Eds.): SOFSEM 2014, LNCS 8327, pp. 315–326, 2014.

as part of their groundbreaking graph minors theory, states that the Vertex-Disjoint Paths problem can be solved in $O(n^3)$ time for every fixed $k$. This implies that Edge-Disjoint Paths can be solved in $O(m^3)$ time for every fixed $k$. As a recent development, an $O(n^2)$-time algorithm for each of the problems, for every fixed $k$, was obtained by Kawarabayashi, Kobayashi and Reed [14]. The above results show that both problems are fixed-parameter tractable when parameterized by the number of terminal pairs. On the negative side, Bodlaender, Thomassé and Yeo [3] showed that, under the same parameterization, the Vertex-Disjoint Paths problem does not admit a polynomial kernel, i.e., an equivalent instance whose size is bounded by a polynomial in $k$, unless NP $\subseteq$ coNP/poly.

Due to their evident importance, both problems have been intensively studied on graph classes. A trivial reduction from Edge-Disjoint Paths to Vertex-Disjoint Paths implies that the latter problem is NP-complete on line graphs. It is known that both problems remain NP-complete when restricted to planar graphs [16,17]. On the positive side, Vertex-Disjoint Paths can be solved in linear time for every fixed $k$ on planar graphs [21], or more generally, on graphs of bounded genus [7,15]. Interestingly, Vertex-Disjoint Paths is polynomial-time solvable on graphs of bounded treewidth [20], while Edge-Disjoint Paths is NP-complete on series-parallel graphs [19], and hence on graphs of treewidth at most 2. Gurski and Wanke [11] proved that Vertex-Disjoint Paths is NP-complete on graphs of clique-width at most 6, but can be solved in linear time on graphs of clique-width at most 2. Natarayan and Sprague [18] proved the NP-completeness of Vertex-Disjoint Paths on interval graphs, and hence also on all superclasses of interval graphs such as circular-arc graphs and chordal graphs. On chordal graphs, Vertex-Disjoint Paths is linear-time solvable for each fixed $k$ [12].

Given the fact that the Vertex-Disjoint Paths problem is unlikely to admit a polynomial kernel on general graphs, and the amount of known results for both problems on graph classes, it is surprising that no kernelization result has been known on either problem when restricted to graph classes. Interestingly, even the classical complexity status of both problems has been open on split graphs, i.e., graphs whose vertex set can be partitioned into a clique and an independent set, which form a well-studied graph class and a famous subclass of chordal graphs [4,10].

We present the first non-trivial kernelization results for Vertex-Disjoint Paths and Edge Disjoint Paths on graph classes, by showing that the problems admit kernels with $O(k^2)$ and $O(k^3)$ vertices, respectively, on split graphs. To complement these results, we prove that both problems remain NP-complete on this graph class. In this extended abstract, the proofs of results marked with a star are omitted due to page restrictions.

## 2 Preliminaries

All the graphs considered in this paper are finite, simple and undirected. We refer to the monograph by Diestel [5] for graph terminology and notation not

defined below. Let $G$ be a graph. For any vertex $v$ in $G$, we write $N_G(v)$ to denote the neighborhood of $v$, and $d_G(v) = |N_G(v)|$ to denote the degree of $v$. A *split graph* is a graph whose vertex set can be partitioned into a clique $C$ and an independent set $I$, either of which may be empty; such a partition $(C, I)$ is called a *split partition*. Note that, in general, a split graph can have more than one split partition.

In any instance of Vertex-Disjoint Paths or Edge-Disjoint Paths, we allow different terminals to coincide. For this reason, by slight abuse of terminology, we define two paths to be *vertex-disjoint* if neither path contains an inner vertex of the other. This implies that in any solution $\mathcal{P}$ for an instance of Vertex-Disjoint Paths, a terminal might be an endpoint of several paths in $\mathcal{P}$, but none of the paths in $\mathcal{P}$ contains a terminal as an inner vertex. Note that edge-disjoint paths are allowed to share vertices by definition. Hence, in any solution $\mathcal{P}$ for an instance of Edge-Disjoint Paths, terminals may appear as inner vertices of paths in $\mathcal{P}$.

Let $(G, \mathcal{X})$ be an instance of the Vertex-Disjoint Paths problem, where $\mathcal{X} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$. A solution $\mathcal{P} = \{P_1, \ldots, P_k\}$ for the instance $(G, \mathcal{X})$ is *minimum* if there is no solution $\mathcal{Q} = \{Q_1, \ldots, Q_k\}$ for $(G, \mathcal{X})$ such that $\sum_{i=1}^{k} |E(Q_i)| < \sum_{i=1}^{k} |E(P_i)|$. Note that every path in a minimum solution for $(G, \mathcal{X})$ is an induced path in $G$.

For any problem $\Pi$, two instances $I_1, I_2$ of $\Pi$ are *equivalent* if $I_1$ is a yes-instance of $\Pi$ if and only if $I_2$ is a yes-instance of $\Pi$.

A *parameterized problem* is a subset $Q \subseteq \Sigma^* \times \mathbb{N}$ for some finite alphabet $\Sigma$, where the second part of the input is called the *parameter*. A parameterized problem $Q \subseteq \Sigma^* \times \mathbb{N}$ is said to be *fixed-parameter tractable* if for each pair $(x, k) \in \Sigma^* \times \mathbb{N}$ it can be decided in time $f(k)\,|x|^{O(1)}$ whether $(x, k) \in Q$, for some function $f$ that only depends on $k$; here, $|x|$ denotes the length of input $x$. We say that a parameterized problem $Q$ has a *kernel* if there is an algorithm that transforms each instance $(x, k)$ in time $(|x|+k)^{O(1)}$ into an instance $(x', k')$, such that $(x, k) \in Q$ if and only if $(x', k') \in Q$ and $|x'| + k' \leq g(k)$ for some function $g$. Here, $g$ is typically an exponential function of $k$. If $g$ is a polynomial or a linear function of $k$, then we say that the problem has a *polynomial* kernel or a *linear* kernel, respectively. We refer the interested reader to the monograph by Downey and Fellows [6] for more background on parameterized complexity. It is known that a parameterized problem is fixed-parameter tractable if and only if it is decidable and has a kernel, and several fixed-parameter tractable problems are known to have polynomial or even linear kernels. Recently, methods have been developed for proving non-existence of polynomial kernels, under some complexity theoretical assumptions [1,2,3].

In the NP-completeness proofs in Section 3, we will reduce from a restricted variant of the Satisfiability (SAT) problem. In order to define this variant, we need to introduce some terminology. Let $x$ be a variable and $c$ a clause of a Boolean formula $\varphi$ in conjunctive normal form (CNF). We say that $x$ *appears* in $c$ if either $x$ or $\neg x$ is a literal of $c$. If $x$ is a literal of clause $c$, then we say that $x$ *appears positively* in $c$. Similarly, if $\neg x$ is a literal of $c$, then $x$ *appears negatively*

in $c$. Given a Boolean formula $\varphi$, we say that a variable $x$ appears positively (respectively negatively) if there is a clause $c$ in $\varphi$ in which $x$ appears positively (respectively negatively). The following result, which we will use to prove that Vertex-Disjoint Paths is NP-complete on split graphs, is due to Tovey [23].

**Theorem 1 ([23]).** *The* SAT *problem is NP-complete when restricted to CNF formulas satisfying the following three conditions:*

- *every clause contains two or three literals;*
- *every variable appears in two or three clauses;*
- *every variable appears at least once positively and at least once negatively.*

## 3 Finding Disjoint Paths in Split Graphs Is NP-Hard

Lynch [17] gave a polynomial-time reduction from SAT to Vertex-Disjoint Paths, thereby proving the latter problem to be NP-complete in general. By modifying his reduction, he then strengthened his result and proved that Vertex-Disjoint Paths remains NP-complete when restricted to planar graphs. In this section, we first show that the reduction of Lynch can also be modified to prove that Vertex-Disjoint Paths is NP-complete on split graphs. We then prove that the Edge-Disjoint Paths problem is NP-complete on split graphs as well, using a reduction from the Edge-Disjoint Paths problem on general graphs.

We first describe the reduction from SAT to Vertex-Disjoint Paths due to Lynch [17]. Let $\varphi = c_1 \vee c_2 \vee \ldots \vee c_m$ be a CNF formula, and let $v_1, \ldots, v_n$ be the variables that appear in $\varphi$. We assume that every variable appears at least once positively and at least once negatively; if this is not the case, then we can trivially reduce the instance to an equivalent instance that satisfies this property. Given the formula $\varphi$, we create an instance $(G_\varphi, \mathcal{X}_\varphi)$ of Vertex-Disjoint Paths as follows.

The vertex set of the graph $G_\varphi$ consists of three types of vertices: variable vertices, clause vertices, and literal vertices. For each variable $v_i$ in $\varphi$, we create two *variable vertices* $v_i$ and $w_i$; we call $(v_i, w_i)$ a *variable pair*. For each clause $c_j$, we add two *clause vertices* $c_j$ and $d_j$ and call $(c_j, d_j)$ a *clause pair*. For each clause $c_j$, we also add a *literal vertex* for each literal that appears in $c_j$ as follows. If $c_j$ contains a literal $v_i$, that is, if variable $v_i$ appears positively in clause $c_j$, then we add a vertex $\ell^+_{i,j}$ to the graph, and we make this vertex adjacent to vertices $c_j$ and $d_j$. Similarly, if $c_j$ contains a literal $\neg v_i$, then we add a vertex $\ell^-_{i,j}$ and make it adjacent to both $c_j$ and $d_j$. This way, we create $|c_j|$ paths of length exactly 2 between each clause pair $(c_j, d_j)$, where $|c_j|$ is the number of literals in clause $c_j$.

For each $i \in \{1, \ldots, n\}$, we add edges to the graph in order to create exactly two vertex-disjoint paths between the variable pair $(v_i, w_i)$ as follows. Let $c_{j_1}, c_{j_2}, \ldots, c_{j_p}$ be the clauses in which $v_i$ appears positively, where $j_1 < j_2 < \cdots < j_p$. Similarly, let $c_{k_1}, c_{k_2}, \ldots, c_{k_q}$ be the clauses in which $v_i$ appears negatively, where $k_1 < k_2 < \cdots < k_p$. Note that $p \geq 1$ and $q \geq 1$ due to the

assumption that every variable appears at least once positively and at least once negatively. We now add the edges $v_i\ell^+_{i,j_1}$ and $\ell^+_{i,j_p}w_i$, as well as the edges $\ell^+_{i,j_1}\ell^+_{i,j_2}$, $\ell^+_{i,j_2}\ell^+_{i,j_3}, \ldots,$ $\ell^+_{i,j_{p-1}}\ell^+_{i,j_p}$. Let $L^+_i = v_i\ell^+_{i,j_1}\ell^+_{i,j_2}\cdots\ell^+_{i,j_{p-1}}\ell^+_{i,j_p}w_i$ denote the path between $v_i$ and $w_i$ that is created this way. Similarly, we add exactly those edges needed to create the path $L^-_i = v_i\ell^-_{i,k_1}\ell^-_{i,k_2}\cdots\ell^-_{i,j_{q-1}}\ell^-_{i,j_q}w_i$. This completes the construction of the graph $G_\varphi$.

Let $\mathcal{X}_\varphi$ be the set consisting of all the variable pairs and all the clause pairs in $G_\varphi$, i.e., $\mathcal{X}_\varphi = \{(v_i, w_i) \mid 1 \leq i \leq n\} \cup \{(c_j, d_j) \mid 1 \leq j \leq m\}$. The pair $(G_\varphi, \mathcal{X}_\varphi)$ is the instance of Vertex-Disjoint Paths corresponding to the instance $\varphi$ of SAT.

**Theorem 2 ([17]).** *Let $\varphi$ be a CNF formula. Then $\varphi$ is satisfiable if and only if $(G_\varphi, \mathcal{X}_\varphi)$ is a yes-instance of the* Vertex-Disjoint Paths *problem.*

We are now ready to prove our first result.

**Theorem 3.** *The* Vertex-Disjoint Paths *problem is NP-complete on split graphs.*

*Proof.* We reduce from the NP-complete variant of SAT defined in Theorem 1. Let $\varphi = c_1 \vee c_2 \vee \ldots \vee c_m$ be a CNF formula that satisfies the three conditions mentioned in Theorem 1, and let $v_1, \ldots, v_n$ be the variables that appear in $\varphi$. Let $(G_\varphi, \mathcal{X}_\varphi)$ be the instance of Vertex-Disjoint Paths constructed from $\varphi$ in the way described at the beginning of this section. Now let $G$ be the graph obtained from $G_\varphi$ by adding an edge between each pair of distinct literal vertices, i.e., by adding all the edges needed to make the literal vertices form a clique. The graph $G$ clearly is a split graph.

We will show that $(G, \mathcal{X}_\varphi)$ is a yes-instance of Vertex-Disjoint Paths if and only if $(G_\varphi, \mathcal{X}_\varphi)$ is a yes-instance of Vertex-Disjoint Paths. Since $(G_\varphi, \mathcal{X}_\varphi)$ is a yes-instance of Vertex-Disjoint Paths if and only if the formula $\varphi$ is satisfiable due to Theorem 2, this suffices to prove the theorem.

If $(G_\varphi, \mathcal{X}_\varphi)$ is a yes-instance of Vertex-Disjoint Paths, then so is $(G, \mathcal{X}_\varphi)$ due to the fact that $G$ is a supergraph of $G_\varphi$. Hence it remains to prove the reverse direction. Suppose $(G, \mathcal{X}_\varphi)$ is a yes-instance of Vertex-Disjoint Paths. Let $\mathcal{P} = \{P_1, \ldots, P_n, Q_1, \ldots, Q_m\}$ be a minimum solution, where each path $P_i$ connects the two terminal vertices in the variable pair $(v_i, w_i)$, and each path $Q_j$ connects the terminals in the clause pair $(c_j, d_j)$. We will show that all the paths in $\mathcal{P}$ exist also in the graph $G_\varphi$, implying that $\mathcal{P}$ is a solution for the instance $(G_\varphi, \mathcal{X}_\varphi)$.

The assumption that $\mathcal{P}$ is a minimum solution implies that every path in $\mathcal{P}$ is an induced path in $G$. By the construction of $G$, this implies that all the inner vertices of every path in $\mathcal{P}$ are literal vertices. Moreover, since the literal vertices form a clique in $G$, every path in $\mathcal{P}$ has at most two inner vertices.

Let $j \in \{1, \ldots, m\}$. Since $N_G(c_j) = N_G(d_j)$, the vertices $c_j$ and $d_j$ are non-adjacent, and $Q_j$ is an induced path between $c_j$ and $d_j$, the path $Q_j$ must have length 2, and its only inner vertex is a literal vertex. Recall that we only added

edges between distinct literal vertices when constructing the graph $G$ from $G_\varphi$. Hence the path $Q_j$ exists in $G_\varphi$.

Now let $i \in \{1, \ldots, n\}$. We consider the path $P_i$ between $v_i$ and $w_i$. As we observed earlier, the path $P_i$ contains at most two inner vertices, and all inner vertices of $P_i$ are literal vertices. If $P_i$ has exactly one inner vertex, then $P_i$ exists in $G_\varphi$ for the same reason as why the path $Q_j$ from the previous paragraph exists in $G_\varphi$. Suppose $P_i$ has two inner vertices. Recall the two vertex-disjoint paths $L_i^+$ and $L_i^-$ between $v_i$ and $w_i$, respectively, that were defined just above Theorem 2. Since $v_i$ appears in at most three different clauses, at least once positively and at least once negatively, one of these paths has length 2, while the other path has length 2 or 3. Without loss of generality, suppose $L_i^+$ has length 2, and let $\ell$ denote the only inner vertex of $L_i^+$. Note that both $v_i$ and $w_i$ are adjacent to $\ell$. Since $P_i$ is an induced path from $v_i$ to $w_i$ with exactly two inner vertices, $P_i$ cannot contain the vertex $\ell$. From the construction of $G$, it is then clear that both inner vertices of $P_i$ must lie on the path $L_i^-$. This implies that $L_i^-$ must have length 3, and that $P_i = L_i^-$. We conclude that the path $P_i$ exists in $G_\varphi$. □

We now prove the analogue of Theorem 3 for Edge-Disjoint Paths.

**Theorem 4.** *The* Edge-Disjoint Paths *problem is NP-complete on split graphs.*

*Proof.* We reduce from Edge-Disjoint Paths on general graphs, which is well-known to be NP-complete [16]. Let $(G, \mathcal{X})$ be an instance of Edge-Disjoint Paths, where $\mathcal{X} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$. Let $G'$ be the graph obtained from $G$ by adding, for every $i \in \{1, \ldots, k\}$, two new vertices $s_i'$ and $t_i'$ as well as two edges $s_i's_i$ and $t_i't_i$. Let $\mathcal{X}' = \{(s_1', t_1'), \ldots, (s_k', t_k')\}$. Clearly, $(G, \mathcal{X})$ is a yes-instance of Edge-Disjoint Paths if and only if $(G', \mathcal{X}')$ is a yes-instance of Edge-Disjoint Paths. From $G'$, we create a split graph $G''$ as follows. For every pair of vertices $u, v \in V(G)$ such that $uv \notin E(G)$, we add to $G'$ the edge $uv$ as well as two new terminals $s_{uv}, t_{uv}$. Let $G''$ be the resulting graph, let $\mathcal{Q} = \{(s_{uv}, t_{uv}) \mid u, v \in V(G), uv \notin E(G)\}$ be the terminal pairs that were added to $G'$ to create $G''$, and let $\mathcal{X}'' = \mathcal{X}' \cup \mathcal{Q}$. We claim that $(G'', \mathcal{X}'')$ and $(G', \mathcal{X}')$ are equivalent instances of Edge-Disjoint Paths.

Since $G''$ is a supergraph of $G'$, it is clear that $(G'', \mathcal{X}'')$ is a yes-instance of Edge-Disjoint Paths if $(G', \mathcal{X}')$ is. For the reverse direction, suppose that $(G'', \mathcal{X}'')$ is a yes-instance. For every pair $(s_{uv}, t_{uv}) \in \mathcal{Q}$, let $P_{uv}$ be unique path of length 3 in $G''$ between $s_{uv}$ and $t_{uv}$, and let $\mathcal{P}'$ be the set consisting of these paths. It can be shown that there is a solution $\mathcal{P}$ for $(G'', \mathcal{X}'')$ such that $\mathcal{P}' \subseteq \mathcal{P}$. Note that the paths in $\mathcal{P}'$ contain all the edges that were added between non-adjacent vertices in $G'$ in the construction of $G''$. This implies that for every $(s, t) \in \mathcal{X}'$, the path in $\mathcal{P}$ connecting $s$ to $t$ contains only edges that already existed in $G'$. Hence $\mathcal{P} \setminus \mathcal{P}'$ is a solution for the instance $(G', \mathcal{X}')$. □

## 4 Two Polynomial Kernels

In this section, we present polynomial kernels for Vertex-Disjoint Paths and Edge-Disjoint Paths on split graphs, parameterized by the number of terminal pairs.

Before we present the kernels, we introduce some additional terminology. Let $(G, \mathcal{X}, k)$ be an instance of either the Vertex-Disjoint Paths problem or the Edge-Disjoint Paths problem, where $\mathcal{X} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$. Every vertex in the set $\{s_1, \ldots, s_k, t_1, \ldots, t_k\}$ is called a *terminal*. If $s_i = v$ (resp. $t_i = v$) for some $v \in V(G)$, then we say that $s_i$ (resp. $t_i$) is a *terminal on* $v$; note that, in general, there can be more than one terminal on $v$. A vertex $v \in V(G)$ is a *terminal vertex* if there is at least one terminal on $v$, and $v$ is a *non-terminal vertex* otherwise. Given a path $P$ in $G$ and a vertex $v \in V(G)$, we say that $P$ *visits* $v$ if $v \in V(P)$.

### 4.1 Polynomial Kernel for Vertex-Disjoint Paths on Split Graphs

Our kernelization algorithm for Vertex-Disjoint Paths on split graphs consists of four reduction rules. In each of the rules below, we let $(G, \mathcal{X}, k)$ denote the instance of Vertex-Disjoint Paths on which the rule is applied, where we fix a split partition $(C, I)$ of $G$. The instance that is obtained after the application of the rule on $(G, \mathcal{X}, k)$ is denoted by $(G', \mathcal{X}', k')$. We say that a reduction rule is *safe* if $(G, \mathcal{X}, k)$ is a yes-instance of Vertex-Disjoint Paths if and only if $(G', \mathcal{X}', k')$ is a yes-instance of this problem. A reduction rule is only applied if none of the previous rules can be applied, i.e., for every $i \in \{2, 3, 4\}$, Rule $i$ is applied only if Rule $j$ cannot be applied for any $j \in \{1, \ldots, i-1\}$.

**Rule 1.** *If there exists a terminal vertex $v \in V(G)$ such that $v = s_i = t_i$ for some terminal pair $(s_i, t_i) \in \mathcal{X}$, then we set $\mathcal{X}' = \mathcal{X} \setminus \{(s_i, t_i)\}$ and $k' = k - 1$. If $v$ becomes a non-terminal vertex, then we set $G' = G - v$; otherwise, we set $G' = G$.*

**Rule 2.** *If there exists a non-terminal vertex $v \in I$, then we set $G' = G - v$, $\mathcal{X}' = \mathcal{X}$, and $k' = k$.*

**Lemma 1.** *Both Rule 1 and Rule 2 are safe.*

*Proof.* Rule 1 is safe since there is no need to find a path between $s_i$ and $t_i$, and we make sure that $v$ cannot serve as an inner vertex of another path. To see why Rule 2 is safe, suppose there exists a non-terminal vertex $v \in I$. It is clear that if $(G', \mathcal{X}', k')$ is a yes-instance of Vertex-Disjoint Paths, then $(G, \mathcal{X}, k)$ is also a yes-instance of Vertex-Disjoint Paths, as $G$ is a supergraph of $G'$. For the reverse direction, suppose $(G, \mathcal{X}, k)$ is a yes-instance of Vertex-Disjoint Paths, and let $\mathcal{P}$ be a minimum solution for this instance. Since all the paths in $\mathcal{P}$ are induced and $v$ is not a terminal vertex, $v$ is not visited by any of the paths in $\mathcal{P}$. Hence $\mathcal{P}$ is also a solution for the instance $(G', \mathcal{X}', k')$. □

**Rule 3.** *If there exists a terminal vertex $v \in I$ with $d_G(v) \geq 2k - p$, where $p \geq 1$ is the number of terminals on $v$, then we set $G'$ to be the graph obtained from $G$ by deleting all edges incident with $v$, adding $p$ new vertices $\{x_1, \ldots, x_p\}$ to $C$, and making these new vertices adjacent to $v$, to each other, and to all the other vertices in $C$. We also set $\mathcal{X}' = \mathcal{X}$ and $k' = k$.*

**Lemma 2.** *Rule 3 is safe.*

*Proof.* Suppose there exists a terminal vertex $v \in I$ with $d_G(v) \geq 2k - p$, where $p \geq 1$ is the number of terminals on $v$. Let $X = \{x_1, \ldots, x_p\}$ be the set of vertices that were added to $C$ during the execution of the rule. Hence, after the execution of the rule, $X \subseteq C$. Let $Y = \{y_1, \ldots, y_p\}$ be the set of terminals on $v$.

First suppose $(G, \mathcal{X}, k)$ is a yes-instance of Vertex-Disjoint Paths, and let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be an arbitrary solution for this instance. We construct a solution $\mathcal{P}' = \{P'_1, \ldots, P'_k\}$ for $(G', \mathcal{X}', k')$ as follows. Let $i \in \{1, \ldots, k\}$. First suppose that neither of the terminals in the pair $(s_i, t_i)$ belongs to the set $Y$. Since the paths in $\mathcal{P}$ are pairwise vertex-disjoint and $v$ is a terminal vertex, the path $P_i$ does not contain an edge incident with $v$. Hence $P_i$ exists in $G'$, and we set $P'_i = P_i$. Now suppose $v \in \{s_i, t_i\}$. The assumption that Rule 1 cannot be applied implies that $s_i \neq t_i$. Suppose, without loss of generality, that $v = s_i$. Then $s_i \in Y$, so $s_i = y_r$ for some $r \in \{1, \ldots, p\}$. Let $vw$ be the first edge of the path $P_i$ in $G$. We define $P'_i$ to be the path in $G'$ obtained from $P_i$ by deleting the edge $vw$ and adding the vertex $x_r$ as well as the edges $vx_r$ and $x_r w$. Let $\mathcal{P}' = \{P'_1, \ldots, P'_k\}$ denote the collection of paths in $G'$ obtained this way. Since the paths in $\mathcal{P}$ are pairwise vertex-disjoint in $G$, and every vertex in $\{x_1, \ldots, x_p\}$ is visited by exactly one path in $\mathcal{P}'$, it holds that the paths in $\mathcal{P}'$ are pairwise vertex-disjoint in $G'$. Hence $\mathcal{P}'$ is a solution for the instance $(G', \mathcal{X}', k')$.

For the reverse direction, suppose $(G', \mathcal{X}', k')$ is a yes-instance of Vertex-Disjoint Paths, and let $\mathcal{Q} = \{Q_1, \ldots, Q_k\}$ be a minimum solution. Then each of the paths in $\mathcal{Q}$ is an induced path in $G'$. Let $\mathcal{Q}^* \subseteq \mathcal{Q}$ be the set of paths in $\mathcal{Q}$ that visit a vertex in the set $X = \{x_1, \ldots, x_p\}$. Since there are $p$ terminals on $v$, and $v$ has exactly $p$ neighbors in $G'$ (namely, the vertices of $X$), every path in $\mathcal{Q}^*$ has $v$ as one of its endpoints and $|\mathcal{Q}^*| = p$. Moreover, as no vertex in $X$ is a terminal vertex, and the only neighbors of a vertex $x_i \in X$ are $v$ and the vertices of $C \setminus \{x_i\}$, every path in $\mathcal{Q}^*$ visits exactly one vertex of $C \setminus X$. Finally, we observe that each of the $k - p$ paths in $\mathcal{Q} \setminus \mathcal{Q}^*$ visits at most two vertices of $C$ and none of $X$, as $C$ is a clique and every vertex in $X$ is a non-terminal vertex that is already visited by some path in $\mathcal{Q}^*$. Recall that $d_G(v) \geq 2k - p$. Therefore, at least $p$ vertices of $N_G(v)$, say $z_1, \ldots, z_p$, are not visited by any path in $\mathcal{Q}$.

Armed with the above observations, we construct a solution $\mathcal{P} = (P_1, \ldots, P_k)$ for $(G, \mathcal{X}, k)$ as follows. For every path $Q_i \in \mathcal{Q} \setminus \mathcal{Q}^*$, we define $P_i = Q_i$. Now let $Q_i \in \mathcal{Q}^*$. The path $Q_i$ visits $v$, one vertex $x_\ell \in X$, and one vertex $z \in C \setminus X$. If $z \in N_G(v)$, then we define $P_i$ to be the path in $G$ whose single edge is $vz$. If $z \notin N_G(v)$, then we define $P_i$ to be the path obtained from $Q_i$ by replacing the vertex $x_\ell$ by $z_\ell$. It is easy to verify that $\mathcal{P}$ is a solution for the instance $(G, \mathcal{X}, k)$. □

**Rule 4.** *If there exists a non-terminal vertex $v \in C$ that has no neighbors in $I$, then we set $G' = G - v$, $\mathcal{X}' = \mathcal{X}$, and $k' = k$.*

**Lemma 3.** (★) *Rule 4 is safe.*

We now prove that the above four reduction rules yield a quadratic vertex kernel for Vertex-Disjoint Paths on split graphs.

**Theorem 5.** *The* Vertex-Disjoint Paths *problem on split graphs has a kernel with at most* $4k^2$ *vertices, where* $k$ *is the number of terminal pairs.*

*Proof.* We describe a kernelization algorithm for Vertex-Disjoint Paths on split graphs. Let $(G, \mathcal{X}, k)$ be an instance of Vertex-Disjoint Paths, where $G$ is a split graph. We fix a split partition $(C, I)$ of $G$. We then exhaustively apply the four reduction rules, making sure that whenever we apply Rule $i$ for some $i \in \{2, 3, 4\}$, Rule $j$ is not applicable for any $j \in \{1, \ldots, i-1\}$. Let $(G', \mathcal{X}', k')$ be the resulting instance on which none of the reduction rules can be applied. From the description of the reduction rules it is clear that $G'$ is a split graph, and that $\mathcal{X}' = \mathcal{X}$ and $k' \leq k$. By Lemmas 1, 2 and 3, $(G', \mathcal{X}', k')$ is a yes-instance of Vertex-Disjoint Paths if and only if $(G, \mathcal{X}, k)$ is a yes-instance. Hence, the algorithm indeed reduces any instance of Vertex-Disjoint Paths to an equivalent instance.

We now determine an upper bound on the number of vertices in $G'$. Let $(C', I')$ be the unique partition of $V(G')$ into a clique $C'$ and an independent set $I'$ such that $I' = V(G') \cap I$, i.e., the independent set $I'$ contains exactly those vertices of $I$ that were not deleted during any application of the reduction rules. Since Rule 2 cannot be applied, every vertex in $I'$ is a terminal vertex, so $|I'| \leq 2k$. Similarly, since Rules 3 and 4 cannot be applied, every vertex in $I'$ has degree at most $2k - 2$ and every vertex in $C'$ has at least one neighbor in $I'$, implying that $|C'| \leq 2k(2k - 2)$. This shows that $|V(G')| \leq 4k^2 - 2k \leq 4k^2$.

It remains to argue that the above algorithm runs in polynomial time. Rule 1 is applied at most $k$ times. Rules 2 and 3 together are applied at most $|I|$ times in total, as each vertex in $I$ is considered only once. Since every vertex $x_i$ that is created in an application of Rule 2 has exactly one neighbor in $I$, Rule 4 is never applied on such a vertex. Consequently, Rule 4 is applied at most $|C|$ times. This means that the algorithm executes all the reduction rules no more than $k + |I| + |C| = k + |V(G)|$ times in total. Since each of the reduction rules can trivially be executed in polynomial time, the overall running time of the kernelization algorithm is polynomial. □

### 4.2 Polynomial Kernel for Edge-Disjoint Paths on Split Graphs

In this section, we present a kernel with $O(k^3)$ vertices for the Edge-Disjoint Paths problem on split graphs. We need the following two structural lemmas.

**Lemma 4.** (★) *Let* $(G, \mathcal{X}, k)$ *be an instance of* Edge-Disjoint Paths *such that* $G$ *is a complete graph. If* $|V(G)| \geq 2k$*, then* $(G, \mathcal{X}, k)$ *is a yes-instance.*

**Lemma 5.** *Let* $(G, \mathcal{X}, k)$ *be an instance of* Edge-Disjoint Paths *such that* $G$ *is a split graph with split partition* $(C, I)$*,* $\mathcal{X} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ *and* $s_i \neq t_i$ *for every* $i \in \{1, \ldots, k\}$*. If the degree of every terminal vertex is at least the number of terminals on it and* $|C| \geq 2k$*, then* $(G, \mathcal{X}, k)$ *is a yes-instance.*

*Proof.* The proof of this lemma consists of two steps: project to $C$, and route within $C$. In the first step, we project the terminals to $C$. Consider any terminal vertex $x \in I$. For each terminal on $x$, we project it to a neighbor of $x$ in such a way that no two terminals on $x$ are projected to the same vertex; if the terminal is $s_i$, denote this neighbor by $s'_i$, and if the terminal is $t_i$, denote this neighbor by $t'_i$. Since the degree of every terminal vertex is at least the number of terminals on it, this is indeed possible. For any terminal $s_i$ that is on a terminal vertex in $C$, let $s'_i = s_i$, and for any terminal $t_i$ that is on a terminal vertex in $C$, let $t'_i = t_i$. Let $\mathcal{X}' = \{(s'_i, t'_i) \mid i = 1, \ldots, k\}$, and let $G' = G[V(G) \setminus I]$.

Since $G'$ is a complete graph and $|V(G')| = |C| \geq 2k$, there exists a solution $\mathcal{P}' = (P'_1, \ldots, P'_k)$ for the instance $(G', \mathcal{X}', k)$ due to Lemma 4. We now show that we can extend the paths in $\mathcal{P}'$ to obtain a solution $\mathcal{P}$ for the instance $(G, \mathcal{X}, k)$. For every $i \in \{1, \ldots, k\}$, we extend the path $P'_i$ using the edges $s_i s'_i$ (if $s_i \neq s'_i$) and $t_i t'_i$ (if $t_i \neq t'_i$); let the resulting path be $P_i$. Since for every terminal vertex $x \in I$, no two terminals on $x$ were projected to the same neighbor of $x$, the paths in $\mathcal{P}$ are pairwise edge-disjoint. We conclude that $(G, \mathcal{X}, k)$ is a yes-instance. □

Our kernelization algorithm for Edge-Disjoint Paths on split graphs consists of two reduction rules. In each of the two reduction rules below, we let $(G, \mathcal{X}, k)$ denote the instance on which the rule is applied, where we fix a split partition $(C, I)$ of $G$, and assume that $\mathcal{X} = \{(s_i, t_i) \mid i = 1, \ldots, k\}$. The instance that is obtained after the application of a rule is denoted by $(G', \mathcal{X}', k')$. A reduction rule is *safe* if $(G, \mathcal{X}, k)$ is a yes-instance of Edge-Disjoint Paths if and only if $(G', \mathcal{X}', k')$ is a yes-instance of this problem. Reduction Rule B is only applied if Rule A cannot be applied on the same instance.

**Rule A.** *If $s_i = t_i$ for some terminal pair $(s_i, t_i) \in \mathcal{X}$, then we set $G' = G$, $\mathcal{X}' = \mathcal{X} \setminus \{(s_i, t_i)\}$, and $k' = k - 1$.*

Rule A is trivially safe. Suppose now that $(G, \mathcal{X}, k)$ is an instance on which Rule A cannot be applied, that is, $s_i \neq t_i$ for every $i \in \{1, \ldots, k\}$. Let $(C, I)$ be an arbitrary split partition of $G$. If $|C| \geq 2k$, then Lemma 5 ensures that $(G, \mathcal{X}, k)$ is a yes-instance, so our kernelization algorithm will output a trivial yes-instance. If $|C| \leq 2k - 1$, then we still need to upper bound the size of $I$ by a polynomial in $k$ in order to obtain a polynomial kernel. This is exactly what the second reduction rule will achieve. Before we describe the rule, we need to define an auxiliary graph.

Let $T$ be the set of all terminal vertices in $G$. We construct an auxiliary bipartite graph $H = (I \setminus T, A, F)$, where $I \setminus T$ and $A$ are the two sides of the bipartition and $F$ is the set of edges. Here, the set $A$ is defined as follows: for each pair $v, w$ of vertices of $C$, we add vertices $a^{vw}_1, \ldots, a^{vw}_{4k+1}$. The set $F$ is then constructed by, for each $x \in I \setminus T$, adding an edge from $x$ to all $a^{vw}_1, \ldots, a^{vw}_{4k+1}$ if and only if $x$ is adjacent to both $v$ and $w$ in $G$.

Using the graph $H$, we can now define our second rule. Here, given a matching $M$ of $H$, we say that $x \in I$ is *covered* by $M$ if $x$ is an endpoint of an edge in $M$.

**Rule B.** *Let $M$ be any maximal matching of $H$, and let $R$ be the set of vertices of $I \setminus T$ that are not covered by $M$. We set $G' = G - R$, $\mathcal{X}' = \mathcal{X}$, and $k' = k$.*

**Lemma 6.** *Rule B is safe.*

*Proof.* It is clear that if $(G', \mathcal{X}', k')$ is a yes-instance of Edge-Disjoint Paths, then $(G, \mathcal{X}, k)$ is also a yes-instance of Edge-Disjoint Paths, as $G$ is a supergraph of $G'$. For the reverse direction, suppose that $(G, \mathcal{X}, k)$ is a yes-instance of Edge-Disjoint Paths. Note that there exists a solution for $(G, \mathcal{X}, k)$ such that no path in the solution visits a vertex more than once. Among all such solutions, let $\mathcal{P} = (P_1, \ldots, P_k)$ be one for which the total number of visits by all paths combined to vertices from $R$ is minimized. We claim that no path in $\mathcal{P}$ visits a vertex in $R$.

For contradiction, suppose that some path $P_j \in \mathcal{P}$ visits some vertex $r \in R$. Since $r \notin T$, there are two vertices $v, w \in C$ such that the edges $vr$ and $wr$ appear consecutively in the path $P_j$. As $r \in R$, it is not covered by the maximal matching $M$ used in Rule B. Since $r$ is adjacent to all the vertices in $\{a_1^{vw}, \ldots, a_{4k+1}^{vw}\}$ and $M$ is a maximal matching, $M$ covers all the vertices in $\{a_1^{vw}, \ldots, a_{4k+1}^{vw}\}$, and consequently at least $4k + 1$ vertices of $I \setminus T$ that are adjacent to both $v$ and $w$. Let $Z$ denote this set of vertices. Note that all vertices in $Z$ are adjacent to both $v$ and $w$ by the construction of $H$. By the choice of $\mathcal{P}$, no path of $\mathcal{P}$ visits a vertex twice. Hence, there are at most $4k$ edges of $\bigcup_{i=1}^{k} E(P_i)$ incident with $v$ or $w$ in $G$. Therefore, there exists a vertex $z \in Z$ such that $\bigcup_{i=1}^{k} E(P_i)$ contains neither the edge $vz$ nor the edge $wz$. Let $P_j'$ be the path obtained from $P_j$ by replacing $r$ with $z$ and shortcutting it if necessary (i.e., if $z \in V(P_j)$). Then, $\mathcal{P}' = (P_1, \ldots, P_{j-1}, P_j', P_{j+1}, \ldots, P_k)$ is a solution for $(G, \mathcal{X}, k)$ where each path visits each vertex at most once, and where the total number of visits by all paths combined to vertices from $R$ is at least one smaller than $\mathcal{P}$, contradicting the choice of $\mathcal{P}$. Therefore, no path of $\mathcal{P}$ visits a vertex of $R$. Hence, $\mathcal{P}$ is also a solution for $(G', \mathcal{X}', k')$, and thus it is a yes-instance. □

Rules A and B, together with Lemmas 4 and 5, yield the following result.

**Theorem 6.** (★) *The* Edge-Disjoint Paths *problem on split graphs has a kernel with at most* $8k^3$ *vertices, where* $k$ *is the number of terminal pairs.*

## 5 Conclusion

It would be interesting to investigate whether or not Vertex-Disjoint Paths or Edge-Disjoint Paths admits a *linear* kernel on split graphs. Another interesting open question is whether either problem admits a polynomial kernel on chordal graphs, a well-known superclass of split graphs.

Bodlaender et al. [3] asked whether or not Vertex-Disjoint Paths admits a polynomial kernel when restricted to planar graphs. What about the Edge-Disjoint Paths problem on planar graphs?

## References

1. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. J. Comp. Syst. Sci. 75(8), 423–434 (2009)

2. Bodlaender, H.L., Fomin, F.V., Lokshtanov, D., Penninkx, E., Saurabh, S., Thilikos, D.M. (Meta) kernelization. In: 50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, pp. 629–638. IEEE Computer Society (2009)
3. Bodlaender, H.L., Thomasse, S., Yeo, A.: Kernel bounds for disjoint cycles and disjoint paths. Theor. Comp. Sci. 412(35), 4570–4578 (2011)
4. Brandstädt, A., Le, V.B., Spinrad, J.: Graph Classes: A Survey. SIAM Monographs on Discrete Mathematics and Applications (1999)
5. Diestel, R.: Graph Theory, Electronic edn. Springer (2005)
6. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Monographs in Computer Science. Springer (1999)
7. Dvorak, Z., Král', D., Thomas, R.: Three-coloring triangle-free planar graphs in linear time. In: Mathieu, C. (ed.) SODA 2009, pp. 1176–1182. ACM-SIAM (2009)
8. Even, S., Itai, A., Shamir, A.: On the complexity of timetable and multicommodity flow problems. SIAM J. Comp. 5, 691–703 (1976)
9. Frank, A.: Packing paths, circuits, and cuts – a survey. In: Korte, B., Lovász, L., Prömel, H.J., Schrijver, A. (eds.) Paths, Flows, and VLSI-Layout, pp. 47–100. Springer, Berlin (1990)
10. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Annals of Disc. Math. 57 (2004)
11. Gurski, F., Wanke, E.: Vertex disjoint paths on clique-width bounded graphs. Theor. Comput. Sci. 359, 188–199 (2006)
12. 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)
13. Karp, R.M.: On the complexity of combinatorial problems. Networks 5, 45–68 (1975)
14. Kawarabayashi, K., Kobayashi, Y., Reed, B.A.: The disjoint paths problem in quadratic time. J. Comb. Theory B 102, 424–435 (2012)
15. Kobayashi, Y., Kawarabayashi, K.: Algorithms for finding an induced cycle in planar graphs and bounded genus graphs. In: Mathieu, C. (ed.) SODA 2009, pp. 1146–1155. ACM-SIAM (2009)
16. Kramer, M., van Leeuwen, J.: The complexity of wirerouting and finding minimum area layouts for arbitrary VLSI circuits. Adv. Comput. Res. 2, 129–146 (1984)
17. Lynch, J.F.: The equivalence of theorem proving and the interconnection problem. ACM SIGDA Newsletter 5(3), 31–36 (1975)
18. Natarajan, S., Sprague, A.P.: Disjoint paths in circular arc graphs. Nordic J. Comp. 3, 256–270 (1996)
19. Nishizeki, T., Vygen, J., Zhou, X.: The edge-disjoint paths problem is NP-complete for series-parallel graphs. Discrete Applied Math. 115, 177–186 (2001)
20. Reed, B.A.: Tree width and tangles: A new connectivity measure and some applications. In: Bailey, R.A. (ed.) Surveys in Combinatorics, pp. 87–162. Cambridge University Press (1997)
21. Reed, B.A., Robertson, N., Schrijver, A., Seymour, P.D.: Finding disjoint trees in planar graphs in linear time. In: Contemp. Math., vol. 147, pp. 295–301. Amer. Math. Soc., Providence (1993)
22. Robertson, N., Seymour, P.D.: Graph minors XIII. The disjoint paths problem. J. Comb. Theory B 63(1), 65–110 (1995)
23. Tovey, C.A.: A simplified NP-complete satisfiability problem. Discrete Applied Math. 8, 85–89 (1984)
24. Vygen, J.: Disjoint paths. Technical report 94816, Research Institute for Discrete Mathematics, University of Bonn (1998)