Algorithmica (2023) 85:2580–2604
https://doi.org/10.1007/s00453-023-01109-z

Induced Disjoint Paths and Connected Subgraphs for *H*-Free Graphs

Barnaby Martin[1] · **Daniël Paulusma**[1] · **Siani Smith**[1] · **Erik Jan van Leeuwen**[2]

Received: 16 July 2022 / Accepted: 20 February 2023 / Published online: 11 March 2023

Abstract
Paths $P^1, \ldots, P^k$ in a graph $G = (V, E)$ are mutually induced if any two distinct P^i and P^j have neither common vertices nor adjacent vertices. The Induced Disjoint Paths problem is to decide if a graph G with k pairs of specified vertices (s_i, t_i) contains k mutually induced paths P^i such that each P^i starts from s_i and ends at t_i. This is a classical graph problem that is NP-complete even for $k = 2$. We introduce a natural generalization, Induced Disjoint Connected Subgraphs: instead of connecting pairs of terminals, we must connect sets of terminals. We give almost-complete dichotomies of the computational complexity of both problems for H-free graphs, that is, graphs that do not contain some fixed graph H as an induced subgraph. Finally, we give a complete classification of the complexity of the second problem if the number k of terminal sets is fixed, that is, not part of the input.

Keywords Induced subgraphs · Connectivity · H-free graph · Complexity dichotomy

An extended abstract has appeared in the proceedings of WG 2022 [22].

✉ Daniël Paulusma
daniel.paulusma@durham.ac.uk

Barnaby Martin
barnaby.d.martin@durham.ac.uk

Siani Smith
siani.smith@durham.ac.uk

Erik Jan van Leeuwen
e.j.vanleeuwen@uu.nl

[1] Department of Computer Science, Durham University, Durham, UK

[2] Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands

1 Introduction

The well-known Disjoint Paths problem is one of the problems in Karp's list of NP-complete problems. It is to decide if a graph has pairwise vertex-disjoint paths $P^1, \ldots, P^k$ where each P^i connects two pre-specified vertices s_i and t_i. Its generalization, Disjoint Connected Subgraphs, plays a crucial role in the graph minor theory of Robertson and Seymour. This problem asks for connected subgraphs $D^1, \ldots, D^k$, where each D^i connects a pre-specified set of vertices Z_i. In a recent paper [17] we classified, subject to a small number of open cases, the complexity of both these problems for H-*free* graphs, that is, for graphs that do not contain some fixed graph H as an *induced* subgraph.

Our Focus We consider the *induced* variants of Disjoint Paths and Disjoint Connected Subgraphs. These problems behave differently. Namely, Disjoint Paths for fixed k, or more generally, Disjoint Connected Subgraphs, after fixing both k and $\ell = \max\{|Z_1|, \ldots, |Z_k|\}$, is polynomial-time solvable [28]. In contrast, Induced Disjoint Paths is NP-complete even when $k = 2$, as shown both by Bienstock [2] and Fellows [5]. Just as for the classical problems [17], we perform a systematic study and focus on H-free graphs. As it turns out, for the restriction to H-free graphs, the induced variants actually become computationally easier for an infinite family of graphs H. We first give some definitions.

Terminology For a subset $S \subseteq V$ in a graph $G = (V, E)$, let $G[S]$ denote the subgraph of G induced by S, that is, $G[S]$ is the graph obtained from G after removing every vertex not in S. Let $G_1 + G_2$ be the disjoint union of two vertex-disjoint graphs G_1 and G_2. We say that paths $P^1, \ldots, P^k$, for some $k \geq 1$, are *mutually induced paths* of G if there exists a set $S \subseteq V$ such that $G[S] = P^1 + \ldots + P^k$; note that every P^i is an induced path and that there is no edge between two vertices from different paths P^i and P^j. A path P is an s-t-*path* (or t-s-path) if the end-vertices of P are s and t.

A *terminal pair* (s, t) is an unordered pair of two distinct vertices s and t in a graph G, which we call *terminals*. A set $T = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of terminal pairs of G is a *terminal pair collection* if the terminals pairs are pairwise disjoint, so, apart from $s_i \neq t_i$ for $i \in \{1, \ldots, k\}$, we also have $\{s_i, t_i\} \cap \{s_j, t_j\} = \emptyset$ for every $1 \leq i < j \leq k$. We now define the following decision problem:

Induced Disjoint Paths	
Instance:	a graph G and terminal pair collection $T = \{(s_1, t_1) \ldots, (s_k, t_k)\}$.
Question:	does G have a set of mutually induced paths P^1,…,P^k such that P^i is an s_i-t_i path for $i \in \{1, \ldots, k\}$?

Note that as every path between two vertices s and t contains an induced path between s and t, the condition that every P^i must be induced is not strictly needed in the above problem definition. We say that the paths $P^1, \ldots, P^k$, if they exist, form a *solution* of Induced Disjoint Paths.

We now generalize the above notions from pairs and paths to sets and connected subgraphs. Subgraphs $D^1, \ldots, D^k$ of a graph $G = (V, E)$ are *mutually induced subgraphs* of G if there exists a set $S \subseteq V$ such that $G[S] = D^1 + \ldots + D^k$.

A connected subgraph D of G is a *Z-subgraph* if $Z \subseteq V(D)$. A *terminal set* Z is an unordered set of distinct vertices, which we again call *terminals*. A set $\mathcal{Z} = \{Z_1, \ldots, Z_k\}$ is a *terminal set collection* if $Z_1, \ldots, Z_k$ are pairwise disjoint terminal sets. We now introduce the generalization:

Induced Disjoint Connected Subgraphs	
Instance:	a graph G and terminal set collection $\mathcal{Z} = \{Z_1, \ldots, Z_k\}$.
Question:	does G have a set of mutually induced connected subgraphs $D^1, \ldots, D^k$ such that D^i is a Z_i-subgraph for $i \in \{1, \ldots, k\}$?

The subgraphs $D^1, \ldots, D^k$, if they exist, form a *solution*. We write Induced Disjoint Connected ℓ- Subgraphs if $\ell = \max\{|Z_1|, \ldots, |Z_k|\}$ is fixed. Note that Induced Disjoint Connected 2- Subgraphs is exactly Induced Disjoint Paths.

1.1 Known Results

Only results for Induced Disjoint Paths are known, and these sometimes hold for a slightly more general problem definition where terminals are no longer required to form an independent set (see also Sect. 6). Namely, Induced Disjoint Paths is linear-time solvable for circular-arc graphs [10]; polynomial-time solvable for chordal graphs [1], AT-free graphs [11], graph classes of bounded mim-width [13]; and NP-complete for claw-free graphs [6], line graphs of triangle-free chordless graphs [27] and thus for (theta,wheel)-free graphs, and for planar graphs; the last result follows from a result of Lynch [21] (see [11]). Moreover, Induced Disjoint Paths is XP with parameter k for (theta,wheel)-free graphs [27] and even FPT with parameter k for claw-free graphs [9] and planar graphs [15]; the latter can be extended to graph classes of bounded genus [18].

1.2 Our Results

Let P_r be the path on r vertices. A *linear forest* is the disjoint union of one or more paths. We write $F \subseteq_i G$ if F is an induced subgraph of G and sG for the disjoint union of s copies of G. We can now present our first two results: the first one includes our dichotomy for Induced Disjoint Paths (take $\ell = 2$).

Theorem 1 *Let $\ell \geq 2$. For a graph H,* Induced Disjoint Connected ℓ-Subgraphs *on H-free graphs is polynomial-time solvable if $H \subseteq_i sP_3 + P_6$ for some $s \geq 0$; NP-complete if H is not a linear forest; and quasipolynomial-time solvable otherwise.*

Theorem 2 *For a graph H such that $H \neq sP_1 + P_6$ for some $s \geq 0$,* Induced Disjoint Connected Subgraphs *on H-free graphs is polynomial-time solvable for H-free graphs if $H \subseteq_i sP_1 + P_3 + P_4$ or $H \subseteq_i sP_1 + P_5$ for some $s \geq 0$, and it is NP-complete otherwise.*

Note the complexity jumps if we no longer fix ℓ. We will show that all the open cases in Theorem 2, so the cases where $H = sP_1 + P_6$ for any $s \geq 0$, are equivalent to exactly **one** open case, namely $H = P_6$.
Comparison The Disjoint Connected Subgraphs problem restricted to H-free graphs is polynomial-time solvable if $H \subseteq_i P_4$ and else it is NP-complete, even if the maximum size of the terminal sets is $\ell = 2$, except for the three unknown cases $H \in \{3P_1, 2P_1 + P_2, P_1 + P_3\}$ [17]. Perhaps somewhat surprisingly, Theorems 1 and 2 show the induced variant is computationally easier for an infinite number of linear forests H (assuming P $\neq$ NP and that some problems in NP are not quasipolynomial-time solvable).

Fixing k If the number k of terminal sets is fixed, we write k- Induced Disjoint Connected Subgraphs and prove the following complete dichotomy.

Theorem 3 *Let $k \geq 2$. For a graph H, k- Induced Disjoint Connected Subgraphs on H-free graphs is polynomial-time solvable for H-free graphs if $H \subseteq_i sP_1 + 2P_4$ or $H \subseteq_i sP_1 + P_6$ for some $s \geq 0$, and it is NP-complete otherwise.*

Comparison We note a complexity jump between Theorems 2 and 3 when $H = sP_1 + 2P_4$ for some $s \geq 0$.
Remark The polynomial functions that bound the running times of the algorithms in Theorems 1 and 3 have k, respectively, ℓ in the exponent.
Paper Outline Section 2 contains terminology, known results and auxiliary results that we will use as lemmas. Hardness results for Theorem 1 transfer to Theorem 2, whereas the reverse holds for polynomial results. As such, we show all our polynomial-time algorithms in Sect. 3 and all our hardness reductions in Sect. 4. The cases $H = sP_3 + P_6$ in Theorem 1 and $H = sP_1 + P_5$ in Theorem 2 are proven by a reduction from Independent Set via so-called *blob graphs*, just as the quasipolynomial-time result if H is a linear forest. Hence, we also include the proof of the latter result in Sect. 3. In Sect. 5 we combine the results from the previous two sections to prove Theorems 1–3.

In our theorems we have infinite families of polynomial cases related to nearly H-free graphs. For a graph H, a graph G is *nearly H-free* if G is $(P_1 + H)$-free. It is easy to see (cf [3]) that Independent Set is polynomial-time solvable on nearly H-free graphs if it is so on H-free graphs. However, for many other graph problems, this might either not be true or less easy to prove (see, for example, [14]). In Sect. 3 we show that it holds for the relevant cases in Theorem 2, in particular for the case $H = P_6$ (see Lemma 7). The latter result yields no algorithm but shows that essentially $H = P_6$ is the only one open case left in Theorem 2 (see also Remark 1).

In Sect. 6 we briefly discuss the aforementioned more general problem definition of Induced Disjoint Paths used in the literature. We show that the complexity dichotomy for this problem differs from the one in Theorem 1 (for $\ell = 2$).

In Sect. 7 we consider a number of directions for future work. In particular we consider the restriction k- Disjoint Connected ℓ- Subgraphs where *both* k and ℓ are fixed and discuss some open problems.

2 Preliminaries

Let $G = (V, E)$ be a graph. A subset $S \subseteq V$ is *connected* if $G[S]$ is connected. A subset $D \subseteq V(G)$ is *dominating* if every vertex of $V(G) \setminus D$ is adjacent to least one vertex of D; if $D = \{v\}$ then v is a *dominating* vertex. The *open* and *closed neighbourhood* of a vertex $u \in V$ are $N(u) = \{v \mid uv \in E\}$ and $N[u] = N(u) \cup \{u\}$. For a set $U \subseteq V$ we define $N(U) = \bigcup_{u \in U} N(u) \setminus U$ and $N[U] = N(U) \cup U$.

For a graph $G = (V, E)$ and a subset $S \subseteq U$, we write $G - S = G[V \setminus S]$. If $S = \{u\}$ for some $u \in V$, we write $G - u$ instead of $G - \{u\}$. A vertex u is a *cut-vertex* of a connected graph G if $G - u$ is disconnected.

The *contraction* of an edge $e = uv$ in a graph G replaces the vertices u and v by a new vertex w that is adjacent to every vertex previously adjacent to u or v; note that the resulting graph G/e is still *simple*, that is, G/e contains no multi-edges or self-loops. The following lemma is easy to see (see, for example, [16]).

Lemma 1 *For a linear forest H, let G be an H-free graph. Then G/e is H-free for every $e \in E(G)$.*

In a solution $(D^1, \ldots, D^k)$ for an instance $(G, \mathcal{Z})$ of Induced Disjoint Connected Subgraphs, if D^i is inclusion-wise minimal and X_i is a minimum connected dominating set of D^i, then $X_i \cup Z_i = D^i$ or, equivalently, $D^i \setminus X_i \subseteq Z_i$. This will be relevant in our proofs, where we use the following result of Camby and Schaudt, in particular for the case $r = 6$ (alternatively, we could use the slightly weaker characterization of P_6-free graphs in [32] but the below characterization gives a faster algorithm).

Theorem 4 *([4]) Let $r \geq 4$ and G be a connected P_r-free graph. Let X be any minimum connected dominating set of G. Then $G[X]$ is either P_{r-2}-free or isomorphic to P_{r-2}.*

Let $G = (V, E)$ be a graph. Two sets $X_1, X_2 \subseteq V$ are *adjacent* if $X_1 \cap X_2 \neq \emptyset$ or there exists an edge with one end-vertex in X_1 and the other in X_2. The *blob graph* G° of G has vertex set $\{X \subseteq V(G) \mid X \text{ is connected}\}$ and edge set $\{X_1X_2 \mid X_1 \text{ and } X_2 \text{ are adjacent}\}$. Note that blob graphs may have exponential size, but in our proofs we will only construct parts of blob graphs that have polynomial size. We need the following known lemma that generalizes a result of Gartland et al. [8] for paths.

Lemma 2 *([25]) For every linear forest H, a graph G is H-free if and only if G° is H-free.*

The Independent Set problem is to decide if a graph G has an *independent set* (set of pairwise non-adjacent vertices) of size at least k for some given integer k.

We need the following two known results for Independent Set. The first one is due to Grzesik, Klimosová, Pilipczuk and Pilipczuk [12]. The second one is due to Pilipczuk, Pilipczuk and Rzążewski [26], who improved the previous quasipolynomial-time algorithm for Independent Set on P_t-free graphs, due to Gartland and Lokshtanov [7] (whose algorithm runs in $n^{O(\log^3 n)}$ time).

Theorem 5 *([12])* INDEPENDENT SET *is polynomial-time solvable for P_6-free graphs.*

Theorem 6 *([26]) For every $r \geq 1$,* INDEPENDENT SET *can be solved in $n^{O(\log^2 n)}$ time for P_r-free graphs.*

Two instances of some decision problem are *equivalent* if one is a yes-instance if and only if the other one is. We make frequently use of the following observation.

Lemma 3 *From an instance $(G, \mathcal{Z})$ of* INDUCED DISJOINT CONNECTED SUBGRAPHS *we can in linear time, either find a solution for $(G, \mathcal{Z})$ or obtain an equivalent instance $(G', \mathcal{Z}')$ with $|V(G')| \leq |V(G)|$, such that the following holds:*

1. *$|\mathcal{Z}'| \geq 2$;*
2. *every $Z'_i \in \mathcal{Z}'$ has size at least 2; and*
3. *the union of the sets in $\mathcal{Z}'$ is an independent set.*

Moreover, if G is H-free for some linear forest H, then G' is also H-free.

Proof Let $(G, \mathcal{Z})$ be an instance of INDUCED DISJOINT CONNECTED SUBGRAPHS, where $\mathcal{Z} = \{Z_1, \ldots, Z_k\}$ for some integer $k \geq 1$. If two adjacent vertices will always appear in the same set of every solution $(D^1, \ldots, D^k)$, then we can safely contract the edge between them at the start of any algorithm. This property holds for every pair of adjacent vertices of every Z_i. Hence, we contract every edge between two vertices that belong to the same set Z_i. This takes linear time.

Let $(G^*, \mathcal{Z}^*)$ be the resulting instance. Note that every $Z \in \mathcal{Z}^*$ is an independent set. If a set $Z \in \mathcal{Z}^*$ has size 1, say $Z = \{z\}$, then we remove z and all its neighbours from G^* to obtain an equivalent instance. After doing this for all singleton sets in $\mathcal{Z}^*$, we obtain our desired instance $(G', \mathcal{Z}')$ in linear time.

If G is H-free for some linear forest H, then G' is also H-free, as H-freeness is preserved by edge contraction (Lemma 1) and by vertex deletion (by definition). If it turns out that two vertices from different sets in $\mathcal{Z}'$ are adjacent, then $(G, \mathcal{Z})$ is a no-instance. Else we find that the union of the sets in $\mathcal{Z}'$ is an independent set. If $|\mathcal{Z}'| = 1$, then the problem is trivial to solve, so we may assume that $|\mathcal{Z}'| \geq 2$. □

We also use the next lemma frequently.

Lemma 4 *Let H be a linear forest. If $(G, \mathcal{Z})$ is a yes-instance of* INDUCED DISJOINT CONNECTED SUBGRAPHS *and G is H-free, then $(G, \mathcal{Z})$ has a solution $(D^1, \ldots, D^k)$, where each D^i has size at most $(2|V(H)| - 1)|Z_i|$.*

Proof Consider a solution $(D^1, \ldots, D^k)$. As G is H-free and H is a linear forest, G is also P_t-free where $t = 2|V(H)| - 1$. For every D^i, fix a vertex u in it. As D^i is connected and G is P_t-free, there exists a path from every vertex of Z_i to u that has length at most $t - 1$. Let A^i be the subgraph of D^i induced by the union of the vertex sets of these paths of D^i. Note that the number of vertices of each A^i is at most $t|Z_i|$. As each A^i is connected, $(A^1, \ldots, A^k)$ is a solution. □

3 Algorithms

In this section we show all the polynomial-time and quasipolynomial-time results needed to prove our main theorems.

3.1 Using the Blob Graph Approach

We start with the following result, which holds for every fixed integer $\ell \geq 2$. In the proof of this result and the next one we use the blob graph approach.

Lemma 5 *Let $\ell \geq 2$. For every $s \geq 0$,* Induced Disjoint Connected ℓ-Subgraphs *is polynomial-time solvable for $(sP_3 + P_6)$-free graphs.*

Proof Let $(G, \mathcal{Z})$ be an instance of the Induced Disjoint Connected ℓ-Subgraphs problem, where G is an $(sP_3 + P_6)$-free graph for some $s \geq 0$. By Lemma 3, we may assume that the union of the sets in $\mathcal{Z} = \{Z_1, \ldots, Z_k\}$ is an independent set.

First suppose that $k \leq s$. By Lemma 4 we may assume that each D^i in a solution $(D^1, \ldots, D^k)$ has size at most $t = (6s + 11)\ell$. So $|D^1| + \ldots + |D^k|$ has size at most $kt \leq st$. Hence, we can consider all $O(n^{st})$ options of choosing a solution. As s and t are constants, this takes polynomial time in total.

Now suppose that $k \geq s + 1$. We consider all $O(n^{(s-1)t})$ options of choosing the first s subgraphs D^i, discarding those with an edge between distinct D^i or between some D^i and Z_j for some $j \geq s + 1$. For each remaining option, let $G' = G - N[V(D^1) \cup \cdots \cup V(D^s)]$ and $\mathcal{Z}' = \{Z_{s+1}, \ldots, Z_k\}$. Note that G' is P_6-free.

Let F be the subgraph of the blob graph G'° induced by all connected subsets X in G' that have size at most 11ℓ, such that X contains all vertices of one set from $\mathcal{Z}'$ and no vertices from any other set of $\mathcal{Z}'$. Then F has polynomial size, as it has $O(n^{11\ell})$ vertices, so we can construct F in polynomial time. By Lemma 2, F is P_6-free.

We claim that $(G', \mathcal{Z}')$ has a solution if and only if F has an independent set of size $k - s$.

First suppose that $(G', \mathcal{Z}')$ has a solution. Then, by Lemma 4, it has a solution $(D^{s+1}, \ldots, D^k)$, where each D^i has size at most 11ℓ. Such a solution corresponds to an independent set of size $k - s$ in F. For the reverse implication, two vertices in F that each contain vertices of the same set Z_i are adjacent. Hence, an independent set of size $k - s$ in F is a solution for $(G', \mathcal{Z}')$.

Due to the above, it remains to apply Theorem 5 to find in polynomial time whether G'° has an independent set of size $k - s$. □

By replacing Theorem 5 by Theorem 6 in the above proof and repeating the arguments of the second part we obtain the following result.

Lemma 6 *Let $\ell \geq 2$. For every $r \geq 1$,* Induced Disjoint Connected ℓ-Subgraphs *is quasipolynomial-time solvable for P_r-free graphs.*

3.2 Nearly H-Free Graphs

In this section we prove a crucial lemma on nearly H-free graphs.

Lemma 7 *For $k \geq 2$, $r \leq 6$ and $s \geq 1$, if (k-)*Induced Disjoint Connected Subgraphs *is polynomial-time solvable for P_r-free, graphs, then it is so for $(sP_1 + P_r)$-free graphs.*

Proof We let $r = 6$, and we let k be part of the input; as will be clear from the proof below, the case where $r \leq 5$ and/or k is fixed can be shown by using exactly the same arguments. Let $(G, \mathcal{Z})$ be an instance of INDUCED DISJOINT CONNECTED SUBGRAPHS, where G is an $(sP_1 + P_6)$-free graph for some integer $s \geq 1$ and $\mathcal{Z} = \{Z_1, \ldots, Z_k\}$. We may assume without loss of generality that $|Z_1| \geq |Z_2| \geq \cdots \geq |Z_k|$. By Lemma 3, we may assume that $k \geq 2$; every $Z_i \in \mathcal{Z}$ has size at least 2; and the union of the sets in $\mathcal{Z}$ is an independent set. We assume that INDUCED DISJOINT CONNECTED SUBGRAPHS is polynomial-time solvable for P_6-free graphs.

Case 1 For every $i \geq 2$, $|Z_i| \leq s - 1$. Let $D^1, \ldots, D^k$ be a solution for $(G, \mathcal{Z})$ (assuming it exists). By Lemma 4, we may assume without loss of generality that for $i \geq 2$, the number of vertices of D^i is at most $(2s + 11)|Z_i| \leq (2s + 11)(s - 1)$.

First assume $k \leq s$. Then $V(D^2) \cup \cdots \cup V(D^k)$ has size at most t, where $t = (s - 1)(2s + 11)(s - 1)$ is a constant. Hence, we can do as follows. We consider all $O(n^t)$ options for choosing the subgraphs $D^2, \ldots, D^k$. For each choice we check in polynomial time if $D^2, \ldots, D^k$ are mutually induced and connected, and if each D^i contains Z^i. We then check in polynomial time if the graph $G - N[(V(D^2) \cup \cdots V(D^k)]$ has a connected component containing Z_1. As the number of choices is polynomial, the total running time is polynomial.

Now assume $k \geq s+1$. We consider all $O(n^{s(2s+11)(s-1)})$ options of choosing the s subgraphs $D^2, \ldots, D^{s+1}$. We discard an option if for some $i \in \{1, \ldots, s\}$, the graph D^i is disconnected. We also discard an option if there is an edge between two vertices from two different subgraphs D^h and D^i for some $2 \leq h < i \leq s + 1$, or if there is an edge between a vertex from some subgraph D^h $(2 \leq h \leq s)$ and a vertex from some set Z_i $(i = 1$ or $i \geq s+2)$. If we did not discard the option, then we solve INDUCED DISJOINT CONNECTED SUBGRAPHS on instance $(G - \bigcup_{i=2}^{s+2} N[V(D^i)], \mathcal{Z} \setminus \{Z_2, \ldots, Z_{s+1}\})$. The latter takes polynomial time as $G - \bigcup_{i=2}^{s+1} N[D^i]$ is P_6-free, due to G being $(sP_1 + P_6)$-free by our assumption. As the number of branches (subproblems) is polynomial as well, the total running time is polynomial.

Case 2 $|Z_2| \geq s$ (and thus also $|Z_1| \geq s$). Let $D^1, \ldots, D^k$ be a solution for $(G, \mathcal{Z})$ (assuming it exists). As $|Z_1| \geq s$, we find that for every $i \geq 2$, D^i is P_6-free. As $|Z_2| \geq s$, we also find that D^1 is P_6-free. Then, by setting $r = 6$ in Theorem 4, every D^i $(i \in \{1, \ldots, k\})$ has a connected dominating set X_i such that $G[X_i]$ is either P_4-free or isomorphic to P_4. We may assume without loss of generality that every X_i is inclusion-wise minimal (as otherwise we could just replace X_i by a smaller connected dominating set of D^i).

Case 2a There exist some X_i with size at least $7s + 2$. As $s \geq 1$, we have that $G[X_i]$ is P_4-free. We now set $r = 4$ in Theorem 4 and find that $G[X_i]$ has a connected dominating set Y_i of size at most 2. Hence, $G[X_i]$ contains a set R of $7s$ vertices that are not cut-vertices of $G[X_i]$. As X_i is minimal, this means that in D^i, each $r \in R$ has at least one neighbour $z \in Z_i$ that is not adjacent to any vertex of $X_i \setminus \{r\}$. We say that z is a *private* neighbour of r. We now partition R into sets $R_1, \ldots, R_7$, each of exactly s vertices. For $h = 1, \ldots, 7$, let $R_h = \{r_h^1, \ldots, r_h^s\}$ and pick a private neighbour z_h^j of r_h^j. For $h = 1, \ldots, 7$, let $Q_h = \{z_h^1, \ldots, z_h^s\}$. Each Q_h is independent, as Z_i is independent and $Q_h \subseteq Z_i$.

We claim that there exists an index $h \in \{1, \ldots, 7\}$ such that $G - (N[Q_h] \setminus R_h)$ is P_6-free. For a contradiction, assume that for every $h \in \{1, \ldots, 7\}$, we have that $G - (N[Q_h] \setminus R_h)$ is not P_6-free. As G is $(sP_1 + P_6)$-free and every Q_h is an independent set of size s, we have that $G - N[Q_h]$ is P_6-free. We conclude that every induced P_6 of G contains a vertex of R_h for every $h \in \{1, \ldots, 7\}$. This is contradiction, as every induced P_6 only has six vertices. Hence, there exists an index $h \in \{1, \ldots, 7\}$ such that $G - (N[Q_h] \setminus R_h)$ is P_6-free.

We exploit the above structural claim algorithmically as follows. We consider all $k \leq n$ options that one of the sets X_i has size at least $7s+2$. For each choice of index i we do as follows. We consider all $O(n^{2s})$ options of choosing a set Q_h of s vertices from the independent set Z_i together with a set R_h of s vertices from $N(Q_h)$. We discard the option if a vertex of Q_h has more than one neighbour in R_h, or if $G' = G-(N[Q_h]\setminus R_h)$ is not P_6-free. Otherwise, we solve Induced Disjoint Connected Subgraphs on instance $(G', \mathcal{Z}')$, where $\mathcal{Z}' = (\mathcal{Z}\setminus\{Z_i\}) \cup \{(Z_i \setminus Q_h) \cup R_h\}$. As G' is P_6-free, the latter takes polynomial time by our initial assumption. Hence, as the total number of branches is $O(n^{2s+1})$ the total running time of this check takes polynomial time.

Case 2b. Every X_i has size at most $7s + 1$. $7s + 2$. First assume $k \leq s$. We consider all $O(n^{s(7s+1)})$ options of choosing the sets $X_1, \ldots, X_k$. For each option we check if $(X_1 \cup Z_1, \ldots, X_k \cup Z_k)$ is a solution for $(G, \mathcal{Z})$. As the latter takes polynomial time and the total number of branches is polynomial, this takes polynomial time.

Now assume $k \geq s + 1$. We consider all $O(n^{s(7s+1)})$ options of choosing the first s sets $X_1, \ldots, X_s$. We discard an option if for some $i \in \{1, \ldots, s\}$, the set $X_i \cup Z_i$ is disconnected. We also discard an option if there is an edge between two vertices from two different sets $X_h \cup Z_h$ and $X_i \cup Z_i$ for some $1 \leq h < i \leq s$, or if there is an edge between a vertex from some set $X_h \cup Z_h$ $(h \leq s)$ and a vertex from some set Z_i $(i \geq s + 1)$. If we did not discard the option, then we solve Induced Disjoint Connected Subgraphs on instance $(G - \bigcup_{i=1}^{s} N[X_i \cup Z_i], \{Z_{s+1}, \ldots, Z_k\})$. The latter takes polynomial time as $G - \bigcup_{i=1}^{s} N[X_i \cup Z_i]$ is P_6-free. As the number of branches is polynomial as well, the total running time is polynomial.

From the above case analysis we conclude that the running time of our algorithm is polynomial. As mentioned, if $r \leq 5$ and/or k is fixed, then we use exactly the same arguments. □

Remark 1 Due to Lemma 7, the missing cases $H = sP_1 + P_6$ in Theorem 2 are all equivalent to the case $H = P_6$.

3.3 Two Applications of Lemma 7

We will first use Lemma 7 for the case where $r = 5$. In the proof of the next result, we also make use of the blob approach again.

Lemma 8 *For every $s \geq 0$,* Induced Disjoint Connected Subgraphs *is polynomial-time solvable for $(sP_1 + P_5)$-free graphs.*

Proof Due to Lemma 7 it suffices to prove the statement for P_5-free graphs only. Let $(G, \mathcal{Z})$ be an instance of Induced Disjoint Connected Subgraphs, where G is a P_5-free graph and $\mathcal{Z} = \{Z_1, \ldots, Z_k\}$. By Lemma 3, we may assume that $k \geq 2$;

every $Z_i \in \mathcal{Z}$ has size at least 2; and the union of the sets in $\mathcal{Z}$ is an independent set. We may also delete every vertex from G that is not in a terminal set from $\mathcal{Z}$ but that is adjacent to two terminals in different sets Z_h and Z_i (such a vertex cannot be used in any subgraph of a solution). We now make a structural observation that gives us a procedure for safely contracting edges; recall that edge contraction preserves P_5-freeness by Lemma 1.

Consider a solution $(D^1 \dots D^k)$ that is *maximal* in the sense that any vertex v outside $V(D^1) \cup \dots \cup V(D^k)$ must have a neighbour in at least two distinct subgraphs D^i and D^j. Since G is P_5-free, v must be adjacent to all vertices of at least one of D^i and D^j. Since v does not have neighbours in both $Z_i \subseteq V(D^i)$ and $Z_j \subseteq V(D^j)$, we find that v is adjacent to all vertices of exactly one of D^i and D^j.

The above gives rise to the following algorithm. Let v be a vertex that is adjacent to at least one vertex $z \in Z_i$ but not to all vertices of Z_i. As v is adjacent to z and z is in Z_i, it holds that v does not belong to any D^h with $h \neq i$ for every (not necessarily maximal) solution $(D^1, \dots, D^k)$. The observation from the previous paragraph tells us that if v is not in any D^h and $(D^1, \dots, D^k)$ is a maximal solution, then v must be adjacent to all vertices of some D^j. As v is adjacent to $z \in Z_i$, it holds by construction that v is not adjacent to any vertex of any $Z_h \subseteq V(D^h)$ with $h \neq i$. Hence, $i = j$ must hold. However, this is not possible, as we assumed that v is not adjacent to all vertices of $Z_i \subseteq V(D^i)$. Hence, we may assume without loss of generality that v belongs to D^i (should a solution exist). This means that we can safely contract the edge vz and put the resulting vertex in Z_i. Then we apply Lemma 3 again and also remove all common neighbours of vertices from Z_i and vertices from other sets Z_j. This takes polynomial time and the resulting graph has one vertex less. Hence, by applying this procedure exhaustively we have, in polynomial time, either solved the problem or obtained an equivalent but smaller instance.

Suppose we have an equivalent instance. For simplicity we denote the obtained instance by $(G, \mathcal{Z})$ again, where G is a P_5-free graph and $\mathcal{Z} = \{Z_1, \dots, Z_k\}$ with $k \geq 2$. Due to our procedure, every $Z_i \in \mathcal{Z}$ has size at least 2; the union of the sets in $\mathcal{Z}$ is an independent set. Moreover, every non-terminal vertex is adjacent either to no terminal vertex or is adjacent to all terminals of exactly one terminal set. We let S be the set of vertices of the latter type. Observe that it follows from the preceding that only vertices of S need to be used for a solution.

We now construct the subgraph F of the blob graph G° that is induced by all connected subsets X of the form $X = Z_i \cup \{s\}$ for some $1 \leq i \leq k$ and $s \in S$. Note that F has $O(kn)$ vertices. Hence, constructing F takes polynomial time. Moreover, F is P_5-free due to Lemma 2. As in the proof of Lemma 5, we observe that $(G, \mathcal{Z})$ has a solution if and only if F has an independent set of size k. It now remains to apply (in polynomial time) Theorem 5. □

We now show a stronger result when k is fixed instead of part of the input. Again we will use Lemma 7.

Lemma 9 *For every integer $s \geq 0$, k-* Induced Disjoint Connected Subgraphs *is polynomial-time solvable for $(sP_1 + P_6)$-free graphs.*

Proof Due to Lemma 7 it suffices to prove the statement for P_6-free graphs only. Let $(G, \mathcal{Z})$ be an instance of k- Induced Disjoint Connected Subgraphs, where G is a P_6-free graph and $\mathcal{Z} = \{Z_1, \ldots, Z_k\}$. By Lemma 3, we may assume that every $Z_i \in \mathcal{Z}$ has size at least 2 and that the union of the sets in $\mathcal{Z}$ is an independent set. We start by deleting from G,

(i) Every common neighbour of a vertex of Z_i and a vertex of Z_j with $i \neq j$.

This takes polynomial time and is safe, as such vertices are not in any subgraph of a solution.

Consider a solution $(D^1, \ldots, D^k)$ (if it exists). As G is P_6-free, each D^i is P_6-free. By Theorem 4 (take $r = 6$), this means that every D^i has a connected dominating set X_i such that $G[X_i]$ is either P_4-free or isomorphic to P_4. In the former case we say that D^i is *difficult* and in the latter we say that D^i is *easy*.

We consider all options of choosing which of the subgraphs D^i is easy and consider all options of choosing the corresponding dominating P_4s. This leads to total number of $O(2^k n^{4k})$ branches (subproblems), which is polynomial as k is fixed. We discard those options that do not result in mutually induced connected subgraphs D^i with $Z^i \subseteq V(D^i)$ and also those with an edge between a vertex of some guessed D^i and some Z_j that will correspond to a difficult subgraph D^j.

For each remaining branch, we delete the vertices of each easy D^i and all their neighbours from G. We also remove the corresponding sets Z_i from $\mathcal{Z}$. For simplicity we denote the new instance by $(G, \mathcal{Z})$ again, and we let $|\mathcal{Z}| = k$. If $k \leq 1$, then we can solve the problem in a trivial way. Suppose that $k \geq 2$. We note that G is still P_6-free; every $Z_i \in \mathcal{Z}$ still has size at least 2 and that the union of the sets in $\mathcal{Z}$ is an independent set such that no two vertices from different Z_i and Z_j have a common neighbour. Moreover, if $(G, \mathcal{Z})$ has a solution, then every connected subgraph in it is difficult.

Consider a solution $(D^1, \ldots, D^k)$ (if it exists). As each D^i is difficult, it contains (by definition) a connected dominating set X_i such that $G[X_i]$ is P_4-free. By Theorem 4 (take $r = 4$), every $G[X_i]$ has a connected dominating set Y_i of size at most 2. We consider all options of choosing the connected sets Y_i. For every chosen Y_i of size 2, we contract the edge between the two vertices of Y_i. The resulting graph is still P_6-free by Lemma 1. Note that the number of branches is $O(n^{2k})$, which is polynomial as k is fixed. In each branch, we may now assume that $Y_i = \{y_i\}$ for $i \in \{1, \ldots, k\}$ and thus y_i will dominate $G[X_i]$. We discard an option if $\{y_1, \ldots, y_k\}$ is not an independent set, or if some y_i is adjacent to a vertex of some Z_j with $i \neq j$. For every other branch we continue as follows. From G we first delete

(ii) For every $i \in \{1, \ldots, k\}$, every neighbour of y_i adjacent to a vertex of $Z_j \cup \{y_j\}$ for some $j \neq i$.

This takes polynomial time and we may do this, as these common neighbours cannot belong to any subgraph in a solution $(D^1, \ldots, D^k)$ with $y_i \in V(D^i)$ for $i \in \{1, \ldots, k\}$.

We may assume without loss of generality that in the solution $(D^1, \ldots, D^k)$ we are looking for the subgraphs that are minimal. That is, we cannot replace a subgraph D^i with some subgraph F^i with $V(F^i) \subset V(D^i)$. So each vertex x of $X_i \setminus \{y_i\}$ is either a vertex of Z_i or has a neighbour z_i in Z_i that is not adjacent to any vertex of $X_i \setminus \{x_i\}$;

in the latter case we say that z_i is a *private* neighbour of x_i. For every $i \in \{1, \ldots, k\}$, at least one such pair (x_i, z_i) exists, as otherwise y_i dominates D^i, and thus D^i would not be difficult.

We now consider for $i \in \{1, \ldots, k\}$, all options of choosing the pairs (x_i, z_i) where x_i is a neighbour of y_i that is not from Z_i, whilst z_i is taken from Z_i. We discard those options where x_i and z_i are not adjacent or where z_i is adjacent to y_i; in both cases z_i will not be a private neighbour of x_i. We also discard every option where the graphs $G[\{y_i, x_i, z_i\}]$ are not mutually induced. Note that the number of branches is $O(n^{2k})$ (so polynomial). For each branch that we have not discarded we continue by first deleting from G, the following sets of vertices:

(iii) For $i \in \{1, \ldots, k\}$, every neighbour of z_i not equal to x_i;
(iv) For $i \in \{1, \ldots, k\}$, every neighbour of x_i that is adjacent to a vertex of $Z_j \cup \{x_j, y_j\}$ for some $j \neq i$;

This takes polynomial time. Moreover, we are allowed to delete all these vertices, as none of them can be used in the solution that we are trying to construct. In particular, every z_i will be a private neighbour of x_i, so every z_i has only one neighbour in $V(D^1) \cup \cdots \cup V(D^k)$. We now use that we have performed the operations of (i)–(iv) to prove the following claim:
A solution for this branch exists if and only if $N_G[y_i]$ dominates Z_i for every $i \in \{1, \ldots, k\}$.
First suppose that $(G, \mathcal{Z})$ has a solution $(D^1, \ldots, D^k)$ such that for every $i \in \{1, \ldots, k\}$ it holds that y_i dominates X_i. Then $N_G[y_i]$ contains X_i, which dominates Z_i by definition.

Now suppose that $N_G[y_i]$ dominates Z_i for every $i \in \{1, \ldots, k\}$. We claim that the k-tuple $(D^1, \ldots, D^k)$ where $V(D^i) = N_G[y_i] \cup Z_i$ for every $i \in \{1, \ldots, k\}$ is a solution. First note that each D^i is connected (as $N_G[y_i]$ dominates Z_i) and contains Z_i. Hence, it remains to show that $D^1, \ldots, D^k$ are mutually induced.

For a contradiction, let $u_i \in V(D^i)$ and $u_j \in V(D^j)$ be adjacent for some $i \neq j$. Due to (i)–(iv), we find that u_i belongs to $N_G[y_i] \setminus \{x_i\}$ and u_j belongs to $N_G[y_j] \setminus \{x_j\}$. First suppose that u_i is not adjacent to x_i or u_j is not adjacent to x_j, say u_i is not adjacent to x_i. Now, $\{z_i, x_i, y_i, u_i, u_j, y_j\}$ induces a P_6, a contradiction. Hence, u_i must be adjacent to x_i and u_j must be adjacent to x_j. However, now $\{z_i, x_i, u_i, u_j, x_j, z_j\}$ induces a P_6, another contradiction. Hence, we have proven the claim.
We can check in polynomial time whether $N_G[y_i]$ dominates Z_i for every $i \in \{1, \ldots, k\}$. By the above claim, this means that we can check in polynomial time if a certain branch leads to a solution. As the total number of branches is polynomial, the running time of our algorithm is polynomial. The correctness of our algorithm follows from its description; note that we examined all possible situations. □

3.4 Two More Algorithmic Results

In this section we present our final two polynomial-time algorithms. The first result holds for fixed k. The second result is for a smaller graph class but holds even when k and ℓ are both part of the input. To prove the second result, we will use the algorithm of the first result as a subroutine.

Lemma 10 *For every $k \geq 2$ and $s \geq 0$, k-* Induced Disjoint Connected Subgraphs *is polynomial-time solvable for $(sP_1 + 2P_4)$-free graphs.*

Proof Let $s \geq 0$. Let $(G, \mathcal{Z})$ be an instance of k- Induced Disjoint Connected Subgraphs, where G is an $(sP_1 + 2P_4)$-free graph and $\mathcal{Z} = \{Z_1, \ldots, Z_k\}$. By Lemma 3, we may assume that every $Z_i \in \mathcal{Z}$ has size at least 2 and that the union of the sets in $\mathcal{Z}$ is an independent set.

By Lemma 4 we may assume without loss of generality that for a solution $(D^1, \ldots, D^k)$ it holds that every D^i has size at most $(2s + 15)|Z_i|$. Call a set Z_i *small* if $|Z_i| \leq s - 1$, else Z_i is *large*.

If Z_i is small, then D^i has size at most t where $t = (2s + 15)(s - 1)$. Let $\mathcal{Z}'$ be the subset of $\mathcal{Z}$ that contains the small sets of $\mathcal{Z}$. We consider all $O(n^{kt})$ options of choosing the corresponding connected subgraphs D^i. We check in polynomial time if there are no forbidden edges between these subgraphs or between such a subgraph and a large set, and we also check if each such D^i contains Z_i. If one of the conditions is violated, we discard the choice. Otherwise, we delete the vertices of $N[V(D^i)]$ from G for each small Z_i and we also delete the small sets from $\mathcal{Z}$. Note that the resulting graph is still $(sP_1 + 2P_4)$-free. For simplicity, we denote the resulting instance by $(G, \mathcal{Z})$ again. If $|\mathcal{Z}| = k \leq 1$, we solve the instance directly in a trivial way. Keep all created instances with more than one set in $\mathcal{Z}$. Note that we created $O(n^{kt})$ branches (subproblems) in this way and the instance of each branch that we kept has no small sets and $k \geq 2$.

We say that a subgraph D^i in a solution is *easy* if D^i is P_4-free; otherwise D^i is *difficult*. As each Z_i in $\mathcal{Z}$ is now large, each D^i contains an induced sP_1. As G is $(sP_1 + 2P_4)$-free, this means that at least $k - 2$ subgraphs of any solution are easy. We consider all $O(k^2)$ options of choosing the easy subgraphs. By Theorem 4 (take $r = 4$) we find that each easy D^i has a connected dominating set X_i of size at most 2. We consider all $O(n^{2(k-2)})$ options of choosing the vertices of the sets X_i corresponding to the easy subgraphs D^i. Again, we discard a choice if some guessed $D^i = G[X_i \cup Z_i]$ is not connected or a vertex of some guessed D^i is adjacent to a vertex of another guessed D^h or to a vertex of a set Z_j that will be contained in a D^j that is difficult. Otherwise, we obtained, in polynomial time, a new instance after deleting the vertices of $N[V(D^i)]$ from G for each easy D^i and deleting the corresponding sets from $\mathcal{Z}$. For simplicity, we denote the resulting instance by $(G, \mathcal{Z})$ again. Observe that G is still $(sP_1 + 2P_4)$-free, and in particular that $\mathcal{Z}$ has size $k \leq 2$. If $k \leq 1$, the problem has become trivial to solve. Assume that $k = 2$. Then, for simplicity, we write $\mathcal{Z} = \{Z_1, Z_2\}$; note both Z_1 and Z_2 are large, as these sets were large at the start of the initial branch.

So, to summarize, we have created in polynomial time $O(k^2 n^{2(k-2)})$ branches and for each branch we have an instance $(G, \mathcal{Z})$ of 2- Induced Disjoint Connected Subgraphs where G is $(sP_1 + 2P_4)$-free and $\mathcal{Z} = \{Z_1, Z_2\}$. Moreover, if an instance has a solution (D^1, D^2) then both D^1 and D^2 are difficult. It remains to show how we can solve this problem in polynomial time for each created instance $(G, \mathcal{Z})$. We do this below.

Consider a created instance $(G, \mathcal{Z})$. Let (D^1, D^2) be a solution for it. As D^1 is difficult, D^1 has an induced P_4 by definition. As G is $(sP_1 + 2P_4)$-free, this means

that D^2 is $(sP_1 + P_4)$-free. As D^2 is difficult, D^2 has an induced P_4 as well, say on vertices a, b, c, d. As D^2 is $(sP_1 + P_4)$-free and Z_2 is an independent set, the set $\{a, b, c, d\}$ must dominate all but at most $s - 1$ vertices of Z_2. Let Z_2^* be the subset of Z_2 that consists of vertices not adjacent to any vertex of $\{a, b, c, d\}$; so, Z_2^* has size at most $s - 1$.

Fix a vertex u of D^2. As D^2 is $(sP_1 + P_4)$-free and thus P_{2s+3}-free, there exists a path P_i of length at most $2s + 1$ from each $z_i \in Z_2^*$ to u. For the same reason we can also choose a path P_d from d to u of length at most $2s + 3$. Then replacing D^2 by the subgraph F^2 of D^2 that is induced by the union of $\{a, b, c, d, u\} \cup Z_2$ and the inner vertices of all these paths leads to an alternative solution (D^1, F^2), where $F^2 - Z_2$ has size at most t', for $t' = (2s + 2)(s + 1)$.

It now remains to consider all $O(n^{t'})$ options of choosing the vertices of $F^2 - Z_2$. For each option we check if F^2 is connected and contains Z_2 and if $G - N[F^2]$ contains a connected component that contains Z_1. This can be done in polynomial time.

Correctness of our algorithm follows from the above description. As the number of branches is polynomial and each branch can be processed in polynomial time, the running time of our algorithm is polynomial. □

As mentioned, we use the algorithm of Lemma 10 as a subroutine in our next result, which holds even when k and ℓ are part of the input. In addition, we also use the algorithm of of Lemma 8 as a subroutine.

Lemma 11 *For every $s \geq 0$,* Induced Disjoint Connected Subgraphs *is polynomial-time solvable for $(sP_1 + P_3 + P_4)$-free graphs.*

Proof Let $(G, \mathcal{Z})$ be an instance of Induced Disjoint Connected Subgraphs, where G is an $(sP_1 + P_3 + P_4)$-free graph for some integer $s \geq 0$ and $\mathcal{Z} = \{Z_1, \ldots, Z_k\}$. We assume without loss of generality that $|Z_1| \geq |Z_2| \geq \cdots \geq |Z_k|$. By Lemma 3, we may assume that $k \geq 2$; every $Z_i' \in \mathcal{Z}$ has size at least 2; and the union of the sets in $\mathcal{Z}$ is an independent set.

Case 1 $|Z_k| \leq s - 1$.
By Lemma 4 we may assume without loss of generality that for a solution $(D^1, \ldots, D^k)$, we have $|V(D^k)| \leq t$, where $t = (2s + 13)(s - 1)$. We consider all $O(n^t)$ options of choosing the subgraph D^k. For each choice we check in polynomial time if D^k is connected, contains Z_k and that no vertex of D^k is adjacent to any vertex of $Z_1 \cup \ldots \cup Z_{k-1}$. If one of these conditions does not hold, then we discard the choice. Otherwise, we solve Induced Disjoint Connected Subgraphs on instance $(G - N[V(D^k)], \mathcal{Z} \backslash \{Z_k\})$. This can be done in polynomial time. Namely, the graph D^k contains an induced P_3, as Z_k is an independent set of size at least 2. Thus, $G - N[D^k]$ is $(sP_1 + P_4)$-free (and hence, $(sP_1 + P_5)$-free) and we can apply Lemma 8. As the number of branches (subproblems) is polynomial as well, the running time for processing Case 1 is polynomial.

Case 2 $|Z_k| \geq s$ (and hence $|Z_i| \geq s$ for every i).
We need to make a distinction into two more subcases (recall that $k \geq 2$).

Case 2a $k = 2$.
As G is $(sP_1 + P_3 + P_4)$-free, G is also $(sP_1 + 2P_4)$-free and we can use Lemma 10 to process Case 2a in polynomial time.

Case 2b. $k \geq 3$.
Let $(D^1, \ldots, D^k)$ be a solution (assuming it exists). As Z_2 and Z_3 are independent sets of size at least s, both D^2 and D^3 contain an induced sP_1 and an induced P_3. Hence, D^1 is P_4-free. By Theorem 4 (take $r = 4$), we find that D^1 has a connected dominating set X_1 of size at most 2. We consider all $O(n^2)$ options of choosing X_1. We discard a choice if $D^1 = G[X_1 \cup Z_1]$ is not connected or a vertex of D^1 is adjacent to a vertex of $Z_2 \cup \cdots \cup Z_k$. For each non-discarded choice, we solve Induced Disjoint Connected Subgraphs on instance $(G - N[V(D^1)], \mathcal{Z} \setminus \{Z_1\})$. The latter takes polynomial time, as we can apply Lemma 8: as D^1 has an induced P_3, the graph $G - N[V(D^1)]$ is $(sP_1 + P_4)$-free and thus $(sP_1 + P_5)$-free. As the number of branches is polynomial as well, the running time for processing Case 2b is polynomial.
The correctness of our algorithm follows from the case descriptions. As each of the cases can be done in polynomial time, the total running time of our algorithm is polynomial. □

4 NP-Completeness Results

In this section we show a number of NP-completeness results that we need for proving our main theorems. Some of these results hold even for more restricted graph classes. If $\ell = 2$, we write Induced Disjoint Paths instead of Induced Disjoint Connected ℓ- Subgraphs.

4.1 High Girth

The *girth* of a graph G that is not a forest is the length of a shortest cycle of G. We prove two results for graphs of high girth.

Lemma 12 *For every $g \geq 3$,* Induced Disjoint Paths *is NP-complete for the class of graphs of girth at least g.*

Proof We reduce from Disjoint Paths. The reduction of Lynch [21] has the property that the terminals are on disjoint vertices. Hence, after subdividing every edge $\lceil g/3 \rceil \geq 1$ times, the terminals in the new graph, which has girth at least g, are disjoint. Moreover, we now obtained an equivalent instance of Induced Disjoint Paths. □

For some of our other results we prove NP-hardness by reducing from the 2- Disjoint Connected Subgraphs problem. Recall that this problem asks if a given graph has two vertex-disjoint connected subgraphs containing pre-specified sets of vertices Z_1 and Z_2, respectively.

Lemma 13 *For every $g \geq 3$,* 2- Induced Disjoint Connected Subgraphs *is NP-complete for the class of graphs of girth at least g.*

Proof We reduce from 2- DISJOINT CONNECTED SUBGRAPHS, which is known to be NP-complete for graphs of girth at least g for every $g \geq 3$ [17, Lemma 6]. Again, the reduction of [17] subdivides the edges of an instance of 2- DISJOINT CONNECTED SUBGRAPHS on general graphs, and we may assume that it does so at least once. Then we obtain an equivalent instance of 2- INDUCED DISJOINT CONNECTED SUBGRAPHS on a graph of girth at least g. □

4.2 Line Graphs

The *line graph* $L(G)$ of a graph G has vertex set $\{v_e \mid e \in E(G)\}$ and an edge between v_e and v_f if and only if e and f are incident on the same vertex in G.

The following two lemmas show NP-completeness for line graphs. Lemma 14 is due to Fiala et al. [6]. They consider a more general variant of INDUCED DISJOINT PATHS, but their reduction holds in our setting as well. Lemma 15 can be derived from the NP-completeness of 2- DISJOINT CONNECTED SUBGRAPHS [31].

Lemma 14 *([6])* INDUCED DISJOINT PATHS *is NP-complete for the class of line graphs.*

Proof Fiala et al. [6, Theorem 24] prove that INDUCED DISJOINT PATHS is NP-complete for line graphs by reducing from DISJOINT PATHS on general graphs. However, in the paper, they consider a more flexible variant where (among others) terminals can be adjacent. Fortunately, this extra freedom is not used in the reduction. We note that for DISJOINT PATHS, the reduction of Lynch [21] guarantees that the terminals are on disjoint vertices. Then the construction of [6] guarantees that the terminals in the constructed graph form an independent set and hardness for our variant follows. □

Lemma 15 2- INDUCED DISJOINT CONNECTED SUBGRAPHS *is NP-complete for the class of line graphs.*

Proof We reduce from 2- DISJOINT CONNECTED SUBGRAPHS, which is known to be NP-complete [31]. We describe a reduction that is similar to Fiala et al. [6, Theorem 24] for INDUCED DISJOINT PATHS.

Let (G, Z_1, Z_2) be an instance of 2- DISJOINT CONNECTED SUBGRAPHS. For each vertex $z \in Z_1 \cup Z_2$, create a new vertex v_z and connect it by an edge e_z to z. Let G' denote the new graph. Note that (G, Z_1, Z_2) is a yes-instance of 2 - DISJOINT CONNECTED SUBGRAPHS if and only if $(G', \{v_z \mid z \in Z_1\}, \{v_z \mid z \in Z_2\})$ is a yes-instance. Now consider the line graph $L(G')$. For each $z \in Z_1 \cup Z_2$, let $v(e_z)$ be the vertex in $L(G')$ corresponding to e_z. Then it can be readily seen that (G, Z_1, Z_2) is a yes-instance of 2- DISJOINT CONNECTED SUBGRAPHS if and only if $(L(G'), \{v(e_z) \mid z \in Z_1\}, \{v(e_z) \mid z \in Z_2\})$ is a yes-instance of 2- INDUCED DISJOINT CONNECTED SUBGRAPHS. □

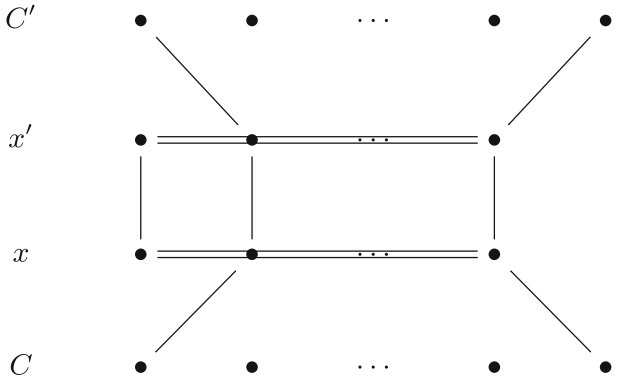

Fig. 1 Connections between cliques in the proof of Lemma 16. The horizontal double lines indicate these vertices are joined in a clique. In the diagram we show variable x_2 appearing in clause C_1 and variable x_n appearing in clause C_m

4.3 Forbidding Some Linear Forest

Finally, we show two lemmas for graphs without certain induced linear forests. Lemma 16 shows that 2- Induced Disjoint Connected Subgraphs is NP-complete for $(3P_2, P_7)$-free graphs. It is readily seen that the gadget constructed in the hardness reduction is not $2P_4$-free. Note that this is in line with Theorem 3. However, Lemma 17 shows that NP-completeness does hold for $2P_4$-free graphs when the number k of terminal sets is part of the input. That is, Induced Disjoint Connected Subgraphs is NP-complete for $2P_4$-free graphs.

Lemma 16 2- Induced Disjoint Connected Subgraphs *is NP-complete for the class of* $(3P_2, P_7)$*-free graphs.*

Proof We reduce from Monotone Not- All- Equal- 3- Sat, which is known to be NP-complete [29]. Let $(\mathcal{X}, \mathcal{C})$ be an instance of Monotone Not- All- Equal- 3- Sat containing n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$ containing only positive literals. The question is whether there exists a truth assignment for $(\mathcal{X}, \mathcal{C})$ such that each C_j contains at least one true variable and at least one false variable.

We construct a graph G as follows. Let X be a clique of size n on vertices $v_1, \ldots, v_n$. Introduce a copy v_i' of each v_i in X. Call the new set X' and make it a clique. Add the edges $v_i v_i'$ for each v_i in X. Let C be an independent set of size m on vertices $c_1, \ldots, c_m$. Introduce a copy c_j' of each vertex c_j in C. Call the new set C' (and keep it an independent set). Now for all $1 \leq i \leq n$ and $1 \leq j \leq m$, add an edge $v_i c_j$ and an edge $v_i' c_j'$ if clause C_j contains variable x_i. Set $Z_1 = C$ and $Z_2 = C'$. Then, (G, Z_1, Z_2) is an instance of 2 - Induced Disjoint Connected Subgraphs. See Fig. 1.

Observe that G is P_7-free. Indeed, let P be any longest induced path in G. Then P can contain at most two vertices from X and at most two vertices from X'. If P contains at most one vertex from C and at most one vertex from C', then P has length at most $2 + 2 + 1 + 1 = 6$. On the other hand, if P contains two vertices from C or two vertices from C', then P has length at most 3.

We also observe that G is $3P_2$-free, as any P_2 must contain at least one vertex from X or from X', and X and X' are cliques. So we are done after proving the following claim: $(\mathcal{X}, \mathcal{C})$ is a yes-instance of MONOTONE NOT- ALL- EQUAL- 3- SAT if and only if (G, Z_1, Z_2) is a yes-instance of 2 - INDUCED DISJOINT CONNECTED SUBGRAPHS.

In the forward direction, let τ be a satisfying truth assignment. We put in A every vertex of X for which the corresponding variable is set to true. We put in A' every vertex of X' for which the corresponding variable is set to false. As each clause C_j contains at least one true variable, c_j is adjacent to a vertex in A. Similarly, each clause C_j contains at least one false variable, so each c'_j is adjacent to a vertex in A'. As X and X' are cliques, A and A' are cliques. Hence, $G[C \cup A]$ and $G[C' \cup A']$ are connected.

Now suppose there is an edge between a vertex of $C \cup A$ and a vertex of $C' \cup A'$. Then, by construction, this edge must be equal to some $v_i v'_i$, which means that v_i is in A and v'_i is in A', so x_i must be true and false at the same time, a contradiction. Hence, there exists no edge between a vertex from $C \cup A$ and a vertex from $C' \cup A'$. We conclude that $(C \cup A, C' \cup A')$ is a solution.

In the backwards direction, let $(C \cup A, C' \cup A')$ be a solution. Then, by definition, there is no edge between $C \cup A$ and $C' \cup A'$, which means that there is no edge between A and A'. Then $A \subseteq X$ and $A' \subseteq X'$, since X and X' are cliques and A (A') needs to contain at least one vertex of X (X'). Also, there is no variable x_i such that v_i is in A and v'_i is in A'. This means we can define a truth assignment τ by setting all variables corresponding to vertices in A to be true, all variables corresponding to vertices in A' to be false, and all remaining vertices in $\mathcal{X}$ to be true (or false, it does not matter).

As C is an independent set and $C \cup A$ is connected, each c_j has a neighbour in A. So each C_j contains a true literal. As C' is an independent set and $C' \cup A'$ is connected, each c'_j has a neighbour in A'. So each C_j contains a false literal. Hence, τ is a satisfying truth assignment. This completes the proof. □

Lemma 17 INDUCED DISJOINT CONNECTED SUBGRAPHS *is NP-complete for the class of* $2P_4$*-free graphs.*

Proof We reduce from MONOTONE 3- SATISFIABILITY [20]. Let Φ be an instance of MONOTONE 3- SATISFIABILITY with n variables $v_1, \ldots, v_n$, l clauses with only positive literals $P_1, \ldots, P_l$ and m clauses with only negated literals $N_1, \ldots, N_m$. We define an instance $(G, Z_1, \ldots Z_{m+1})$ of INDUCED DISJOINT CONNECTED SUBGRAPHS as follows (see also Fig. 2).

- Add a clique, with one vertex v_i for each variable.
- Add an independent set consisting of one vertex p_i for each positive clause together with one further vertex x adjacent to every variable vertex.
- Add an edge between each vertex p_i and the variable vertices contained in the corresponding clause P_i.
- For each negative clause add a complete bipartite graph $K_{2,3}$ where the vertices contained in the part of size 3, $n_{i,1}, n_{i,2}$ and $n_{i,3}$, represent the literals of the clause N_i, whilst the vertices contained in the part of size 2 are denoted as $n_{i,4}$ and $n_{i,5}$.
- Add edges from each literal vertex $n_{i,j}$, $1 \leq j \leq 3$ to the corresponding variable vertex.

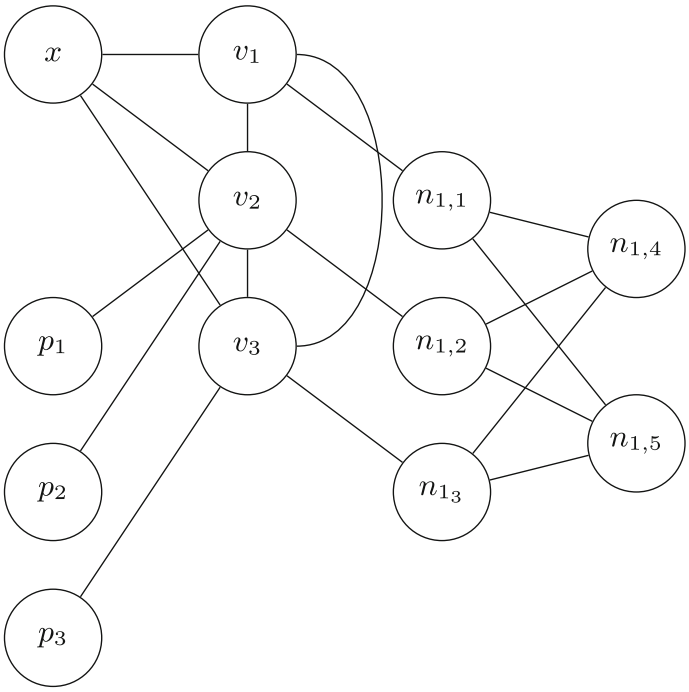

Fig. 2 The graph constructed in the proof of Lemma 17 corresponding to an instance of MONOTONE 3- SATISFIABILITY with three positive clauses and one negative clause $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

- Let Z_1 consist of each positive clause vertex p_i together with the vertex x.
- Let Z_i consist of the two vertices $n_{i-1,4}$ and $n_{i-1,5}$ for $2 \leq i \leq m+1$.

We first show that G is $2P_4$-free. Note that at most one of the two paths in an induced $2P_4$ contains any variable vertex. Since every neighbour of the vertices $\{p_1 \ldots p_l, x\}$ is a variable vertex, none of these vertices is contained in an induced P_4 which excludes variable vertices. As the complete bipartite graph $K_{2,3}$ is P_4-free, this implies that G is $2P_4$-free.

Next we show that G is a yes-instance of INDUCED DISJOINT CONNECTED SUBGRAPHS if and only if Φ is a yes-instance of MONOTONE 3- SATISFIABILITY. Given a satisfying assignment of Φ, let $S_1 = Z_1 \cup T$ where T is the set of variable vertices corresponding to true variables. For $2 \leq i \leq m+1$, let $S_i = Z_i \cup F_i$ where F_i is the set of literal vertices $n_{i-1,j}$ adjacent to variable vertices appearing in N_{i-1} which are assigned to be false. Note that no subgraph $S_i, 2 \leq i \leq m+1$ contains a variable vertex. S_1 is connected since at least one variable appearing in each positive clause must be true in any satisfying assignment. Similarly each S_i is connected for $2 \leq i \leq m+1$ since any negative clause must contain at least one variable which is assigned to be false. Any edge between S_i and S_j for $i \neq j$ must contain a variable vertex since the remaining edges are those contained in copies of $K_{2,3}$ and hence either have two endpoints in the same subgraph S_i or one endpoint contained in no subgraph S_i. Therefore we may assume that $i = 1$. If a variable vertex v_i is contained in S_1 and

has a neighbour in a second subgraph S_j it must be both true and false in a satisfying assignment, a contradiction.

If $(G, Z_1 \dots Z_{m+1})$ is a yes-instance of Induced Disjoint Connected Subgraphs, consider any solution $(S_1, \dots, S_{m+1})$ such that $Z_i \subseteq S_i$ for $1 \leq i \leq m+1$. Note that $\{n_{i-1,1}, n_{i-1,2}, n_{i-1,3} \mid 2 \leq i \leq m+1\} \subseteq N(Z_2 \cup \dots \cup Z_{m+1})$, and thus $S_1 \subseteq Z_1 \cup \{v_1, \dots, v_n\}$. Since the variable vertices form a clique and S_1 will need to contain at least one variable vertex, $(S_2 \cup \dots \cup S_{m+1}) \cap \{v_1, \dots, v_n\} = \emptyset$. Set each variable whose corresponding vertex is contained in S_1 to true and each remaining variable to false. We claim this yields a satisfying assignment for Φ. For a positive clause P_i, note that S_1 must connect p_i to x, which is only possible if a variable vertex adjacent to p_i is in S_1. This variable is contained in the clause and set to true, and will thus satisfy the clause. For a negative clause N_{i-1} with $2 \leq i \leq m+1$, we note that $Z_i = \{n_{i-1,4}, n_{i-1,5}\}$ is connected by S_i, which is only possible if one of $n_{i-1,1}, n_{i-1,2}, n_{i-1,3}$ is in S_i, say $n_{i-1,1}$. But then the variable vertex corresponding to the first literal of the clause cannot be in S_1, and thus is set to false and satisfies the clause. □

5 The Proofs of Theorems 1–3

We are now ready to prove Theorems 1–3, which we restate below.

Theorem *1* (restated). *Let $\ell \geq 2$. For a graph H,* Induced Disjoint Connected ℓ-Subgraphs *on H-free graphs is polynomial-time solvable if $H \subseteq_i sP_3 + P_6$ for some $s \geq 0$; NP-complete if H is not a linear forest; and quasipolynomial-time solvable otherwise.*

Proof If H contains a cycle C_s, then we use Lemma 12 by setting the girth to $g = s+1$. Lemma 12 holds only for $\ell = 2$. If $\ell \geq 3$, we add $\ell - 2$ new terminal vertices to some set Z_i with $|Z_i| = 2$ and make them all adjacent to exactly one vertex that is from Z_i.

Suppose that H contains no cycle, that is, H is a forest. If H contains a vertex of degree at least 3, then we use Lemma 14, as in that case the class of H-free graphs contains the class of $K_{1,3}$-free graphs, which in turn contains the class of line graphs. Lemma 14 is for line graphs but holds only for $\ell = 2$. If $\ell \geq 3$, we add a clique of $\ell - 2$ new terminal vertices to some set Z_i with $|Z_i| = 2$ and make them adjacent to exactly one vertex $u \in Z_i$ and to all neighbours of u. The resulting graph is still a line graph.

In the remaining cases, H is a linear forest. If $H \subseteq_i sP_3 + P_6$ for some $s \geq 0$ we use Lemma 5. Else we use Lemma 6. □

Theorem *2* (restated). *For a graph H such that $H \neq sP_1 + P_6$ for some $s \geq 0$,* Induced Disjoint Connected Subgraphs *on H-free graphs is polynomial-time solvable for H-free graphs if $H \subseteq_i sP_1 + P_3 + P_4$ or $H \subseteq_i sP_1 + P_5$ for some $s \geq 0$, and it is NP-complete otherwise.*

Proof If H is not a linear forest, we use Theorem 1. Suppose H is a linear forest. If $H \subseteq_i sP_1 + P_5$ for some $s \geq 0$ we use Lemma 8. If $H \subseteq_i sP_1 + P_3 + P_4$ for some

$s \geq 0$ we use Lemma 11. If $3P_2 \subseteq_i H$ or $P_7 \subseteq_i H$ we use Lemma 16. Otherwise $2P_4 \subseteq_i H$ and we use Lemma 17. □

Theorem *3* (restated). *Let $k \geq 2$. For a graph H, k-* Induced Disjoint Connected Subgraphs *on H-free graphs is polynomial-time solvable for H-free graphs if $H \subseteq_i sP_1 + 2P_4$ or $H \subseteq_i sP_1 + P_6$ for some $s \geq 0$, and it is NP-complete otherwise.*

Proof If H contains a cycle C_s, then we use Lemma 13 by setting the girth to $g = s+1$. Suppose that H contains no cycle, that is, H is a forest. If H contains a vertex of degree at least 3, then we use Lemma 15, as in that case the class of H-free graphs contains the class of $K_{1,3}$-free graphs, which in turn contains the class of line graphs. In the remaining cases, H is a linear forest. If $H \subseteq_i sP_1 + P_6$ for some $s \geq 0$ we use Lemma 9. If $H \subseteq_i sP_1 + 2P_4$ for some $s \geq 0$ we use Lemma 10. Otherwise we have that $3P_2 \subseteq_i H$ or $P_7 \subseteq_i H$ and we use Lemma 16. □

6 A Slight Problem Generalization

In this section we consider a more general variant of the problem. So far, we required that the terminals must all form an independent set. This condition has been relaxed in some papers in the literature, such as [19] (see also Sect. 1). Given a graph G, we say that *vertex-disjoint* paths $P^1, \ldots, P^k$, for some integer $k \geq 1$, with set R of endpoints are *flexibly mutually induced paths* of G if there exists a set $S \subseteq V \setminus R$ such that $G[S \cup R] = (P^1 + \ldots + P^k) \cup G[R]$. So, there is no edge between two vertices from different paths P^i and P^j except possibly between the endpoints of the paths. We can now define the following decision problem:

Flexibly Induced Disjoint Paths	
Instance:	a graph G and terminal pair collection $T = \{(s_1, t_1) \ldots, (s_k, t_k)\}$.
Question:	does G have a set of flexibly mutually induced paths $P^1, \ldots, P^k$ such that P^i is an s_i-t_i path for $i \in \{1, \ldots, k\}$?

Requiring terminals to form an independent set is crucial for our quasipolynomial results. Namely, Theorem 1 is unlikely to hold in the relaxed setting, as shown below.

Theorem 7 *The* Flexibly Induced Disjoint Paths *problem is NP-complete for the class of P_{14}-free graphs.*

Proof We reduce from 3- Satisfiability, which is well known to be NP-hard. Let Φ be an instance of 3- Satisfiability with n variables $v_1, \ldots, v_n$ and p clauses $c_1, \ldots, c_p$. We may assume that each variable occurs at most once in each clause. Create a set L of $n + p$ vertices, denoted $l_1^v, \ldots, l_n^v, l_1^c, \ldots, l_p^c$, and a set R of $n + p$ vertices, denoted $r_1^v, \ldots, r_n^v, r_1^c, \ldots, r_p^c$. We make L into a clique and R into a clique. For each variable v_i, create two vertices m_i^t and m_i^f, which we both make adjacent to l_i^v and r_i^v. For each clause c_j and each literal ℓ of c_j, create a new vertex m_j^ℓ, which we make adjacent to both l_j^c and r_j^c. If ℓ is the negation of variable v_i, make m_j^ℓ

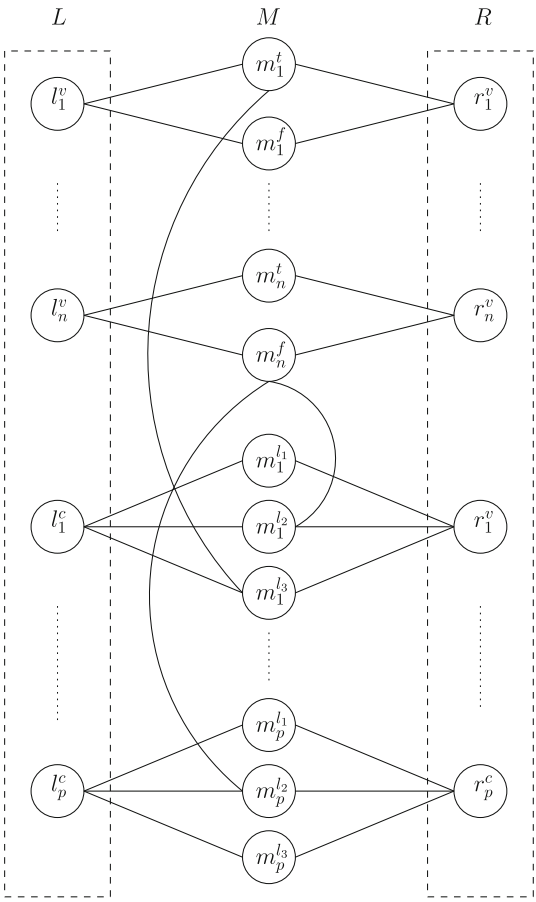

Fig. 3 The hardness construction for FLEXIBLY INDUCED DISJOINT PATHS on P_{14}-free graphs. The sets L and R are cliques; the corresponding edges are not drawn. The three curved edges between m-vertices correspond to an instance of 3- SATISFIABILITY where variable v_1 occurs negatively in c_1 (as the third literal), variable v_n occurs positively in c_1 (as the second literal) and positively in c_p (as the second literal)

adjacent to m_i^t; if ℓ is variable v_i, make m_j^ℓ adjacent to m_i^f. Call M the set of these m-vertices for the variables and the literals. Call the resulting graph G. Let the terminal pair collection $T = \{(l_1^v, r_1^v), \ldots, (l_n^v, r_n^v), (l_1^c, r_1^c), \ldots, (l_p^c, r_p^c)\}$. The construction is illustrated in Fig. 3. We claim (G, T) is a yes-instance if and only if Φ is satisfiable.

First suppose that (G, T) is a yes-instance. Let $P_1^v, \ldots, P_n^v, P_1^c, \ldots, P_p^c$ be a solution for the paths between $(l_1^v, r_1^v), \ldots, (l_n^v, r_n^v), (l_1^c, r_1^c), \ldots, (l_p^c, r_p^c)$ respectively. For $1 \leq i \leq n$, since terminal l_i^v is adjacent to other terminals and to m_i^t and m_i^f, we know that P_i^v contains one of m_i^t and m_i^f immediately after l_i^v. Since m_i^t and m_i^f are adjacent to r_i^v, we may assume without loss of generality that P_i^v then continues directly to r_i^v. We create a truth assignment σ where we set v_i to true if and only if P_i^v contains m_i^t. Similarly, we can argue that P_j^c goes from l_j^c to a vertex m_j^ℓ for some literal ℓ in c_j, and then continues to r_j^c. If ℓ is the negation of v_i, then m_j^ℓ is adjacent to m_i^t. Hence, m_i^t is not in P_i^v and thus the clause is satisfied by σ. Otherwise, if ℓ is v_i, then m_j^ℓ is adjacent to m_i^f. Hence, m_i^f is not in P_i^v and thus the clause is satisfied by σ. It follows that each clause is satisfied by σ and thus Φ is satisfiable.

Now suppose that Φ is satisfiable. Let σ be a truth assignment that satisfies every clause of Φ. For each variable i, let P_i^v be the path from l_i^v to r_i^v that goes via m_i^t if $\sigma(i)$ is set to true and goes via m_i^f otherwise. For each clause $1 \leq j \leq p$, let P_j^c be the path from l_i^c to r_i^c that goes via m_j^ℓ, where ℓ is any literal in c_j that is satisfied by σ. Observe that when ℓ is satisfied, then if ℓ is the negation of v_i, then m_j^ℓ is adjacent to m_i^t but m_i^t is not in P_i^v. Similarly, if ℓ is v_i, then m_j^ℓ is adjacent to m_i^f but m_i^f is not in P_i^v. It follows that $P_1^v, \ldots, P_n^v, P_1^c, \ldots, P_p^c$ is a set of flexibly mutually induced paths. Hence, (G, T) is a yes-instance.

It remains to argue that G is P_{14}-free. Consider a longest induced path P in G. Since both L and R are cliques, P contains at most two vertices of L and at most two vertices of R, and if P contains two vertices of L (or R), then these must appear consecutively. We also note that the vertices in M corresponding to literals have degree 3, and thus when P contains such a vertex m', the next or previous vertex on P must be in L or R, or m' is an endpoint of P. The vertices in M corresponding to variables can have large degree; however, when P contains such a vertex m'', the next or previous vertex on P must be in L or R or must be a vertex in M corresponding to a literal, or P has length 0. Hence, at most three vertices in M can lie consecutively on P before (or after) a vertex of L or R must appear or an endpoint of P is reached: the m-vertex for a literal, a variable, and a literal consecutively. Therefore, in the worst case, P contains three vertices of M, followed by two of L or R, followed by three of M, followed by two of L or R, followed by three of M. Hence, P has at most 13 vertices and thus, G is P_{14}-free. □

7 Future Work

We proved a number of new complexity results on induced paths and subgraphs connecting terminals. These results naturally lead to some open problems. First of all, can we find polynomial-time algorithms for the quasipolynomial cases in Theorem 1? This is a challenging task that is also open for INDEPENDENT SET; note that we reduce to the latter problem in our proof for the case where $H = sP_1 + P_6$ for some $s \geq 0$. Interesting open cases are when $H \in \{2P_4, P_7\}$.

We also recall that the case $H = P_6$ is essentially the only remaining open case left in Theorem 2, which is for the setting where k and ℓ are both part of the input. As shown in Theorems 1 and 3, respectively, we have a positive answer for the settings where ℓ is fixed (and k is part of the input) and where k is fixed (and ℓ is part of the input), respectively. However, it seems challenging to combine the techniques used for proving these results for $H = P_6$ when both k and ℓ are part of the input.

We did not yet discuss the k- INDUCED DISJOINT CONNECTED ℓ- SUBGRAPHS problem, which is the variant where both k and ℓ are fixed; note that if $\ell = 2$, then we obtain the k- INDUCED DISJOINT PATHS problem. The latter problem restricted to $k = 2$ is closely related to the problem of deciding if a graph contains a cycle passing through two specified vertices and has been studied for hereditary graph classes as well; see [19]. Recently, we made some more progress on k-INDUCED DISJOINT PATHS, as we discuss below.

A *subdivided claw* is obtained from a claw after subdividing each edge zero or more times. In particular, the *chair* is the graph obtained from the claw by subdividing one of its edges exactly once. The set $\mathcal{S}$ consists of all graphs with the property that each of their connected components is either a path or a subdivided claw. We proved in [23] that for every integer $k \geq 2$ and graph H, k-Induced Disjoint Paths is polynomial-time solvable if H is a subgraph of the disjoint union of a linear forest and a chair, and it is NP-complete if H is not in $\mathcal{S}$.

From the above it follows in particular that k-Induced Disjoint Paths is polynomial-time solvable for claw-free graphs (just like Independent Set [24, 30]). This is in contrast to the three problems in this paper, which are NP-complete for claw-free graphs (see Theorems 1–3). We leave completing the classification of k- Induced Disjoint Paths as future work and refer to [23] for a more in-depth discussion.

We also leave classifying Flexibly Induced Disjoint Paths to future research; recall that Theorem 1 is unlikely to hold for this problem.

Acknowledgements The authors thank Paweł Rzążewski for the argument using blob graphs, which simplified two of our proofs and led to the case $H = P_6$ in Theorem 1. The paper did not receive support from external funding agencies.

Author Contributions All authors contributed to the paper.

Data Availibility Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

References

1. Belmonte, R., Golovach, P.A., Heggernes, P., van't Hof, P., Kaminski, M., Paulusma, D.: Detecting fixed patterns in chordal graphs in polynomial time. Algorithmica **69**, 501–521 (2014)
2. Bienstock, D.: On the complexity of testing for odd holes and induced odd paths. Discrete Math. **90**, 85–92 (1991)
3. Brandstädt, A., Hoàng, C.T.: On clique separators, nearly chordal graphs, and the maximum weight stable set problem. Theoret. Comput. Sci. **389**, 295–306 (2007)
4. Camby, E., Schaudt, O.: A new characterization of P_k-free graphs. Algorithmica **75**, 205–217 (2016)
5. Fellows, M.R.: The Robertson-Seymour theorems: a survey of applications. Proc. AMS-IMS-SIAM Jt. Summer Res. Conf. Contemp. Math. **89**, 1–18 (1989)
6. Fiala, J., Kamiński, M., Lidický, B., Paulusma, D.: The k-in-a-Path problem for claw-free graphs. Algorithmica **62**, 499–519 (2012)

7. Gartland, P., Lokshtanov, D.: Independent set on P_k-free graphs in quasi-polynomial time. Proc. FOCS **2020**, 613–624 (2020)
8. Gartland, P., Lokshtanov, D., Pilipczuk, M., Pilipczuk, M., Rzążewski, P.: Finding large induced sparse subgraphs in $C_{>t}$-free graphs in quasipolynomial time. Proc. STOC **2021**, 330–341 (2021)
9. Golovach, P.A., Paulusma, D., van Leeuwen, E.J.: Induced disjoint paths in claw-free graphs. SIAM J. Discrete Math. **29**, 348–375 (2015)
10. Golovach, P.A., Paulusma, D., van Leeuwen, E.J.: Induced disjoint paths in circular-arc graphs in linear time. Theor. Comput. Sci. **640**, 70–83 (2016)
11. Golovach, P.A., Paulusma, D., van Leeuwen, E.J.: Induced disjoint paths in AT-free graphs. J. Comput. Syst. Sci. **124**, 170–191 (2022)
12. Grzesik, A., Klimosová, T., Pilipczuk, M., Pilipczuk, M.: Polynomial-time algorithm for maximum weight independent set on P_6-free graphs. ACM Trans. Algorithms **18**, 4:1-4:57 (2022)
13. Jaffke, L., Kwon, O., Telle, J.A.: Mim-width I. Induced path problems. Discrete Appl. Math. **278**, 153–168 (2020)
14. Johnson, M., Paesani, G., Paulusma, D.: Connected vertex cover for $(sP_1 + P_5)$-free graphs. Algorithmica **82**, 20–40 (2020)
15. Kawarabayashi, K., Kobayashi, Y.: A linear time algorithm for the induced disjoint paths problem in planar graphs. J. Comput. Syst. Sci. **78**, 670–680 (2012)
16. Kern, W., Paulusma, D.: Contracting to a longest path in H-free graphs. Proc. ISAAC 2020 LIPIcs **181**, 22:1-22:18 (2020)
17. Kern, W., Martin, B., Paulusma, D., Smith, S., van Leeuwen, E.J.: Disjoint paths and connected subgraphs for H-free graphs. Theor. Comput. Sci. **898**, 59–68 (2022)
18. Kobayashi, Y., Kawarabayashi, K.: Algorithms for finding an induced cycle in planar graphs and bounded genus graphs. Proc. SODA **2009**, 1146–1155 (2009)
19. Lévêque, B., Lin, D.Y., Maffray, F., Trotignon, N.: Detecting induced subgraphs. Discrete Appl. Math. **157**, 3540–3551 (2009)
20. Li, W.N.: Two-segmented channel routing is strong NP-complete. Discrete Appl. Math. **78**, 291–298 (1997)
21. Lynch, J.: The equivalence of theorem proving and the interconnection problem. SIGDA Newsletter **5**, 31–36 (1975)
22. Martin, B., Paulusma, D., Smith, S., van Leeuwen, E.J.: Induced disjoint paths and connected subgraphs for H-free graphs. Proc. WG 2022 LNCS **13453**, 398–411 (2022)
23. Martin, B., Paulusma, D., Smith, S., van Leeuwen, E.J.: Few induced disjoint paths for H-free graphs. Theor. Comput. Sci. **939**, 182–193 (2023)
24. Minty, G.J.: On maximal independent sets of vertices in claw-free graphs. J. Comb. Theory Ser. B **28**, 284–304 (1980)
25. Paesani, G., Paulusma, D., Rzążewski, P.: Feedback vertex set and even cycle transversal for H-free graphs: finding large block graphs. SIAM J. Discrete Math. **36**, 2453–2472 (2022)
26. Pilipczuk, M., Pilipczuk, M., Rzążewski, P.: Quasi-polynomial-time algorithm for independent set in P_t-free graphs via shrinking the space of induced paths. Proc. SOSA **2021**, 204–209 (2021)
27. Radovanović, M., Trotignon, N., Vušković, K.: The (theta,wheel)-free graphs Part IV: induced paths and cycles. J. Combin. Theory Ser. B **146**, 495–531 (2021)
28. Robertson, N., Seymour, P.D.: Graph minors .XIII. The disjoint paths problem. J. Combin. Theory Ser. B **63**, 65–110 (1995)
29. Schaefer, T.J.: The complexity of satisfiability problems. In: STOC, pp. 216–226 (1978)
30. Shibi, N.: Algorithme de recherche d'un stable de cardinalité maximum dans un graphe sans étoile. Discrete Math. **29**, 53–76 (1980)
31. van't Hof, P., Paulusma, D.: Partitioning graphs into connected parts. Theor. Comput. Sci. **410**, 4834–4843 (2009)
32. van't Hof, P., Paulusma, D.: A new characterization of P_6-free graphs. Discrete Appl. Math. **158**, 731–740 (2010)