The k-in-a-path problem for claw-free graphs ⋆

Jiří Fiala[1], Marcin Kamiński[2], Bernard Lidický[1], and Daniël Paulusma[3]

[1] Charles University, Faculty of Mathematics and Physics,
DIMATIA and Institute for Theoretical Computer Science (ITI)
Malostranské nám. 2/25, 118 00, Prague, Czech Republic[+]
{fiala,bernard}@kam.mff.cuni.cz
[2] Computer Science Department, Université Libre de Bruxelles,
Boulevard du Triomphe CP212, B-1050 Brussels, Belgium
marcin.kaminski@ulb.ac.be
[3] Department of Computer Science, University of Durham,
Science Laboratories, South Road,
Durham DH1 3LE, England[++]
daniel.paulusma@durham.ac.uk

Abstract. The k-IN-A-PATH problem is to test whether a graph contains an induced path spanning k given vertices. This problem is NP-complete in general graphs, already when $k = 3$. We show how to solve it in polynomial time on claw-free graphs, when k is an arbitrary fixed integer not part of the input. As a consequence, also the k-INDUCED DISJOINT PATHS and the k-IN-A-CYCLE problem are solvable in polynomial time on claw-free graphs for any fixed k. The first problem has as input a graph G and k pairs of specified vertices (s_i, t_i) for $i = 1, \ldots, k$ and is to test whether G contain k mutually induced paths P_i such that P_i connects s_i and t_i for $i = 1, \ldots, k$. The second problem is to test whether a graph contains an induced cycle spanning k given vertices. When k is part of the input, we show that all three problems are NP-complete, even for the class of line graphs, which form a subclass of the class of claw-free graphs.

Keywords. induced path, claw-free graph, polynomial time algorithm

1 Introduction

Many interesting graph classes are closed under vertex deletion. Every such class can be characterized by a set of forbidden induced subgraphs. One of the best-known examples is the class of perfect graphs. A little over 40 years after Berge's conjecture, Chudnovsky et al. [3] proved that a graph is perfect if and only if it contains neither an *odd hole* (induced cycle of odd length at least five) nor an *odd antihole* (complement of an odd hole). This motivates the research of detecting induced subgraphs such as paths and cycles, which is the topic of

[+] Supported by the Ministry of Education of the Czech Republic as project 1M0021620808 and GACR 201/09/0197.
[++] Supported by EPSRC (EP/D053633/1 and by the Royal Society (JP090172).
⋆ An extended abstract of this paper will be presented at STACS 2010.

this paper. To be more precise, we specify some vertices of a graph called the *terminals* and study the computational complexity of deciding if a graph has an induced subgraph of a certain type containing all the terminals. In particular, we focus on the following problem.

k-IN-A-PATH
Instance: a graph G with k terminals.
Question: does there exist an induced path of G containing the k terminals?

Note that in the problem above, k is a fixed integer. Clearly, the problem is polynomially solvable for $k = 2$. Haas and Hoffmann [15] consider the case $k = 3$. After pointing out that this case is NP-complete as a consequence of a result by Fellows [11], they prove W[1]-completeness (where they take as parameter the length of an induced path that is a solution for 3-IN-A-PATH). Derhy and Picouleau [8] proved that the case $k = 3$ is NP-complete even for graphs with maximum degree at most three.

A natural question is what will happen if we relax the condition of "being contained in an induced path" to "being contained in an induced tree". This leads to the following problem.

k-IN-A-TREE
Instance: a graph G with k terminals.
Question: does there exist an induced tree of G containing the k terminals?

As we will see, also this problem has received a lot of attention in the last two years. It is NP-complete if k is part of the input [8]. However, Chudnovsky and Seymour [5] have recently given a deep and complicated polynomial time algorithm for the case $k = 3$.

Theorem 1 ([5]) *The* 3-IN-A-TREE *problem is solvable in polynomial time.*

The computational complexity of k-IN-A-TREE for $k = 4$ is still open. So far, only partial results are known, such as a polynomial time algorithm for $k = 4$ when the input is triangle-free by Derhy, Picouleau and Trotignon [9]. This result and Theorem 1 were extended by Trotignon and Wei [25] who showed that k-IN-A-TREE is polynomially solvable for graphs of girth at least k. Derhy, Picouleau and Trotignon [9] show that it is NP-complete to decide if a graph G contains an induced tree T covering four specified vertices such that T has at most one vertex of degree at least three.

In general, k-IN-A-PATH and k-IN-A-TREE are only equivalent for $k \leq 2$. However, in this paper, we study *claw-free* graphs (graphs with no induced 4-vertex star). Claw-free graphs are a rich and well-studied class containing,

e.g., the class of (quasi)-line graphs and the class of complements of triangle-free graphs; see Faudree, Flandrin and Ryjáček [10] for a survey. Note that any induced tree in a claw-free graph is in fact an induced path.

Observation 2 *The k-IN-A-PATH and k-IN-A-TREE problem are equivalent for the class of claw-free graphs.*

Motivation. The polynomial time algorithm for 3-IN-A-TREE [5] has already proven to be a powerful tool for several problems. For instance, it is used as a subroutine in polynomial time algorithms for detecting induced thetas and pyramids [5] and several other induced subgraphs [21]. The authors of [16] use it to solve the PARITY PATH problem in polynomial time for claw-free graphs. (This problem is to test if a graph contains both an odd and even length induced paths between two specified vertices. It is NP-complete in general as shown by Bienstock [1].)

Lévêque et al. [21] use the algorithm of Chudnovsky and Seymour [5] to solve the 2-IN-A-CYCLE problem in polynomial time for graphs not containing an induced path or induced subdivided claw on some fixed number of vertices. The k-IN-A-CYCLE problem is to test if a graph contains an induced cycle spanning k terminals. In general it is NP-complete already for $k = 2$ [1]. For fixed k, an instance of this problem can be reduced to a polynomial number of instances of the k-INDUCED DISJOINT PATHS problem, which we define below. Paths $P_1, \ldots, P_k$ in a graph $G = (V, E)$ are said to be *mutually induced* if for any $1 \leq i < j \leq k$, P_i and P_j have neither common vertices, i.e., $V(P_i) \cap V(P_j) = \emptyset$ nor adjacent vertices, i.e., $uv \notin E$ for any $u \in V(P_i), v \in V(P_j)$.

k-INDUCED DISJOINT PATHS
Instance: a graph G with k pairs of terminals (s_i, t_i) for $i = 1, \ldots, k$.
Question: does G contain k mutually induced paths P_i such that P_i connects s_i and t_i for $i = 1, \ldots, k$?

This problem is NP-complete for $k = 2$ [1]. Kawarabayashi and Kobayashi [19] showed that, for any fixed k, the k-INDUCED DISJOINT PATHS problem is solvable in linear time on planar graphs and that consequently k-IN-A-CYCLE is solvable in polynomial time on this graph class for any fixed k. The same authors [20] improve the latter result by presenting a linear time algorithm for this problem, and they extend the results for both these problems to graphs of bounded genus. As we shall see, we can also solve k-INDUCED DISJOINT PATHS and k-IN-A-CYCLE in polynomial time in claw-free graphs. The version of the problem in which any two paths are vertex-disjoint but may have adjacent vertices is called the k-DISJOINT PATHS problem. For this problem Robertson and Seymour [22] proved the following result.

Theorem 3 ([22]) *For fixed k, the k-*DISJOINT PATHS *problem is solvable in polynomial time.*

Our Results and Paper Organization. In Section 2 we define some basic terminology. Section 3 contains our main result: k-IN-A-PATH is solvable in polynomial time in claw-free graphs for any fixed integer k. This, in fact, follows from a stronger theorem proven in the same section; the problem is solvable in polynomial time even if the terminals are to appear on the path in a fixed order. Our polynomial time algorithm for the latter problem first performs "cleaning of the graph". This is an operation that has been introduced for claw-free graphs in [16]. After cleaning the graph is free of odd antiholes of length at least seven. Next we treat odd holes of length five that are contained in the neighborhood of a vertex. The resulting graph is quasi-line. We then use a recent characterization of quasi-line graphs by Chudnovsky and Seymour [4] and related algorithmic results of King and Reed [18] to solve the problem for quasi-line graphs.

In Section 4 we show a number of consequences of our result. We first prove that both the k-INDUCED DISJOINT PATHS and the k-IN-A-CYCLE problem are polynomially solvable in claw-free graphs for any fixed integer k. We then show how to solve for any fixed graph H the problem of finding a so-called induced realization of H in a graph in polynomial time. This containment notion introduced by Lévêque et al. [21] generalizes the notion of a topological minor and will be explained later.

In Section 5 we prove that the three problems k-IN-A-PATH, k-INDUCED DISJOINT PATHS and k-IN-A-CYCLE are NP-complete, even for the class of line graphs, a subclass of the class of claw-free graphs, when k is part of the input instead of being fixed. Section 6 contains the conclusions. There, we also mention a number of relevant open problems.

2 Preliminaries

All graphs in this paper are undirected, finite, and neither have loops nor multiple edges. Let G be a graph. We refer to the vertex set and edge set of G by $V = V(G)$ and $E = E(G)$, respectively. The *neighborhood* of a vertex u in G is denoted by $N_G(u) = \{v \in V \mid uv \in E\}$. The subgraph of G induced by $U \subseteq V$ is denoted $G[U]$. Analogously, the *neighborhood* of a set $U \subseteq V$ is $N(U) := \bigcup_{u \in U} N(u) \setminus U$. We say that two vertex-disjoint subsets of V are *adjacent* if some of their vertices are adjacent. The *distance* $d(u, v)$ between two vertices u and v in G is the number of edges on a shortest path between them. The *contraction* of an edge $e = uv$ removes its two end vertices u, v and replaces it by a new vertex adjacent to all vertices in $N(u) \cup N(v)$ (without introducing loops or multiple edges).

We denote the path and cycle on n vertices by P_n and C_n, respectively. Let $P = v_1 v_2 \dots v_p$ be a path with a fixed orientation. The successor v_{i+1} of v_i is denoted by v_i^+ and its predecessor v_{i-1} by v_i^-. The segment $v_i v_{i+1} \dots v_j$ is denoted by $v_i \overrightarrow{P} v_j$. The reverse segment $v_j v_{j-1} \dots v_i$ is denoted by $v_j \overleftarrow{P} v_i$.

A *hole* is an induced cycle of length at least 4 and an *antihole* is the complement of a hole. We say that a hole is *odd* if it has an odd number of edges. An antihole is called odd if it is the complement is an odd hole.

A *claw* is the graph $(\{x, a, b, c\}, \{xa, xb, xc\})$, where vertex x is called the *center* of the claw. A graph is *claw-free* if it does not contain a claw as an induced subgraph. A *clique* is a subgraph isomorphic to a complete graph. A *diamond* is a graph obtained from a clique on four vertices after removing one edge. A vertex u in a graph G is *simplicial* if $G[N(u)]$ is a clique.

Let s and t be two specified vertices in a graph $G = (V, E)$. A vertex $v \in V$ is called *irrelevant* for vertices s and t if v does not lie on any induced path from s to t. A graph G is *clean* if none of its vertices is irrelevant. We say that we *clean* G for s and t by repeatedly deleting irrelevant vertices for s and t as long as possible. In general, determining if a vertex is irrelevant is NP-complete [1]. However, for claw-free graphs, the authors of [16] showed the following (where they used Theorem 1 and Observation 2 for obtaining the polynomial time bound).

Lemma 4 ([16]) *Let s, t be two vertices of a claw-free graph G. Then G can be cleaned for s and t in polynomial time. Moreover, the resulting graph contains no odd antihole of length at least seven.*

The *line graph* of a graph G with edges $e_1, \dots, e_p$ is the graph $L = L(G)$ with vertices $u_1, \dots, u_p$ such that there is an edge between any two vertices u_i and u_j if and only if e_i and e_j share an end vertex in H. We note that mutually induced paths in a line graph $L(G)$ are in one-to-one correspondence with vertex-disjoint paths in G. Combining this observation with Theorem 3 leads to the following result.

Corollary 5 *For fixed k, the k-*INDUCED DISJOINT PATHS *problem can be solved in polynomial time in line graphs.*

A graph $G = (V, E)$ is called a *quasi-line graph* if for every vertex $u \in V$ there exist two vertex-disjoint cliques A and B in G such that $N(u) = V(A) \cup V(B)$ (where $V(A)$ and $V(B)$ might be adjacent). Clearly, every line graph is quasi-line and every quasi-line graph is claw-free. The following observation is useful and easy to see by looking at the complement of a neighborhood in a graph.

Observation 6 *A claw-free graph G is a quasi-line graph if and only if G does not contain a vertex that has an odd antihole in its neighborhood.*

A clique in a graph G is called *nontrivial* if it contains at least two vertices. A nontrivial clique A is called *homogeneous* if every vertex in $V(G)\backslash V(A)$ is either adjacent to all vertices of A or to none of them. Note that it is possible to check in polynomial time if an edge of the graph is a homogeneous clique. This justifies the following observation.

Observation 7 *The problem of detecting a homogeneous clique in a graph is solvable in polynomial time.*

Two disjoint cliques A and B form a *homogeneous pair* in G if the following two conditions hold. First, at least one of A, B contains more than one vertex. Second, every vertex $v \in V(G)\backslash(V(A)\cup V(B))$ is either adjacent to all vertices of A or to none vertex of A as well as either adjacent to all of B or to none of B. The following result by King and Reed [18, Section 3] will be useful.

Lemma 8 ([18]) *The problem of detecting a homogeneous pair of cliques in a graph is solvable in polynomial time.*

Let V be a finite set of points of a real line, and $\mathcal{I}$ be a collection of intervals. Two points are adjacent if and only if they belong to a common interval $I \in \mathcal{I}$. The resulting graph is a *linear interval graph*. Analogously, if we consider a set of points of a circle and set of intervals (angles) on the circle we get a *circular interval graph*. Graphs in both classes are claw-free.

As we shall see, linear interval graphs and circular interval graphs are closely related to interval graphs and circular arc graphs, respectively. A graph is an *interval graph* if intervals of the real line can be associated with its vertices such that two vertices are adjacent if and only if their corresponding intervals overlap. An interval graph is *proper interval* if it has an interval representation in which no interval is properly contained in any other interval. The class of *(proper) circular arc graphs* is defined analogously.

Linear interval graphs coincide with proper interval graphs. In order to see this we need two results from the literature. First, let G be a connected graph with maximal cliques $K_1, \ldots, K_p$, and let $\mathcal{K}_v$ denote the set of maximal cliques in G containing vertex $v \in V(G)$. Then G is an interval graph if and only if G has a path decomposition that is a *clique path* [12], i.e., a path $P = K_1 \cdots K_p$ such that for each $v \in V(G)$ the set $\mathcal{K}_v$ induces a connected subpath in P. Second, an interval graph is a proper interval graph if and only if it is claw-free [23]. As we shall see, the above two results enable us to show that a graph is a linear interval graph if and only if it is a proper interval graph.

Suppose G is a linear interval graph resulting from a set of points V and a collection of intervals $\mathcal{I}$. Then the maximal cliques of G are in 1-to-1 correspondence with the intervals in I. Consequently, they form a clique path. Then, by the result of Fulkerson and Gross [12], G is an interval graph. Because G is claw-free, we use the result of Roberts [23] to find that G is proper interval.

To prove the reverse implication, suppose G is a proper interval graph. Then we associate an interval with each maximal clique in G. Because the maximal cliques of G form a clique path, this results in a desired collection of intervals $\mathcal{I}$.

The result for circular interval graphs and proper circular arc graphs can be proven analogously. Due to these two equivalences, we can make use of the following result of Deng, Hell, and Huang [7] originally proven for proper interval graphs and proper circular arc graphs.

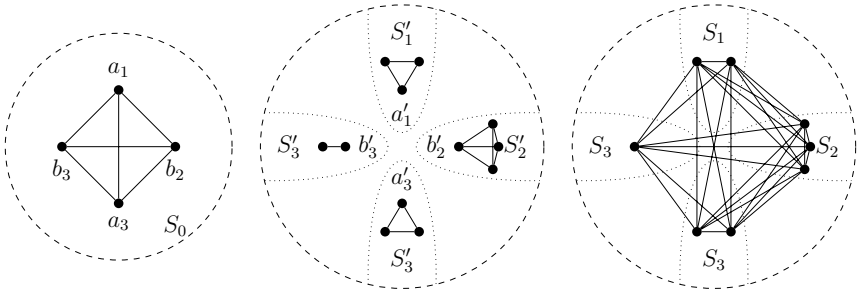

Fig. 1. Composition of three linear interval strips (only part of the graph is displayed).

Theorem 9 ([7]) *Circular interval graphs and linear interval graphs can be recognized in linear time. Furthermore, a corresponding representation of such graphs can be constructed in linear time as well.*

A *linear interval strip* (S, a, b) is a linear interval graph S where a and b are the leftmost and the rightmost points (vertices) of its representation. Observe that in such a graph the vertices a and b are simplicial. Let S_0 be a graph with vertices $a_1, b_1, \ldots, a_m, b_m$ that is isomorphic to an arbitrary disjoint union of complete graphs. Let $(S'_1, a'_1, b'_1), \ldots, (S'_m, a'_m, b'_m)$ be a collection of linear interval strips. The *composition* S_m is defined inductively where S_i is formed from the disjoint union of S_{i-1} and S'_i, where:

- all neighbors of a_i are connected to all neighbors of a'_i;
- all neighbors of b_i are connected to all neighbors of b'_i;
- vertices a_i, a'_i, b_i, b'_i are removed.

See Figure 1 for an example. We are now ready to state the structure of quasi-line graphs as characterized by Chudnovsky and Seymour [4].

Theorem 10 ([4]) *A connected quasi-line graph G with no homogeneous pair of cliques is either a circular interval graph or a composition of linear interval strips.*

Finally, we need another algorithmic result of King and Reed [18]. They observe that the composition of the final strip in a composition of linear interval graphs is a so-called nontrivial interval 2-join and that every nontrivial interval 2-join contains a so-called canonical interval 2-join. In Lemma 13 of their paper they show how to find in polynomial time a canonical interval 2-join in a quasi-line graph with no homogeneous pairs of cliques and no simplicial vertices, or else to conclude that none exists. Applying this result recursively leads to the following lemma.

Lemma 11 ([18]) *Let G be a quasi-line graph that contains no homogeneous pairs of cliques and no simplicial vertices and that is a composition of linear interval strips. Then the collection of linear interval strips that define G can be found in polynomial time.*

3 Main Result

Now we are ready to state our main result.

Theorem 12 *For any fixed k, the k-IN-A-PATH problem is solvable in polynomial time in claw-free graphs.*

In order to prove Theorem 12 we define the following problem.

ORDERED-k-IN-A-PATH
Instance: a graph G with k terminals ordered as $t_1, \ldots, t_k$.
Question: does there exist an induced path of G starting in t_1 then passing through $t_2, \ldots, t_{k-1}$ and ending in t_k?

We can resolve the original k-IN-A-PATH problem by $k!$ rounds of the more specific version defined above, where in each round we order the terminals by a different permutation. Hence, since we assume that k is fixed, it suffices to prove Theorem 13 in order to obtain Theorem 12.

Theorem 13 *For any fixed k, the* ORDERED-k-IN-A-PATHS *problem is solvable in polynomial time in claw-free graphs.*

We now prove Theorem 13 by presenting a polynomial time algorithm that solves the ORDERED-k-IN-A-PATH problem on a claw-free graph G with terminals in order $t_1, \ldots, t_k$ for any fixed integer k. We call an induced path P from t_1 to t_k that contains the other terminals in order $t_2, \ldots, t_{k-1}$ a *solution* of this problem. Furthermore, an operation in this algorithm on input graph G with terminals $t_1, \ldots, t_k$ *preserves the solution* if the following holds: the resulting graph G' with resulting terminals $t'_1, \ldots, t'_{k'}$ for some $k' \leq k$ is a YES-instance of the ORDERED-k'-IN-A-PATH problem if and only if G is a YES-instance of the ORDERED-k-IN-A-PATH problem. We call G *convenient* (for our purposes) if G has the following three properties:

(i) t_1, t_k are of degree one in G and all other terminals t_i ($1 < i < k$) are of degree two in G, and the two neighbors of such t_i are not adjacent;
(ii) for $1 < i < j < k$, the distance between t_i and t_j is at least four;
(iii) G is connected.

THE ALGORITHM AND PROOF OF THEOREM 13

Let G be a claw-free graph with terminals $t_1, \ldots, t_k$ for some $k \geq 2$.

If $k = 2$, we compute a shortest path from t_1 to t_2. Suppose $k \geq 3$. As we will see, our algorithm might create a set of new graphs that we must process one by one. While doing so, we make the following two implicit assumptions.

A. The number of terminals in the graph under consideration is always at least three; if the number of terminals drops to 2 in a newly created graph, we just try to find the shortest path between the two terminals and do not process the graph any further.

B. The graph under consideration is clean for t_1 and t_k. We can always ensure this by running the polynomial time algorithm of Lemma 4 on top of every operation we perform. If during a cleaning operation we remove a terminal, then we do not process the graph any further. In particular, we note that cleaning keeps properties (i)-(iii) intact.

Step 1. Reduce to a set of convenient graphs.
We apply Lemma 14 and obtain in polynomial time a set $\mathcal{G}$ that consists of a polynomial number of convenient claw-free graphs of size at most $|V(G)|$ such that there is a solution for G if and only if there is a solution for one of the graphs in $\mathcal{G}$. We consider *each* graph in $\mathcal{G}$. For simplicity, we denote such a graph by G as well.

Step 2. Reduce to a quasi-line graph.
Because G is assumed to be clean for t_1 and t_k, we note that G contains no

antihole of length at least seven by Lemma 4. We apply Lemma 15 as long as we can. This lemma removes a vertex, the neighborhood of which contains an odd hole of length five. Because every time we apply Lemma 15 the number of vertices in G decreases by one, this takes $O(|V(G)|^6)$ time in total. Note that G stays connected, because we do not remove any cut-vertices due to the claw-freeness. By property (i), we do not remove a terminal either. Then G has become a convenient quasi-line graph due to Observation 6.

Step 3. Reduce to a circular interval graph or to a composition of linear interval strips.
We first exhaustively search for homogeneous cliques by running the polynomial time algorithm of Observation 7. Each time we find such a clique, we reduce it in polynomial time to a single vertex by applying Lemma 16. By Lemma 16, the resulting graph stays convenient and quasi-line, while we preserve the solution. At some moment it contains no homogeneous cliques anymore. We then exhaustively search for homogeneous pairs by running the polynomial time algorithm of Lemma 8. Each time we find such a pair, we reduce its two cliques in polynomial time to two single vertices by applying Lemma 17. By Lemma 17, the resulting graph stays convenient and quasi-line, while we preserve the solution. At some moment it contains no homogeneous pairs anymore. Then, by Theorem 10, G is either a circular interval graph or a composition of linear interval strips. We use Theorem 9 to recognize in polynomial time in which case we are.

Step 4a. Solve the problem for a circular interval graph.
Let G be a circular interval graph. Observe that the order of vertices in an induced path must respect the natural order of points on the circle. Hence, deleting all points that lie on the circle between t_k and t_1 preserves the solution. So, we may assume that G is a linear interval graph. We solve the problem for these graphs in Theorem 18.

Step 4b. Solve the problem for a composition of linear interval strips.
Let G be a composition of linear interval strips. We replace t_1 and t_k by their unique neighbors and repeat this process until they get degree at least one. Then, because G is assumed to be clean for t_1 and t_k, G contains no simplicial vertex. This means that we can find these strips in polynomial time by Lemma 11. We use this information in Lemma 19. There we create a line graph G' with $|V(G')| \leq |V(G)|$, while preserving the solution. Moreover, this can be done in polynomial time by the same theorem. In Theorem 20 we prove that the problem is polynomially solvable for line graphs.

In order to finish the correctness proof and running time analysis of our algorithm, it remains to state and prove Lemmas 14–17, Theorem 18, Lemma 19 and Theorem 20.

Lemma 14 *Let G be a claw-free graph on n vertices with terminals ordered $t_1, \ldots, t_k$. Then there exists a set $\mathcal{G}$ of $n^{O(k)}$ convenient claw-free graphs, each of size at most $|V(G)|$, such that G has a solution if and only if there exists a graph in $\mathcal{G}$ that has a solution. Moreover, $\mathcal{G}$ can be constructed in polynomial time.*

Proof. Initially set $\mathcal{G} := \emptyset$. We branch as follows. For $i = 1, \ldots, k$, we guess two ordered disjoint vertex subsets R_i and R'_i of G; the vertices of R_i are supposed to be the last $|R_i|$ vertices preceding t_i in a possible solution (path), whereas the vertices of R'_i are supposed to be the first $|R'_i|$ vertices following t_i. We require $|R_1| = |R'_k| = 0$ while all other sets R_i and R'_i must have size three with the following exceptions. First, $|R_i| \leq 2$ is allowed for $i = 2, 3$, but in that case we require $t_1 \in R_i$. Second, $|R'_i| \leq 2$ is allowed for $i = k-2, k-1$, but in that case we require $t_k \in R_i$.

Note that we allow that two sets from $\{R_1, \ldots, R_k\} \cup \{R'_1, \ldots, R'_k\}$ overlap. However, we do check if every component of the subgraph G^* induced by $\bigcup_i R_i \cup \bigcup_j R'_j \cup \{t_1, \ldots, t_k\}$ is a path. If this is not the case then we discard the whole guess and start all over again. Otherwise, we observe that t_1 and t_k are end vertices of path components in G^*, while all other terminals are inner vertices of path components. Next, we consider the path components in G^* containing more than one terminal. If the order of the terminals on such a path component is not $t_i, t_{i+1}, \ldots, t_j$ for some $i < j$ when passing through the path component, we discard the whole guess. Otherwise, we remove $t_{i+1}, \ldots, t_{j-1}$ from the list of terminals. We also remove t_j if $i = 1$ or t_i if $j = k$. If we kept both t_i and t_j then, if necessary, we replace t_i and t_j on this path component such that on the path component (i) they are at distance at least four of each other and (ii) also are of distance at least one of each end vertex of the path component. Because the path component has length at least eight, such a replacement is possible.

We now do the following in the graph G. For every terminal and for every non-terminal vertex in $\bigcup_i R_i \cup \bigcup_j R'_j$ that is not an end vertex of a path component in G^*, we remove all its neighbors from G that are not vertices of G^*. The resulting graph G' is claw-free, because G is claw-free and we only removed vertices. If there are two terminals in different components of G', then we discard G'. If not, then we put the component of G' that contains all the terminals in the set $\mathcal{G}$, because this graph is convenient; we note that property (ii) also holds with respect to $i = 1$ or $j = k$. We then try to extend $\mathcal{G}$ by taking a new guess.

Because k is fixed and the number of guesses is $O(n^{6k})$, this procedure ends in polynomial time with a set $\mathcal{G}$ that satisfies all conditions of the lemma. □

Lemma 15 *Let G be a convenient claw-free graph. Removing a vertex $u \in V(G)$, the neighborhood of which contains an induced odd hole of length five, preserves the solution.*

Proof. Because G is convenient, u is not a terminal. We first show the following claim.

Claim 1. Let $G[\{v, w, x, y\}]$ be a diamond in which vw is a non-edge. If there is a solution P that contains v, x, w, then there is another solution that contains y (and that does not contain x).

In order to see this take the original solution P and note that by claw-freeness any neighbor of y on P must be in the (closed) neighborhood of either v or w. This way the solution can be rerouted via y, without using x. This proves Claim 1.

Now suppose that u is a vertex which has an odd hole C of length five in its neighborhood. Obviously, G is a YES-instance if $G - u$ is a YES-instance. To prove the reverse implication, suppose G is a YES-instance. Let P be a solution. If u does not belong to P then we are done. Hence, we suppose that u belongs to P and consider three cases depending on $|V(C) \cap V(P)|$.

Case 1. $|V(C) \cap V(P)| \geq 2$. Then $|V(C) \cap V(P)| = 2$, as any vertex on P will have at most two neighbors. We are done by Claim 1.

Case 2. $|V(C) \cap V(P)| = 1$. Let $w \in V(C)$ belong to P and let the other neighbor of u that belongs to P be x. We note that x must be adjacent to at least one of the neighbors of w in C due to the claw-freeness of G. Then we can apply Claim 1 again.

Case 3. $|V(C) \cap V(P)| = 0$. Let the two neighbors of u on P be x and y. To avoid a claw at u, every vertex of C must be adjacent to x or y. If there is a vertex in C adjacent to both, we apply Claim 1. Suppose there is no such vertex and that the vertices of the C are partitioned in two sets X (vertices of C only adjacent to x) and Y (vertices of C only adjacent to y). We assume without loss of generality that $|X| = 3$, and hence contains a pair of independent vertices which together with u and y form a claw. This is a contradiction. □

Lemma 16 *Let G be a convenient quasi-line graph with a homogeneous clique A. Then contracting A to a single vertex preserves the solution and results in a convenient quasi-line graph that has the same terminals as G.*

Proof. Each vertex in A lies on a triangle, unless G is isomorphic to P_2, which is not possible. Hence, by property (i), A does not contain a terminal. We remove all vertices of A except one. The resulting graph will be a convenient quasi-line graph containing the same terminals, and we will preserve the solution. □

Lemma 17 *Let G be a convenient quasi-line graph with terminals ordered $t_1, \ldots, t_k$ that has no homogeneous clique but that has a homogeneous pair (A, B). If G is clean for t_1 and t_k, then contracting A and B to single vertices preserves the solution and results in a convenient quasi-line graph.*

Proof. We assume that G is clean for t_1 and t_k. Because G does not contain a homogeneous clique, $V(A)$ and $V(B)$ must be adjacent. In all the cases discussed below we will actually not contract edges but only remove vertices from A and B. Hence, the resulting graph will always be a quasi-line graph.

Case 1. A contains t_1 or t_k.
By property (i), t_1 and t_k are of degree one. Then A cannot contain t_1 and t_k. We assume without loss of generality that A contains t_1.

First suppose $|V(A)| = 1$, so A only contains t_1. Then the (unique) neighbor of t_1 is in B. Because G is clean and $|V(B)| \geq 2$, we find that G is isomorphic to $P_3 = t_1t_2t_3$ and $V(G) = \{t_1, t_2, t_3\}$. Hence, the statement of the lemma holds if we remove t_3 and set $k = 2$.

Now suppose $|V(A)| \geq 2$. Because t_1 is of degree one, A consists of two vertices, namely t_1 and its neighbor t_1'. Note that t_1' has no neighbor outside A and B, because t_1 is of degree one. Suppose t_k is in B. Then $|V(B)| \leq 2$, because t_k is of degree one. If $V(B) = \{t_k\}$ then we return to the previous case. Suppose $|V(B)| = 2$. Because G is clean for t_1 and t_k, we find that G is isomorphic to a 4-vertex path starting in t_1 and ending in t_k. By property (ii), G can pass through at most one other terminal, so $k = 3$. Hence, the statement of the lemma holds if we remove t_1 and t_3 and set $k = 2$.

Suppose t_k is not in B. Because G is clean, t_1' is adjacent to all vertices in B. This means that $|V(B)| = 1$, as otherwise B is a homogeneous clique. Then deleting t_1 and replacing it by t_1' in the set of terminals results in a convenient graph and preserves the solution.

Case 2. A contains a terminal t_i for some $2 \leq i \leq k-1$.
We may assume that neither t_1 nor t_k is in $A \cup B$, as otherwise we return to Case 1. Then, by property (ii), t_i is the only terminal in $A \cup B$.

First suppose $|V(A)| = 1$, so A only contains t_i. Because $V(A)$ and $V(B)$ are adjacent, t_i is adjacent to a vertex u in B. By property (i), u is the only vertex in B adjacent to t_i. We delete all vertices of B except u. The resulting graph is convenient, and we preserve the solution.

Now suppose $|V(A)| \geq 2$. By property (ii), A contains only one other vertex t_i' and t_i, t_i' do not have a common neighbor. Then A must be separated of the rest of the graph by B. Because B is a clique, this means that every induced path from t_1 to t_k does not contain t_i (and t_i'). Hence, this case is not possible, because G is clean for t_1 and t_k.

Case 3. A contains no terminal.
Then we may assume that B contains no terminal either, as otherwise we return to a previous case by symmetry. Let $a'b' \in E(G)$ with $a' \in V(A)$ and $b' \in V(B)$. Let G' be the graph obtained from G by removing all vertices of $A \cup B$ except a' and b'. Note that we have kept all terminals and that the resulting graph is convenient. Any solution P' for G' is a solution for G.

Now assume we have a solution P for G. We claim that $|P \cap A| \leq 1$ and $|P \cap B| \leq 1$. Suppose otherwise, say $|P \cap A| \geq 2$. Then $|P \cap A| = 2$, because P is an indued path and A is a clique. Since t_1 and t_k are not in A, we find that P contains a subpath $xuvy$ with $u, v \in A$. Since x is adjacent to $u \in A$, but non-adjacent to $v \in A$, we find that $x \in B$. Analogously we get that $y \in B$. However, then $xy \in E(G)$. This is a contradiction.

Suppose $|P \cap A| = 0$ and $|P \cap B| = 0$. Then P is a solution for G' as well. Suppose $|P \cap A| = 0$ and $|P \cap B| = 1$. Then we may without loss of generality assume that $b' \in V(P)$. We find that P is a solution for G' as well. The case $|P \cap A| = 1$ and $|P \cap B| = 0$ follows by symmetry. Suppose $|P \cap A| = |P \cap B| = 1$, say P intersects A in a and B in b. If $ab \in E(G)$ then we replace ab by $a'b'$ and obtain a solution for G'. Suppose $ab \notin E(G)$. Because a is not a terminal, a has neighbors x and y on P. If $x, y \notin N(b)$ then $\{a', x, y, b'\}$ induces a claw in G with center a'. This is not possible. Hence, we may assume without loss of generality that y is adjacent to b. Since A or B contains at least two vertices, y has degree at least three. Then y is not a terminal. Thus we can skip y and exchange ayb in P with $a'b'$ to get the desired induced path P'. □

Theorem 18 *The* ORDERED-k-IN-A-PATH *problem can be solved in polynomial time in linear interval graphs.*

Proof. Let G be a linear interval graph. We may assume without loss of generality that the terminals form an independent set. We use its linear representation that we obtain in polynomial time by Theorem 9. In what follows the notions of predecessors (left) and successors (right) are considered for the linear ordering of the points on the line. Without loss of generality we may assume that t_1 is the first point and that t_k is the last and that no two points coincide. By our assumption, t_i and t_{i+1} are nonadjacent. From the set of points belonging to the closed interval $[t_i, t_{i+1}]$ we remove all neighbors of t_i except the rightmost one and all

neighbors of t_{i+1} except the leftmost. Then the shortest path between t_i and t_{i+1} is induced. In addition, these partial paths combined together provide a solution unless for some terminal t_i its leftmost predecessor is adjacent to its rightmost successor. Hence, no induced path may have t_i among its inner vertices. □

Lemma 19 *Let G be a composition of linear interval strips. It is possible to create in polynomial time a line graph G' with $|V(G')| \leq |V(G)|$, while preserving the solution.*

Proof. Let $G = S_m$ be a composition of linear interval strips. We first analyze the structure of a possible solution. This includes checking a number of necessary conditions that every linear interval strip (S'_i, a'_i, b'_i) must satisfy should G allow a solution.

Suppose S'_i contains some terminals other than t_1 and t_k. Then a solution P must enter S'_i through one of the neighbors of a'_i and leave through one neighbor of b'_i, or vice versa. Hence, we output No if the terminals on S'_i do not form a subsequence $t_j, t_{j+1}, \ldots, t_\ell$. Otherwise, the order of the terminals on the line determines whether P should leave S'_i through a neighbor of a'_i or through a neighbor of b'_i. In the first case we solve ORDERED-$(\ell - j + 3)$-IN-A-PATH on $(S'_i, a'_i, t_j, t_{j+1}, \ldots, t_\ell, b'_i)$ and in the second case on $(S'_i, b'_i, t_j, t_{j+1}, \ldots, t_\ell, a'_i)$. Because S'_i is an interval graph, we can do this in polynomial time due to Theorem 18. If this does not yield an induced path P_i, then we output No.

Suppose S'_i contains either t_1 or t_k together with at least one other terminal. We treat this case in the same way as in the previous case.

Suppose S'_i contains t_1 and t_k but not t_i for some $1 < i < k$. Then we simplify G by removing all vertices of S'_i between t_k and t_1 on the corresponding line. Hence, we may assume that this case does not occur.

Suppose S'_i contains either t_1 or t_k and no other terminals, say S'_i only contains t_1. A solution P will either leave S'_i through a neighbor of a'_i or through a neighbor of b'_i. Then we may without loss of generality assume that the corresponding subpath P_i is a shortest path from t_1 to a'_i or to b'_i.

Suppose S'_i contains no terminals. In that case a possible solution P can still pass through S'_i. Then we may without loss of generality assume that corresponding subpath P_i is a shortest path between a neighbor of a'_i and a neighbor of b'_i.

From now on, assume that we considered every linear interval strip and that we did not output No. It remains to show how the partial solutions P_i on the corresponding strips can be combined together. Without loss of generality we assume that the labels of a_i and b_i in S_0 conform the order of the terminals in the strips as discussed above.

We create a graph G' as follows. We connect any pair a_i, b_i by a path of length two, involving an extra new vertex c_i. If a strip S'_i contains a single terminal t_j, we let $t'_j := c_i$. If such strip contains more terminals $t_j, \ldots, t_l$, but neither t_1 nor t_k we let $t'_j := t'_{j+1} := \cdots := t'_{l-1} := a_i$ and $t'_l := b_i$. Otherwise, if $t_1, \ldots, t_l$ are in S'_i we let $t'_1 := \cdots := t'_{l-1} := c_i$ and $t'_l := b_i$. The case of t_k being the last terminal in S'_i is treated analogously. See Figure 2 for an example.

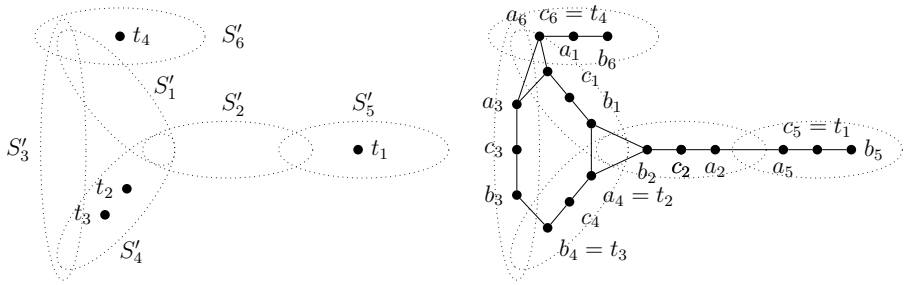

Fig. 2. A graph G'.

We claim that G has a solution P if and only if G' has a solution P'. This can be seen as follows. Any solution of G can be directly translated into a solution of G'. Suppose G' has a solution P'. We exchange subpaths $a_i c_i b_i$ by corresponding induced paths P_i in G, as explained above. Note that this is possible indeed, because we did not output `No` at some stage and ensured that there is no strip containing both t_1 and t_k. We conclude that creating G' preserves the solution.

What remains to show is that G' is a line graph. Consider the graph H constructed as follows.

- For every clique on S_0 we take a star with edges corresponding to the vertices of the clique.
- We connect the leaves incident with the edges corresponding to a_i and b_i by a new edge for $1 \leq i \leq m$ representing c_i.

Then G' is the line graph of H. This completes the proof of Lemma 19. □

Theorem 20 *For fixed k,* ORDERED-k-IN-A-PATH *can be solved in polynomial time for line graphs.*

Proof. Let G be a line graph with k terminals ordered as $t_1, \ldots, t_k$. Because we only remove vertices in Lemma 14 and line graphs are closed under vertex deletion, we may without loss of generality assume that G is convenient. For

$1 < i < k$, let s_i and u_i be the two neighbors of t_i. We set $a_1 = t_1$ and $b_{k-1} = t_k$. For $1 < i < k$ we choose b_{i-1} and a_i from $\{s_i, t_i\}$. This leads to $O(2^{k-2})$ equivalent instances of k-INDUCED DISJOINTS PATHS, which we can solve in polynomial time for line graphs by Corollary 5. □

4 Algorithmic Consequences

In this section we state some algorithmic consequences of our main result. We first consider the k-INDUCED DISJOINT PATHS and the k-IN-A-CYCLE problem for any fixed k.

Corollary 21 *For any fixed k, the k-INDUCED DISJOINT PATHS problem and the k-IN-A-CYCLE problem are solvable in polynomial time for claw-free graphs.*

Proof. Let G be a claw-free graph that together with terminals $t_1, \ldots, t_k$ is an instance of k-IN-A-CYCLE. We fix an order of the terminals, say, the order is $t_1, \ldots, t_k$. We fix two neighbors a_i and b_{i-1} of each terminal t_i. This way we obtain an instance of k-INDUCED DISJOINT PATHS with pairs of terminals (a_i, b_i) where $b_0 = b_k$. Clearly, the total number of instances we have created is polynomial. Hence, we can solve k-IN-A-CYCLE in polynomial time if we can solve k-INDUCED DISJOINT PATHS in polynomial time.

Let G be a claw-free graph that together with k pairs of terminals (a_i, b_i) for $i = 1, \ldots, k$ is an instance of the k-INDUCED DISJOINT PATHS problem. First we add an edge between each pair of non-adjacent neighbors of every terminal in $T = \{a_1, \ldots, a_k, b_1, \ldots, b_k\}$. We denote the resulting graphs obtained after performing this operation on a terminal by $G_1, \ldots, G_{2k}$, and define $G_0 := G$. We claim that $G' = G_{2k}$ is claw-free and prove this by induction.

The claim is true for G_0. Suppose the claim is true for G_j for some $0 \leq j \leq 2k-1$. Consider G_{j+1} and suppose, for contradiction, that G_{j+1} contains an induced subgraph isomorphic to a claw. Let $K := \{x, a, b, c\}$ be a set of vertices of G_{j+1} inducing a claw with center x. Let $s \in T$ be the vertex of G_j that becomes simplicial in G_{j+1}. Then $x \neq s$. Since G_j is claw-free, we may without loss of generality assume that at least two vertices of K must be in $N_{G_{j+1}}(s) \cup \{s\}$. Since $N_{G_{j+1}}(s) \cup \{s\}$ is a clique of G_{j+1} and $\{a, b, c\}$ is an independent set of G_{j+1}, we may without loss of generality assume that $K \cap (N_{G_{j+1}}(s) \cup \{s\}) = \{x, a\}$ and $\{b, c\} \subseteq V(G_{j+1}) \setminus (N_{G_{j+1}}(s) \cup \{s\})$. Then $\{x, b, c, s\}$ induces a claw in G_j with center x, a contradiction. Hence, G' is indeed claw-free. We note that G with terminals $(a_1, b_1), \ldots, (a_k, b_k)$ forms a YES-instance of k-INDUCED DISJOINT PATHS if and only if G' with the same terminal pairs is a YES-instance of this problem.

We modify G' by adding a new vertex c_i and edges b_ic_i and c_ia_{i+1} for $i = 1, \ldots, k-1$. We call the resulting graph G'' and observe that G'' is claw-free, because the neighborhood of every terminal in G' forms a clique. We will show that G' with terminal pairs $(a_1, b_1), \ldots, (a_k, b_k)$ forms a YES-instance of the k-INDUCED PATHS problem if and only if G'' with ordered terminals $a_1, b_1, c_1, a_2, b_2, c_2, \ldots, a_{k-1}, b_{k-1}, c_{k-1}, a_k, b_k$ forms a YES-instance of the ORDERED-$(3k-1)$-IN-A-PATH problem. Then we can apply Theorem 13 and are done.

In order to see the above claim, suppose G' contains k mutually induced paths P_i such that P_i connects a_i to b_i for $1 \leq i \leq k$. Then

$$P = a_1\overrightarrow{P_1}b_1c_1a_2\overrightarrow{P_2}b_2 \ldots b_{k-1}c_{k-1}a_k\overrightarrow{P_k}b_k$$

is an induced path passing through the $3k-1$ terminals in prescribed order. Now suppose G'' contains an induced path P passing through the $3k-1$ terminals in desired order. Then P contains segments $b_ic_ia_{i+1}$ for $i = 1, \ldots, k-1$. This means that for $i = 1, \ldots, k$ we can define paths $P_i = a_i\overrightarrow{P}b_i$, which are mutually induced. This completes the proof of Corollary 21. □

We finish this section with a consequence of Corollary 21. We first need some additional terminology. The *subdivision* of an edge uw removes the edge uw and replaces it by a new vertex v and two new edges uv and vw. Let H be a graph. A graph is a *subdivision* of H if it can be obtained from H by a sequence of edge subdivisions. A graph G contains H as an *induced topological minor* if G contains an induced subgraph that is isomorphic to a subdivision of H. The decision problem H-INDUCED TOPOLOGICAL MINOR is to test if a given graph contains H as an induced topological minor. Here, H is fixed, so not part of the input.

Lévêque et al. [21] generalize induced topological minors as follows. They define a *subdivisible graph* (or *s-graph*) as a graph with two types of edges, namely so-called *subdivisible* edges and so-called *real* edges. Let H be an s-graph. Then a graph is a *realization* of H if it can be obtained from H by a sequence of subdivisions of subdivisible edges, i.e., by replacing subdivisible edges in H by paths of length at least one. This leads to the decision problem H-INDUCED REALIZATION which has as input a graph G and asks whether G contains a realization of H as an induced subgraph. Also here, H is to be considered as fixed. If H is an s-graph with only subdivisible edges, then H-INDUCED REALIZATION is equivalent to H-INDUCED TOPOLOGICAL MINOR. Note that in the other extreme, i.e., when H only contains real edges, the problem comes down to testing if a graph has an induced subgraph isomorphic to H.

The complexity of H-INDUCED REALIZATION and H-INDUCED TOPOLOGICAL MINOR depends on the fixed graph H. Already for the second prob-

lem, many polynomially solvable and NP-complete cases are known. For example, Lévêque et al. [21] show that K_5-INDUCED TOPOLOGICAL MINOR is NP-complete, whereas e.g. C_3-INDUCED REALIZATION is polynomially solvable. We refer to their paper [21] for more results.

The complexity classification of H-INDUCED TOPOLOGICAL MINOR, and consequently, of H-INDUCED REALIZATION is far from being finished for general graphs. A notorious open problem is determining the complexity of $2C_3$-INDUCED TOPOLOGICAL MINOR, where $2C_3$ denotes the disjoint union of two 3-vertex cycles. This problem is polynomially equivalent to the $2C_4$-INDUCED TOPOLOGICAL MINOR problem. We note that the latter can also be formulated as the problem that asks whether a graph $G = (V, E)$ contains two *mutually induced* holes, i.e., two holes C and D with $V(C) \cap V(D) = \emptyset$ and $uv \notin E$ for any $u \in V(C)$ and $v \in V(D)$. However, for claw-free graphs, we can solve this case and any other case in polynomial time by applying Corollary 21 as follows.

Corollary 22 *For any fixed s-graph H, the H-INDUCED REALIZATION problem can be solved in polynomial time for claw-free graphs.*

Proof. We first slightly modify the k-INDUCED DISJOINT PATHS problem. Fix an integer k, and let $G = (V, E)$ be a claw-free graph with terminal pairs (a_i, b_i) for $i = 1, \ldots, k$. Then we allow $a_i = a_j$, $a_i = b_j$ or $b_i = b_j$ for any $1 \leq i \leq j \leq k$, and we also allow the existence of an edge between any two terminals in $\{a_1, \ldots, a_k, b_1, \ldots, b_k\}$. So, in this modification, we test if G contains a path P_i from a_i to b_i for $i = 1, \ldots, k$ such that for any $1 \leq i < j \leq k$, $(V(P_i) \backslash \{a_i, b_i\}) \cap (V(P_j) \backslash \{a_j, b_j\}) = \emptyset$ and $uv \notin E$ for every $u \in V(P_i)$ and $v \in V(P_j)$ with $uv \notin \{a_i a_j, a_i b_j, b_i a_j, b_i b_j\}$. The following argument shows why we can make these two assumptions. Suppose $a_i = a_j$. We choose two non-adjacent neighbors u, v of a_i and remove all its other neighbors. If such neighbors do not exist, we output No. Otherwise, this results in a new claw-free graph, where terminal pairs (u, b_i) and (v, b_j) replace (a_i, b_i) and (a_j, b_j). We perform a similar operation for the second assumption. In this way we create at most $O(n^{2k})$ new instances of the original k-INDUCED DISJOINT PATHS problem, which we can solve in polynomial time due to Corollary 21.

Now, let H be a fixed s-graph. Let D be the set of real edges and F the set of subdivisible edges, so $E(H) = D \cup F$ with $D \cap F = \emptyset$. Let G be a claw-free graph on n vertices. We are to test if G contains an induced subgraph that is isomorphic to a realization of H. We do this as follows.

For each vertex in H we guess a corresponding vertex in G. This results in a set U of guessed vertices in G. We check if every real edge of H is represented in $G[U]$. We also check if every non-edge in H corresponds to a non-edge in $G[U]$. If one of these checks is negative, we output No. Otherwise, we continue

as follows. Let $U' \subseteq U$ denote the set of vertices that correspond to vertices in H that are not incident with any subdivisible edge. Let W denote the set of pairs $\{u, v\}$ with $u, v \in U \backslash U'$ such that uv corresponds to a subdivisible edge in H. For every vertex in U' we remove all its neighbors that are not in U from G. The resulting graph is claw-free and forms together with the pairs in W an instance of $|W|$-INDUCED DISJOINTS PATHS, which we can solve in polynomial time as explained above. Because the number of possible sets U is at most $O(n^{|V(H)|})$ and H is fixed, our algorithm runs in polynomial time. □

5 When k Is Part of the Input

We accompany our constructive algorithm for the three problems k-IN-A-PATH, k-INDUCED DISJOINT PATHS and k-IN-A-CYCLE with NP-hardness proofs for the case when the number of terminals becomes unbounded.

Theorem 23 *The k-IN-A-PATH problem and the k-IN-A-CYCLE problem are* NP*-complete even for the class of line graphs, when k is part of the input.*

Proof. It is straightforward to verify whether a given path or cycle is induced and contains all prescribed terminals. Hence membership of these problems in the class NP is settled.

A *cubic* graph is a graph in which every vertex has degree three. For the hardness part for the k-IN-A-PATH problem we show a reduction from the classical HAMILTONIAN PATH problem, which is already NP-complete for cubic graphs [13, problem GT39].

Let G be a cubic input graph for the HAMILTONIAN PATH problem. We replace every vertex (of degree three) in G by a triangle. Formally, if u is incident with edges $e(u, 1), e(u, 2)$ and $e(u, 3)$, then we replace u with three vertices u_1, u_2 and u_3, where each u_i becomes incident with $e(u, i)$ instead of u. Moreover, these three new vertices induce a triangle C_3. Let G' be the graph resulting from G after all original vertices have been replaced.

Observe that a hamiltonian path in G exists if and only if G' has a path that passes through all edges $\{u_1u_2 \mid u \in V(G)\}$:

- having the hamiltonian path in G, one can easily extend it inside every triangle in G' through the associated prescribed edge;
- as the path in G' may visit each triangle at most once, after contraction of all triangles into the original vertices of G we get a hamiltonian path of G.

We now take $H = L(G')$ to be the line graph of G' and the terminals to be the images of the chosen edges u_1u_2 for all $u \in V(G)$, as displayed in Figure 3.

The NP-completeness proof for k-IN-A-PATH is concluded by the already mentioned argument that induced paths in H are in one-to-one correspondence with ordinary paths in G'.

In the same way we prove that the k-IN-A-CYCLE problem is NP-complete for the class of line graphs when k is part of the input. The only differences are the following. Firstly, we make a reduction from the HAMILTONIAN CYCLE problem, which is also NP-complete for the class of cubic graphs (cf. [13]). Secondly, we observe that induced cycles (except the C_3) in a line graph are in one-to-one correspondence with cycles in the original graph. □

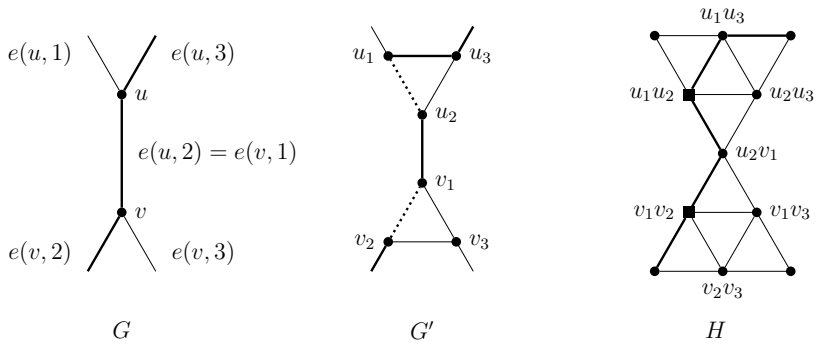

Fig. 3. The reduction from the HAMILTONIAN PATH problem. The thick edges show a fragment of the solution, and the emphasized objects are the prescribed edges or the terminals.

We now consider the k-INDUCED DISJOINT PATHS problem.

Theorem 24 *The k-INDUCED DISJOINT PATHS problem is NP-complete even for the class of line graphs, when k is part of the input.*

Proof. We can check in polynomial time if a given collection of paths is a solution of the k-INDUCED DISJOINT PATHS problem. Hence, this problem is in NP. In order to show NP-completeness we reduce from the k-DISJOINT PATHS problem, which is NP-complete when k is part of the input [17].

Let G together with k mutually disjoint vertex pairs $(s_1, t_1), \ldots, (s_k, t_k)$ form an instance of the k-DISJOINT PATHS problem. For $i = 1, \ldots, k$ we add a vertex s'_i with edge $s_i s'_i$ and a vertex t'_i with edge $t_i t'_i$ to G. This results in graph G'. We then observe that the line graph of G' contains k mutually induced paths connecting $s_i s'_i$ to $t_i t'_i$ for $i = 1, \ldots, k$ if and only if G' contains k mutually disjoint paths connecting s'_i to t'_i for $i = 1, \ldots, k$ if and only if G contains k mutually disjoint paths connecting s_i to t_i for $i = 1, \ldots, k$. □

6 Conclusions and Further Research

We studied the three problems k-IN-A-PATH, k-INDUCED DISJOINT PATHS and k-IN-A-CYCLE. If k is part of the input these problems are known to be NP-complete, and we showed this stays true, even when the input graphs are line graphs. On the positive side, we showed that, for any fixed k, all three problems are polynomially solvable on claw-free graphs.

Our algorithms run in $k!n^{O(k)}$ time on n-vertex graphs. So, contrary to the cubic algorithm of Robertson and Seymour [22] that solves the k-DISJOINT PATHS problem, the order of the polynomial in the running time of our algorithms heavily depends on the integer k. We refrain from stating a more precise running time, because our motivation was strictly theoretical. Our goal was to show membership of the three problems in the class P for claw-free graphs. Hence, we were less concerned with decreasing the running time of our algorithms (which certainly seems possible by a more refined analysis). Nevertheless, it is an interesting question whether these problems are *fixed parameter tractable* in k for claw-free graphs, i.e., whether they can be solved in $O(f(k)n^c)$ time, where f denotes a computable function and c a constant independent of k. We note that for line graphs this is the case due to the aforementioned observation that vertex-disjoint paths in a graph are in one-to-one correspondence with mutually induced paths in its line graph. This enables us to use the $O(f(k)n^3)$ time algorithm of Robertson and Seymour [22] that solves the k-DISJOINT PATHS problem.

The following two related problems are also fascinating open problems.

Problem 1. Determine the computational complexity of deciding whether a graph contains an odd hole.

Problem 2. Determine the computational complexity of deciding whether a graph contains two mutually induced holes.

For claw-free graphs, Problems 1 and 2 are solved. For this graph class, we solved Problem 2 in Corollary 22, while it is well known that Problem 1 can be solved in polynomial time for claw-free graphs by using the polynomial time algorithm for recognizing claw-free perfect graphs of Chvátal and Sbihi [6], and by the following lemma, a proof of which can be found in [6].

Ben Rebea's Lemma *If a connected claw-free graph with three independent vertices contains an odd antihole, then it contains a hole of length 5.*

In order to see this, let G be a claw-free graph on n vertices and assume without loss of generality that G is connected. We first run the algorithm of Chvátal and Sbihi [6] to decide if G is perfect. If so, then it contain no odd hole. If not, we do as follows. If G has at least three independent vertices (this can be checked

in $O(n^3)$ time) then G contains an odd hole due to Ben Rebea's Lemma, regardless whether an odd antihole is present or not. Finally, if G contains at most 2 independent vertices, then it does not contain an odd hole on 7 vertices or more. Hence, we are left to check whether G contains a hole on 5 vertices. This can be done in $O(n^5)$ time.

Recently, an $O(nm^2)$ algorithm that finds a shortest odd hole in a claw-free graph on n vertices and m edges has been given by Shrem, Stern and Golumbic [24]. In addition, we note that the problem of finding an *even* hole is polynomially solvable for general graphs [2] and that the problem of testing whether a graph contains two mutually induced *odd* holes is NP-complete [14].

Acknowledgments We would like to thank the two anonymous referees for their helpful comments.

References

1. D. Bienstock. On the complexity of testing for odd holes and induced odd paths. *Discrete Mathematics* **90** (1991) 85–92, See also Corrigendum, Discrete Mathematics **102** (1992) 109.
2. M. Chudnovsky, K. Kawarabayashi and P.D. Seymour. Detecting even holes. *Journal of Graph Theory* **48** (2005) 85–111.
3. M. Chudnovsky, N. Robertson, P.D. Seymour, and R. Thomas. The strong perfect graph theorem. *Annals of Mathematics* **164** (2006) 51–229.
4. M. Chudnovsky and P.D. Seymour. The structure of claw-free graphs. In: B.S. Webb (ed.) *Surveys in combinatorics, 2005*, London Mathematical Society Lecture Notes Series, vol. 327, pp. 153–171, Cambridge University Press, Cambridge (2005).
5. M. Chudnovsky and P.D. Seymour. The three-in-a-tree problem. *Combinatorica* **30** (2010) 387–417.
6. V. Chvátal and N. Sbihi. Recognizing claw-free perfect graphs. *Journal of Combinatorial Theory, Series B* **44** (1988)154–176.
7. X. Deng, P. Hell, and J. Huang. Linear time representation algorithm for proper circular-arc graphs and proper interval graphs. *SIAM Journal on Computing* **25** (1996) 390–403.
8. N. Derhy and C. Picouleau. Finding induced trees. *Discrete Applied Mathematics* **157** (2009) 3552–3557.
9. N. Derhy, C. Picouleau, and N. Trotignon. The four-in-a-tree problem in triangle-free graphs. *Graphs and Combinatorics* **25** (2009) 489–502.
10. R. Faudree, E. Flandrin, and Z. Ryjáček. Claw-free graphs—a survey. *Discrete Mathematics* **164** (1997) 87–147.
11. M.R. Fellows. The RobertsonSeymour theorems: A survey of applications. In: R.B. Richter (ed.) *Proceedings of the AMS-IMS-SIAM Joint Summer Research Conference*, Contemporary Mathematics, vol. 89, pp. 1–18, American Mathematical Society, Providence (1989).
12. D. Fulkerson and O. Gross, Incidence matrices and interval graphs, *Pacific Journal of Mathematics* **15** (1965) 835–855.
13. M.R. Garey and D.S. Johnson. *Computers and Intractability*. W. H. Freeman and Co., New York, 1979.

14. P. Golovach, M. Kamiński, D. Paulusma, and D. M. Thilikos. Induced packing of odd cycles in a planar graph. In: Y. Dong, D.Z. Du, O.H. Ibarra (eds.) *Proceedings of the 20th International Symposium on Algorithms and Computation*, ISAAC 2009, Lecture Notes in Computer Science, vol. 5878, pp. 514–523, Springer, Berlin (2009).
15. R. Haas and M. Hoffmann. Chordless paths through three vertices. *Theoretical Computer Science* **351** (2006) 360–371.
16. P. van 't Hof, M. Kamiński and D. Paulusma. Finding induced paths of given parity in claw-free graphs. In: C. Paul, M. Habib (eds.) *Proceedings of the 35th International Workshop on Graph-Theoretic Concepts in Computer Science*, WG 2009, Lecture Notes in Computer Science, vol. 5911, pp. 341–352, Springer, Berlin (2009).
17. R.M. Karp. On the complexity of combinatorial problems. *Networks* **5** (1975) 45–68.
18. A. King and B. Reed. Bounding χ in terms of ω and δ for quasi-line graphs. *Journal of Graph Theory* **59** (2008) 215-228.
19. Y. Kobayashi and K. Kawarabayashi. The induced disjoint paths problem. In: A. Lodi, A. Panconesi, G. Rinaldi (eds.) *Proceedings of the 13th Conference on Integer Programming and Combinatorial Optimization*, IPCO 2008, Lecture Notes in Computer Science, vol. 5035, pp. 47–61, Springer, Berlin (2008).
20. Y. Kobayashi and K. Kawarabayashi. Algorithms for finding an induced cycle in planar graphs and bounded genus graphs. In: C. Mathieu (ed.) *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 2009, pp. 1146–1155, ACM Press, New York (2009).
21. B. Lévêque, D.Y. Lin, F. Maffray, and N. Trotignon. Detecting induced subgraphs. *Discrete Applied Mathematics* **157** (2009) 3540–3551.
22. N. Robertson and P.D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B* **63** (1995) 65–110.
23. F.S. Roberts, Indifference Graphs, In: F. Harary (ed.) *Proof Techniques in Graph Theory*, pp. 139–146, Academic Press, New York (1969).
24. S. Shrem, M. Stern and M.C. Golumbic. Smallest odd holes in claw-free graphs. In: C. Paul, M. Habib (eds.) *Proceedings of the 35th International Workshop on Graph-Theoretic Concepts in Computer Science*, WG 2009, Lecture Notes in Computer Science, vol. 5911, pp. 329–340, Springer, Berlin (2009).
25. W. Liu and N. Trotignon. The k-in-a-tree problem for graphs of girth at least k. *Discrete Applied Mathematics* **158** (2010) 1644–1649.