Finding induced paths of given parity in claw-free graphs[⋆,⋆⋆]

Pim van 't Hof[1], Marcin Kamiński[2] and Daniël Paulusma[1]

[1] School of Engineering and Computing Sciences, University of Durham, Science Laboratories, South Road, Durham DH1 3LE, England.
{pim.vanthof,daniel.paulusma}@durham.ac.uk

[2] Computer Science Department, Université Libre de Bruxelles, Boulevard du Triomphe CP212, B-1050 Brussels, Belgium
marcin.kaminski@ulb.ac.be

Abstract. The Parity Path problem is to decide if a given graph contains both an induced path of odd length and an induced path of even length between two specified vertices. In the related problems Odd Induced Path and Even Induced Path, the goal is to determine whether an induced path of odd, respectively even, length between two specified vertices exists. Although all three problems are NP-complete in general, we show that they can be solved in $\mathcal{O}(n^5)$ time for the class of claw-free graphs. Two vertices s and t form an even pair in G if every induced path from s to t in G has even length. Our results imply that the problem of deciding if two specified vertices of a claw-free graph form an even pair, as well as the problem of deciding if a given claw-free graph has an even pair, can be solved in $\mathcal{O}(n^5)$ time and $\mathcal{O}(n^7)$ time, respectively. We also show that we can decide in $\mathcal{O}(n^7)$ time whether a claw-free graph has an induced cycle of given parity through a specified vertex. Finally, we show that a *shortest* induced path of given parity between two specified vertices of a claw-free perfect graph can be found in $\mathcal{O}(n^7)$ time.

1 Introduction

Finding a shortest path, a maximum stable set or a hamiltonian cycle in a graph are just a few examples from the wide spectrum of problems dealing with finding a subgraph (or induced subgraph) with some particular property. In this context, very simple subgraphs, such as paths, trees and cycles, with some prescribed property are often studied. The following problem has been extensively studied in the context of perfect graphs. Here, the *length* of a path refers to its number of edges, and a path is said to be *odd* (respectively *even*) it has odd (respectively even) length.

[⋆] An extended abstract of this paper has been presented at the 35th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2009).

[⋆⋆] This work has been supported by EPSRC (EP/D053633/1) and the Actions de Recherche Concertées (ARC) fund of the Communauté française de Belgique.

Parity Path
Instance: A graph G and two vertices s, t of G.
Question: Does G contain both an odd induced path and an even induced path from s to t?

We focus on the closely related problem of deciding whether there exists an induced path of given parity between a pair of vertices. In particular, we study the following two problems.

Odd Induced Path
Instance: A graph G and two vertices s, t of G.
Question: Does G contain an odd induced path from s to t?

Even Induced Path
Instance: A graph G and two vertices s, t of G.
Question: Does G contain an even induced path from s to t?

The Odd Induced Path problem was shown to be NP-complete by Bienstock [6]. Consequently, the Even Induced Path problem and the Parity Path problem are NP-complete as well. Several authors however have identified a number of graph classes that admit polynomial-time algorithms for these problems. Below we survey those results, as well as results on related problems, before stating our contribution. Throughout the paper, we use n and m to denote the number of vertices and the number of edges of the input graph, respectively.

Odd Path and Even Path. In the Odd Path and Even Path problems the task is to find a (not necessarily induced) path of given parity between a specified pair of vertices. These problems were considered by LaPaugh and Papadimitriou [24]. They mention an $\mathcal{O}(n^3)$ time algorithm for solving both problems due to Edmonds and propose a faster one of $\mathcal{O}(m)$ time complexity. Their algorithm also finds a *shortest* (not necessarily induced) path of given parity between two vertices in $\mathcal{O}(m)$ time, even in a weighted graph. Interestingly, as they also show in their paper, the problem of finding a directed path of given parity is NP-complete for directed graphs.

Arkin, Papadimitriou and Yannakakis [1] generalized the result of [24] and designed a linear-time algorithm deciding if all (not necessarily induced) paths between two specified vertices are of length $p \bmod q$, for fixed integers p and q.

Even Pair. First interest in induced paths of given parity comes from the theory of perfect graphs. Two non-adjacent vertices are called an *even pair* if every induced path between them is even. The Even Pair problem is to decide if a given pair of vertices forms an even pair. The Even Pair problem is co-NP-complete due to Bienstock [6], as is the problem of deciding if a graph contains an even pair. The interest in even pairs was sparked by an observation of Fonlupt and Uhry [19]: if a graph is perfect and contains an even pair, then the graph obtained by identifying the vertices that form the even pair is also perfect. Later Meyniel showed that minimal non-perfect graphs contain no even pair [29]. Those two facts triggered a series of theoretical and algorithmic results which are surveyed in [16] and its updated version [17].

There is some evidence that perfect graphs without an even pair can be generated by performing a small number of composition operations on some basic graphs. Using such a structural result could then lead to a combinatorial algorithm for coloring perfect graphs. Indeed, for coloring perfect graphs using at most three colors this approach turned out to be successful, as was shown by Chudnovsky and Seymour [11]. Linhares Sales and Maffray [27] study even pairs in order to give characterizations of claw-free graphs that are strict quasi-parity and perfectly contractile, respectively.

Parity Path and Group Path. Arikati, in a series of papers with different coauthors [2–4], developed polynomial-time algorithms for the Parity Path problem in different classes of graphs. Chordal graphs are considered by Arikati and Peled [2], who present a linear-time algorithm for the Group Path problem, a generalization of the Odd Induced Path problem. In the Group Path problem the edges of the input graph are weighted with elements of some group $\mathcal{G}$. The problem is to find an induced path of given weight between two specified vertices, where the weight of a path is defined as the product of the weights of the edges of the path. They present an $\mathcal{O}(|\mathcal{G}|m + n)$ time algorithm for the Group Path problem on chordal graphs using a perfect elimination ordering.

Arikati, Rangan, and Manacher [4] consider Parity Path on circular-arc graphs and show how to reduce the problem to interval graphs by recursively applying a set of reductions. Since interval graphs are chordal, the algorithm of [2] can be used to obtain the solution. This way they obtain a polynomial-time algorithm for circular-arc graphs.

Satyan and Rangan [34] present polynomial-time algorithms for the Parity Path problem on comparability and cocomparability graphs, and a linear-time algorithm for permutation graphs. A polynomial-time algorithm for Parity Path on perfectly orientable graphs is presented by Arikati and Peled [3]. Sampaio and Sales [33] obtain a polynomial-time algorithm for planar perfect graphs. Figueiredo et al. [18] characterize even pairs and odd pairs in comparability and P_4-comparability graphs and give polynomial-time algorithms for the Parity Path problem in those classes. Hoàng and Le [22] show that Parity Path can be solved in polynomial time for the class of 2-split graphs.

Note that a set F of vertices of a line graph $G = L(H)$ form an odd (respectively even) induced path in G if and only if the set of edges corresponding to F form an even (respectively odd) path in the preimage graph H of G. It is well-known that the preimage graph of a line graph can be found in polynomial time [31]. Combining these two facts with the polynomial-time algorithm for finding (not necessarily induced) paths of given parity in [24] yields a polynomial-time algorithm for solving the Parity Path problem for the class of line graphs (cf. [37]).

Our results. Our interest in the Odd Induced Path problem was in part stirred by studying Bienstock's NP-completeness reduction [6]. He builds a graph out of a 3-Sat formula and shows that the formula is satisfiable if and only if there exists an odd induced path between a certain pair of vertices. This is also shown to be equivalent to the existence of two disjoint induced paths (with no edges between the two paths) between certain pairs of vertices in the construction. Finding such two paths is then NP-hard in general but has been proved solvable in polynomial time for claw-free graphs [25]. A natural question to ask is whether the Odd Induced Path problem can also be solved in polynomial time for this class of graphs. In this paper, we answer this question in the affirmative by presenting an algorithm that solves both the Odd Induced Path problem and the Even Induced Path problem in $\mathcal{O}(n^5)$ time for the class of claw-free graphs. This implies that the Even Pair problem can be solved in $\mathcal{O}(n^5)$ time for claw-free graphs.

As we saw earlier in this section, the Parity Path problem has been extensively studied in different graph classes. However, a polynomial-time algorithm for claw-free graphs has never been proposed; somewhat surprising, since claw-free graphs form a large and important class containing, e.g., the class of line graphs and the class of complements of triangle-free graphs. Our $\mathcal{O}(n^5)$ time algorithm for solving the Odd Induced Path and Even Induced Path problems for claw-free graphs immediately implies that we can solve the Parity Path problem for claw-free graphs in $\mathcal{O}(n^5)$ time, thus generalizing the aforementioned polynomial-time result on line graphs. Making use of the structure of claw-free perfect graphs we also obtain an $\mathcal{O}(n^7)$ time algorithm for finding *shortest* induced paths of given parity between two specified vertices in a claw-free perfect graph.

Apart from the Odd Induced Path problem, Bienstock [6] mentioned two more NP-complete problems in the abstract of his paper. The first one is to decide whether a graph has an odd hole passing through a given vertex. The second one is to decide whether a graph has an odd induced path between *every* pair of vertices. We show that our polynomial-time algorithm for the Odd Induced Path problem implies that both these problems are solvable in $\mathcal{O}(n^7)$ time when restricted to the class of claw-free graphs. The same holds for the problem of deciding whether or not a claw-free graph contains an even pair.

Paper organization. We start in Section 2 by performing a running time analysis of the algorithm for recognizing claw-free perfect graphs due to Chvátal and Sbihi [12]. This algorithm is used as a subroutine in our algorithm presented in Section 3, which solves both the Odd Induced Path problem and the Even Induced Path problem in $\mathcal{O}(n^5)$ time for the class of claw-free graphs. The key ideas behind the algorithm in [12] will be used to obtain the results in Section 4, where we present an $\mathcal{O}(n^7)$ time algorithm for finding shortest induced paths of given parity between two specified vertices in a claw-free perfect graph. Section 5 contains the conclusions and mentions some open problems.

Preliminaries. All graphs in this paper are undirected, finite, and have no loops or multiple edges. We refer to [15] for terminology not defined below. Let G be a graph. We refer to the vertex set and edge set of G by $V(G)$ and $E(G)$, respectively. The *neighborhood* of a vertex v in G is the set $N_G(v) = \{w \in V(G) \mid vw \in E(G)\}$ of neighbors of v in G. The *closed neighborhood* of v is the set $N_G[v] = N_G(v) \cup \{v\}$. If $N_G[v] = V(G)$ for every vertex v, then G is called *complete*. For any subset $S \subseteq V(G)$, we write $G[S]$ to denote the subgraph of G induced by S. The vertices of S form a *clique* in G if $G[S]$ is complete. A vertex v is called *simplicial* if its neighbors form a clique. For any proper subset $S \subset V(G)$, we write $G - S$ to denote the graph $G[V(G) \setminus S]$, i.e., the graph obtained from G by removing all vertices of S and their incident edges. If $S = \{v\}$, we write $G - v$ instead of $G - \{v\}$. A *separator* S of a connected graph G is a set of vertices of G such that $G - S$ is not connected.

A graph is called *claw-free* if it has no induced subgraph isomorphic to the *claw*, i.e., the four-vertex star $K_{1,3} = (\{x, a, b, c\}, \{xa, xb, xc\})$, where vertex x is called the *center* of the claw. 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* (respectively *even*) if it has an odd (respectively even) number of edges. An antihole is called odd (respectively even) if its complement is an odd (respectively even) hole. The *length* of an antihole is the number of edges in its complement. A graph is called *Berge* if it does not contain an odd hole or an odd antihole. The *chromatic number* of a graph is the smallest number of colors needed to color its vertices in such a way that no two adjacent vertices receive the same color. A graph G is *perfect* if for every induced subgraph H the chromatic number of H equals the size of a largest clique in H.

We denote the path on k vertices by P_k. Let $P = v_1v_2 \dots v_p$ be a path with a fixed orientation. For $j \geq i$, we write $v_i\overrightarrow{P}v_j$ to denote the path $v_iv_{i+1} \dots v_j$, and $v_j\overleftarrow{P}v_i$ to denote the path $v_jv_{j-1} \dots v_i$.

2 Recognizing claw-free perfect graphs in $\mathcal{O}(n^4)$ time

A little over 40 years after Berge [5] conjectured that a graph is perfect if and only if it is Berge, Chudnovsky et al. [9] confirmed his intuition by proving the following theorem.

Theorem 1 (Strong Perfect Graph Theorem, [9]). *A graph is perfect if and only if it contains no odd hole and no odd antihole.*

Shortly afterwards, Chudnovsky et al. [7] presented an $\mathcal{O}(n^9)$ time algorithm for recognizing Berge graphs. This means we can determine in $\mathcal{O}(n^9)$ time whether or not a graph is perfect. The main goal of this section is to show that the problem of deciding whether or not a claw-free graph is perfect can be solved in $\mathcal{O}(n^4)$ time. We point out that we do not actually present a new algorithm for recognizing claw-free perfect graphs, but merely perform a running time analysis of an existing recognition algorithm due to Chvátal and Sbihi [12]. Chvátal and Sbihi did not explicitly state the time complexity of their recognition algorithm, and to the best of our knowledge no better upper bound on the time needed to recognize claw-free perfect graphs than the aforementioned $\mathcal{O}(n^9)$ can be found in the literature. We will use the recognition algorithm for claw-free perfect graphs as a subroutine in the algorithm presented in Section 3. Moreover, in Section 4 we will exploit the fascinating structure of perfect claw-free graphs exhibited by Chvátal and Sbihi [12] in order to obtain an algorithm for finding shortest induced paths of given parity in such graphs. Before we present the key ideas behind Chvátal and Sbihi's algorithm, we need to introduce some terminology and techniques they use in their paper.

A set $X \subseteq V(G)$ is a *clique separator* of a connected graph G if X is both a clique and a separator of G. Suppose that the graph obtained from G by deleting X consists of k connected components with vertex sets $V_1, \ldots, V_k$. We call the graphs $G[V_1 \cup X], \ldots, G[V_k \cup X]$ the *children* of G produced by X. If any child $G[V_i \cup X]$ contains a clique separator X_i, we continue decomposing the graph G by replacing $G[V_i \cup X]$ by the children of $G[V_i \cup X]$ produced by X_i. Repeating this procedure until the graph cannot be decomposed any further yields a collection $\mathcal{C} = \{G_1, \ldots, G_p\}$ of induced subgraphs of G without a clique separator, called the *atoms* of G. We refer to the set $\mathcal{C}$ as a *clique separator decomposition* of G. Several authors designed polynomial-time algorithms for finding a clique separator decomposition of a graph, the fastest one being due to Tarjan [36]. We give an explicit description of Tarjan's algorithm in Section 4, where we also prove some properties about the obtained clique separator decomposition. These properties are then used in the proof of the main result of that section. For now, we only mention the following result.

Theorem 2 ([36]). *For a connected graph G, Tarjan's decomposition algorithm finds a clique separator decomposition with at most $n-1$ atoms in $\mathcal{O}(nm)$ time.*

According to Whitesides [38], the original motivation to study clique separator decompositions is their relation to the problem of recognizing perfect graphs.

Theorem 3 ([38]). *A graph is perfect if and only if all its atoms are perfect.*

Chvátal and Sbihi [12] discovered that all atoms in a clique separator decomposition of a claw-free perfect graph G belong to one of two classes of graphs, which they called "elementary" and "peculiar". We now give the definitions of elementary and peculiar graphs, and show that we can determine in $\mathcal{O}(n^3)$ time whether a graph belongs to one of those classes.

Definition 1. *A graph H is* elementary *if its edges can be colored with two colors such that every induced path on three vertices has its two edges colored differently. We call such a coloring an* elementary coloring *of H.*

See Figure 1 for an example of an elementary graph with an elementary coloring, where the light edges and heavy edges are colored differently. This graph was presented as an example of an elementary graph in [28]. The authors of [12] describe how elementary graphs can be recognized in polynomial time. Using their arguments, it is easy to prove the time complexity in the following lemma.

Lemma 1 ([12]). *It is possible to decide in $\mathcal{O}(n^3)$ time whether or not a graph H is elementary. If it is, an elementary coloring of H can be found in $\mathcal{O}(n^3)$ time.*

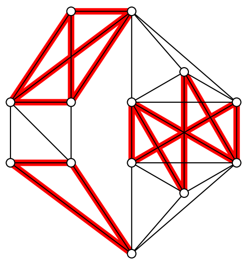

Fig. 1. An elementary graph with an elementary coloring.

Proof. Let H be a graph on n vertices and m edges. We construct a graph $\Gamma(H)$ on m vertices as follows: $V(\Gamma(H)) = E(H)$ and two vertices in $\Gamma(H)$ are adjacent if and only if the corresponding edges in G induce a path on three vertices. It is clear that we can compute the graph $\Gamma(H)$ in $\mathcal{O}(n^3)$ time by checking for each triple of vertices in $V(H)$ whether they induce a path on three vertices in H. For any edge uv of H, there are at most n vertices of H that are adjacent to exactly one vertex of $\{u, v\}$, which means that the graph $\Gamma(H)$ has at most nm edges. It is easy to see that H is elementary if and only if $\Gamma(H)$ is bipartite, and any 2-coloring of $\Gamma(H)$ corresponds to an elementary coloring of H. Clearly, finding a 2-coloring of a graph with m vertices and nm edges, or concluding that such a coloring does not exist, can be done in $\mathcal{O}(m + nm) = O(n^3)$ time. $\square$

Definition 2. *A graph H is* peculiar *if it can be obtained from a complete graph K as follows. Partition $V(K)$ into six mutually disjoint non-empty sets A_i, B_i, $i = 1, 2, 3$. For each $i = 1, 2, 3$, remove at least one edge with one end-vertex in A_i and the other end-vertex in B_{i+1}, where the subscript 4 is interpreted as 1. Finally, add three new mutually disjoint non-empty complete graphs D_i, $i = 1, 2, 3$, and for each $i = 1, 2, 3$ make each vertex in D_i adjacent to all vertices in $V(K) \setminus (A_i \cup B_i)$.*

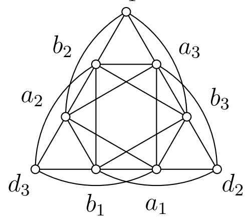

Fig. 2. The smallest possible peculiar graph.

The smallest possible peculiar graph is depicted in Figure 2. Li and Zang [26] present a simple polynomial-time algorithm for recognizing peculiar graphs.

Lemma 2 ([26]). *It is possible to decide in $\mathcal{O}(n^3)$ time whether or not a graph is peculiar.*

It is not hard to verify that both elementary graphs and peculiar graphs cannot contain an odd hole or an odd antihole (cf. [28]), which means they are perfect by virtue of Theorem 1. As mentioned before, Chvátal and Sbihi [12] proved that every atom of a clique separator decomposition of any claw-free perfect graph is either elementary or peculiar. This means we can formulate their main result as follows.

Theorem 4 ([12]). *A claw-free graph G with no clique separator is perfect if and only if it is either elementary or peculiar.*

Using the explicit time complexities of the recognition algorithms for elementary and peculiar graphs in Lemma 1 and Lemma 2, we can determine the time complexity of the recognition algorithm for claw-free perfect graphs by Chvátal and Sbihi [12].

Theorem 5 ([12]). *It is possible to decide in $\mathcal{O}(n^4)$ time whether or not a claw-free graph is perfect.*

Proof. Let G be a claw-free perfect graph. To test whether or not G is perfect, we act as follows. First we find a clique separator decomposition $\mathcal{C} = \{G_1, \ldots, G_p\}$ of G, which we can do in $\mathcal{O}(nm)$ time by Theorem 2. Since every atom $G_i \in \mathcal{C}$ is a claw-free graph without a clique separator, G_i is perfect if and only if G_i is elementary or peculiar by Theorem 4. Lemma 1 and Lemma 2 together imply that for each atom G_i we can decide in $\mathcal{O}(n^3)$ time whether G_i is elementary, peculiar, or neither. By Theorem 3, G is perfect if and only if every atom $G_i \in \mathcal{C}$ is perfect. Since we only have to consider at most $n-1$ atoms by Theorem 2, this yields an overall time complexity of $\mathcal{O}(n^4)$. □

Corollary 1. *Let G be a claw-free graph. It is possible to find an odd hole or an odd antihole of G, or conclude that such a subgraph does not exist, in $\mathcal{O}(n^5)$ time.*

Proof. Let G be a claw-free graph. We test whether or not G is perfect, which we can do in $\mathcal{O}(n^4)$ time by Theorem 5. By Theorem 1, G only contains an odd hole or an odd antihole if G is not perfect. In that case, we remove a vertex from G and check in $\mathcal{O}(n^4)$ time if the obtained subgraph G' is perfect. If so, we restore the vertex and repeat the procedure on G, removing another vertex. If not, we repeat the whole procedure on the smaller graph G'. By repeating this procedure as long as possible, we find a minimal imperfect induced subgraph H of G. (A graph is called *minimal imperfect* if it is not perfect, but all its proper induced subgraphs are perfect.) By Theorem 1, H is an odd hole or an odd antihole of G. The $\mathcal{O}(n^5)$ overall time complexity follows from the fact that we have to apply the $\mathcal{O}(n^4)$ time recognition algorithm for claw-free perfect graphs $\mathcal{O}(n)$ times. □

3 Finding induced paths of given parity

In this section we present an algorithm that solves the Odd Induced Path problem in $\mathcal{O}(n^5)$ time for claw-free graphs. We show that, apart from solving the decision problem, it is also possible to *find* an odd induced path between two given vertices of a claw-free graph, or conclude that such a path does not exist, in $\mathcal{O}(n^5)$ time. Here is an outline of our algorithm.

Algorithm solving Odd Induced Path **for claw-free graphs**

```
Input  : a claw-free graph $G$, and vertices $s$ and $t$ of $G$
Output : YES if $G$ contains an odd induced path from $s$ to $t$
         NO otherwise

Preprocess $G$ to obtain graph $G''$
    Step 1: add edges to make $s$ and $t$ simplicial
    Step 2: delete irrelevant vertices
Test whether or not $G''$ is perfect
If $G''$ is not perfect, output YES
If $G''$ is perfect, find a shortest path $P$ from $s$ to $t$
    If $P$ is odd, output YES
    If $P$ is even, define graph $G^* := (V(G'') \cup \{x\}, E(G'') \cup \{sx, tx\})$
    Test whether or not $G^*$ is perfect
        If $G^*$ is not perfect, output YES
        If $G^*$ is perfect, output NO
```

As shown in the outline, we first preprocess the input graph G in order to obtain a new graph G'' with certain desirable properties. This preprocessing procedure is described in Section 3.1. We then distinguish two cases, depending on whether or not G'' is perfect. The case that G'' is not perfect is discussed in Section 3.2, while Section 3.3 deals with the case that G'' is perfect. In Section 3.4 we prove correctness of our algorithm and show that its time complexity is $\mathcal{O}(n^5)$. We also explain in Section 3.4 how our algorithm can be slightly modified in such a way that it also solves the Even Induced Path problem for claw-free graphs in $\mathcal{O}(n^5)$ time.

3.1 Preprocessing the input graph G

Let G be a claw-free graph and let s and t be two vertices of G. Note that we may without loss of generality assume that G is connected and that s and t are not adjacent. We make these assumptions throughout the paper.

Step 1. We add an edge between each pair of non-adjacent neighbors of s, and we do the same for each pair of non-adjacent neighbors of t. Then in the resulting graph G', both s and t are simplicial vertices. In general, adding edges is not a claw-freeness preserving operation. However, the following lemma states that we do not create claws in Step 1.

Lemma 3. *The graph G' is claw-free.*

Proof. Suppose, for contradiction, that G' contains an induced subgraph isomorphic to a claw. Let $K := \{x, a, b, c\}$ be a set of vertices of G' inducing a claw with center x. Note that the fact that s is simplicial implies $x \neq s$. Since G is claw-free, we may without loss of generality assume that at least two vertices of K must be in $N_{G'}[s]$. Since $N_{G'}[s]$ is a clique in G' and $\{a, b, c\}$ is an independent set of G', we may without loss of generality assume that $K \cap N_{G'}[s] = \{x, a\}$ and $\{b, c\} \subseteq V(G') \setminus N_{G'}[s]$. Then $\{x, b, c, s\}$ induces a claw in G with center x, contradicting the claw-freeness of G. $\square$

Step 2. We "clean" G' by repeatedly deleting irrelevant vertices. A vertex $v \in V(G')$ is called *irrelevant* (for vertices s and t) if v does not lie on any induced path from s to t, and we say that G' is *clean* (for s and t) if none of its vertices is irrelevant. Let G'' denote the graph obtained from G' by repeatedly deleting vertices that are irrelevant. Note that G'' is claw-free, as G'' is an induced subgraph of G'.

We now show that we can perform Step 2 in polynomial time by showing that we can identify irrelevant vertices in polynomial time. In general, the problem of deciding whether a vertex is irrelevant is NP-complete. This follows from a result by Derhy and Picouleau [14], who prove that the following problem is NP-complete for the class of graphs of maximum degree at most 3.

Three-in-a-Path
Instance: A graph G and three vertices v_1, v_2, v_3 of G.
Question: Does there exist an induced path of G containing v_1, v_2 and v_3?

Chudnovsky and Seymour [10] study the following closely related problem.

Three-in-a-Tree
Instance: A graph G and three vertices v_1, v_2, v_3 of G.
Question: Does there exist an induced tree of G containing v_1, v_2 and v_3?

Theorem 6 ([10]). *The* Three-in-a-Tree *problem can be solved in $\mathcal{O}(n^4)$ time, and a desired tree can be found in $\mathcal{O}(n^4)$ time in case one exists.*

Observe that the Three-in-a-Path problem is equivalent to the Three-in-a-Tree problem for the class of claw-free graphs, since every induced tree in a claw-free graph is an induced path. Hence, using Theorem 6, we can prove the following result.

Lemma 4. *The problem of deciding whether a vertex v of a claw-free graph G is irrelevant for two simplicial vertices s and t of G can be solved in $\mathcal{O}(n^4)$ time.*

Proof. We claim that there exists an induced path in G from s to t containing v if and only if G together with s, t, v is a YES-instance of the THREE-IN-A-TREE problem. By Theorem 6, this proves that we can decide in $\mathcal{O}(n^4)$ time if v is irrelevant.

If there exists a path in G from s to t containing v, then that path is an induced tree containing all three vertices. Now suppose that there exists an induced subgraph T of G which is a tree containing s, t and v. Recall that any induced subgraph of a claw-free graph which is a tree is in fact an induced path. Since vertices s and t are simplicial, any induced path containing s and t contains exactly one neighbor of s and one neighbor of t. Hence s and t must be the endpoints of the path T. □

After preprocessing the input graph G we have obtained a graph G'' that satisfies the following three conditions:

(1) G'' is claw-free;
(2) both s and t are simplicial vertices of G'';
(3) G'' is clean for s and t.

The following lemma implies that solving the ODD INDUCED PATH problem for G is equivalent to solving the problem for G''. The lemma also shows that the entire preprocessing procedure can be performed in $\mathcal{O}(n^5)$ time.

Lemma 5. *Every induced path from s to t in G'' is also an induced path from s to t in G, and vice versa. Moreover, G'' can be obtained from G in $\mathcal{O}(n^5)$ time.* □

Proof. It is clear that by adding edges in Step 1 no new induced path from s to t is created. Since any induced path from s to t in G contains exactly one vertex of $N_G(s)$ and exactly one vertex of $N_G(t)$, the graph G' obtained after Step 1 contains all induced paths from s to t that were contained in G. In Step 2, we only remove vertices that do not lie on any induced path from s to t. This implies that every induced path from s to t in G'' is also an induced path from s to t in G, and vice versa. It is clear that we can perform Step 1 in $\mathcal{O}(n^2)$ time. In Step 2, we have to check for $\mathcal{O}(n)$ vertices whether or not they are irrelevant. Since we can do this in $\mathcal{O}(n^4)$ time per vertex by Lemma 4, we can perform Step 2, and consequently the entire preprocessing procedure, in $\mathcal{O}(n^5)$ time. □

We now distinguish two cases, depending on whether or not G'' is perfect.

3.2 G'' is not perfect

Suppose G'' is not perfect. Then G'' contains an odd hole or an odd antihole by virtue of the Strong Perfect Graph Theorem. We consider odd antiholes and odd holes in Lemma 6 and Lemma 7, respectively.

Lemma 6. *Let H be a connected claw-free graph. If H contains a simplicial vertex, then H does not contain an odd antihole of length more than 5.*

Proof. Let s be a simplicial vertex of a connected claw-free graph H. For contradiction, suppose H contains an odd antihole X such that $\overline{X} = x_1x_2 \dots x_{2k+1}x_1$ is an odd induced cycle with $k \geq 3$. Vertex s does not belong to X, since s is simplicial. Let P be an induced path from s to a vertex of X such that $|V(P)|$ is minimum. Note that such a path P exists since H is connected. Without loss of generality assume that $V(P) \cap V(X) = \{x_1\}$.

Let s' be the neighbor of x_1 on P. We claim that s' is adjacent to at most one vertex of $\{x_i, x_{i+1}\}$ for $1 \leq i \leq 2k$. If $s' = s$, this claim immediately follows from the assumption that s is simplicial and the fact that x_i and x_{i+1} are not adjacent. Suppose $s' \neq s$, and let s'' be the neighbor of s' on P not equal to x_1. Note that s'' is not adjacent to any vertex of X due to the minimality of $|V(P)|$. Vertex s' cannot be adjacent to both x_i and x_{i+1}, since then the set $\{s', s'', x_i, x_{i+1}\}$ induces a claw in H with center s'. Hence s' is adjacent to at most one vertex of $\{x_i, x_{i+1}\}$ for $1 \leq i \leq 2k$.

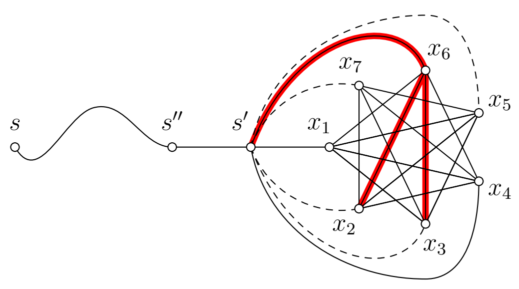

Fig. 3. A claw induced by $\{x_6, s', x_2, x_3\}$ with center x_6.

Note that vertex s' is adjacent to at least one vertex of $\{x_i, x_{i+1}\}$ for $3 \leq i \leq 2k-1$, as otherwise $\{x_1, s', x_i, x_{i+1}\}$ induces a claw in H with center x_1. This, together with the fact that s' is adjacent to at most one vertex of $\{x_i, x_{i+1}\}$ for $1 \leq i \leq 2k$, implies that s' is adjacent to exactly one vertex of $\{x_3, x_{2k}\}$. Without loss of generality, assume that s' is adjacent to x_{2k} and not to x_3. Since s' is adjacent to x_1 and s' is adjacent to at most one vertex of $\{x_i, x_{i+1}\}$ for $1 \leq i \leq 2k$, s' is not adjacent to x_2. Note that x_3 is adjacent to x_{2k}, since $k \geq 3$. But then $\{x_{2k}, s', x_2, x_3\}$ induces a claw in H with center x_{2k}; see Figure 3 for an illustration of the case where $k = 3$. This contradiction finishes the proof of Lemma 6. □

We point out that the arguments in the proof of Lemma 6 can also be used to prove that every odd antihole X of length more than 5 in a connected claw-free graph H is dominating, i.e., every vertex of H either belongs to X or has a neighbor in X.

Lemma 7. *Let H be a connected claw-free graph that is clean for two simplicial vertices s and t. If H contains an odd hole, then there exists both an odd and an even induced path from s to t.*

Proof. Let C be an odd hole of H. Let P be an induced path from s to a vertex p of C and let Q be an induced path from t to a vertex q of C, such that there is no edge in H connecting a vertex in $V(P) \setminus \{p\}$ to a vertex in $V(Q) \setminus \{q\}$ and such that $|V(P)| + |V(Q)|$ is minimum. Note that such paths P and Q exist since H is clean and connected. Let s' be the neighbor of p on P, and let t' be the neighbor of q on Q; we note that possibly $s' = s$ and $t' = t$.

Claim 1. *Both s' and t' are adjacent to exactly two adjacent vertices of C.*

Suppose p is the only vertex of C that is adjacent to s'. Let p^- (respectively p^+) denote the neighbor of p on C when we traverse C in counter-clockwise (respectively clockwise) order. The set $\{p, p^-, p^+, s'\}$ induces a claw in H with center p, contradicting the claw-freeness of H. Hence s' must be adjacent to at least one vertex of $\{p^-, p^+\}$. Suppose there exists a set $D \subseteq V(C)$ such that $|D| \geq 3$ and s' is adjacent to every vertex in D. Since C is an induced cycle, we know that D contains two vertices d_1 and d_2 that are not adjacent. Since s is simplicial and therefore does not have two non-adjacent neighbors, we must have $s' \neq s$. Let $s'' \neq p$ be a neighbor of s' on P; possibly $s'' = s$. Vertex s'' is not adjacent to any vertex of C due to the minimality of $|V(P)| + |V(Q)|$, which means the set $\{s', d_1, d_2, s''\}$ induces a claw in H with center s'. This contradiction finishes the proof of Claim 1 for vertex s'. By symmetry the claim also holds for vertex t'.

We assume, without loss of generality, that $N_H(s') \cap V(C) = \{p, p^+\}$ and $N_H(t') \cap V(C) = \{q, q^+\}$. We distinguish three cases.

Suppose $|\{p, p^+\} \cap \{q, q^+\}| = 0$. Since C is an odd hole, the induced path $s'p^+\overrightarrow{C}qt'$ and the induced path $s'p\overleftarrow{C}q^+t'$ have different parity. Since by definition there is no edge connecting a vertex in $V(P) \setminus \{p\}$ to a vertex in $V(Q) \setminus \{q\}$, this means there exists both an odd and an even induced path from s to t in H; see Figure 4 for an illustration.

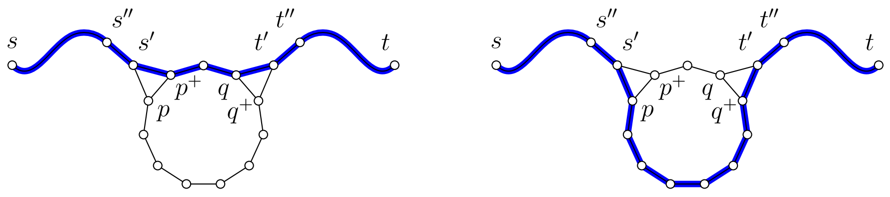

Fig. 4. Two induced paths from s to t of different parity.

Suppose $|\{p, p^+\} \cap \{q, q^+\}| = 1$. Without loss of generality, suppose $p^+ = q$. Then the path $s'qt'$ is an even induced path from s' to t', and the path $s'p\overleftarrow{C}q^+t'$ is an odd induced path from s' to t'. Since by definition there is no edge connecting a vertex in $V(P) \setminus \{p\}$ to a vertex in $V(Q) \setminus \{q\}$, this means there exists both an odd and an even induced path from s to t in H.

Suppose $|\{p, p^+\} \cap \{q, q^+\}| = 2$. By Claim 1, neither s' nor t' is adjacent to p^-. Since s' and t' are not adjacent by the choice of P and Q, the set $\{p, p^-, s', t'\}$ induces a claw in H with center p. This contradiction finishes the proof. □

Recall that G'' is not perfect and has two simplicial vertices s and t. This, together with Lemma 6 and Lemma 7, implies that G'' contains both an odd and an even induced path from s to t. We now show that we can also *find* such paths in $\mathcal{O}(n^5)$ time.

Lemma 8. *If G'' is not perfect, then it is possible to find both an odd and an even induced path from s to t in G'' in $\mathcal{O}(n^5)$ time.*

Proof. Since G'' has two simplicial vertices s and t, G'' does not contain an odd antihole of length more than 5 by Lemma 6. Since an odd antihole of length 5 is also an odd hole of length 5, G'' contains an odd hole by virtue of the Strong Perfect Graph Theorem. We can find such a hole C in $\mathcal{O}(n^5)$ time by Corollary 1. Let c be any vertex of C, and let P be an induced path in G'' from s to t containing c. Note that such a path P exists since G'' is clean for s and t. We can find P in $\mathcal{O}(n^4)$ time as a result Theorem 6. It is clear from the proof of Lemma 7 that we can use P to find both an odd and an even induced path from s to t in G''. □

3.3 G'' is perfect

Suppose G'' is perfect. In the concluding remarks of their paper, Corneil and Fonlupt [13] pointed out that a polynomial-time recognition algorithm for perfect graphs implies a polynomial-time algorithm for the Parity Path problem for the class of perfect graphs. The arguments they used to prove this implication were also mentioned by Hsu [23]. Using their arguments, we can prove the following lemma.

Lemma 9. *If G'' is perfect, then it is possible to find an odd induced path from s to t in G'', or conclude that such a path does not exist, in $\mathcal{O}(n^5)$ time.*

Proof. Let P be a shortest path from s to t in G''. If P has odd length, then we are done. Suppose P has even length. Let G^* be the graph obtained from G by adding a vertex x and edges sx and tx. Note that the graph G^* is claw-free, since s and t are simplicial vertices of G''. We determine whether or not G^* is perfect, which we can do in $\mathcal{O}(n^4)$ time by Theorem 5. If G^* is perfect, then G^* does not contain an odd hole or an odd antihole by virtue of Theorem 1. This means that all induced paths from s to t must be even, so we conclude that there does not exist an odd induced path from s to t. Suppose G^* is not perfect. Then G^* must contain an odd hole or an odd antihole, and vertex x must be in this odd hole or odd antihole since G is perfect. Since x has degree two, G^*

cannot contain an odd antihole. Hence G^* contains an odd hole. We can find an odd hole C of G^* in $\mathcal{O}(n^5)$ time by Corollary 1. The graph obtained from C by removing vertex x is an odd induced path from s to t in G''. □

3.4 Finding induced paths of given parity from s to t in G

We are now ready to prove the main result of this section.

Theorem 7. *The* Odd Induced Path *problem and the* Even Induced Path *problem can each be solved in $\mathcal{O}(n^5)$ time for the class of claw-free graphs. Moreover, an induced path from s to t of given parity can be found in $\mathcal{O}(n^5)$ time, if one exists.*

Proof. Let G be a claw-free graph, and let s and t be two vertices of G. Recall that we may without loss of generality assume that G is connected and that s and t are not adjacent. We preprocess G in $\mathcal{O}(n^5)$ time as described in Section 3.1, thus obtaining a graph G''. Recall that G'' is claw-free, that s and t are simplicial vertices in G'', and that G'' is clean for s and t. We test whether or not G'' is perfect, which we can do in $\mathcal{O}(n^4)$ time by Theorem 5. Below we show that we can find an induced path of given parity from s to t in G'', or conclude that such a path does not exist, in $\mathcal{O}(n^5)$ time. Lemma 5 implies that this suffices to prove Theorem 7.

If G'' is not perfect, then we can find both an odd and an even induced path from s to t in G'' in $\mathcal{O}(n^5)$ time by Lemma 8. If G'' is perfect, then we can find an odd induced path from s to t in G'', or conclude that such a path does not exist, in $\mathcal{O}(n^5)$ time by Lemma 9. In order to find an even induced path from s to t, we define the graph G^* as the graph obtained from G'' by adding the edge st. It is easy to verify that adding the edge st creates neither a claw nor an odd antihole. Hence the arguments used in the proof of Lemma 9 can also be used to find an even induced path from s to t in G'', or conclude that such a path does not exist, in $\mathcal{O}(n^5)$ time. □

Theorem 7 immediately implies the following.

Corollary 2. *Both the* Parity Path *problem and the* Even Pair *problem can be solved in $\mathcal{O}(n^5)$ time for the class of claw-free graphs.*

Bienstock [6] proved that it is NP-complete to decide if a graph contains an odd induced path between every pair of vertices. The following corollary of Theorem 7 implies that this problem can be solved in polynomial time when restricted to the class of claw-free graphs.

Corollary 3. *Deciding whether or not a claw-free graph has an even pair can be done in $\mathcal{O}(n^7)$ time.*

Proof. Let G be a claw-free graph. For each pair s, t of vertices of G, we can check in $\mathcal{O}(n^5)$ whether or not they form an even pair by Corollary 2. Hence we can decide whether or not G has an even pair by performing this check $\mathcal{O}(n^2)$ times, each time with a different pair of vertices of G in the input. □

Another problem Bienstock [6] proved to be NP-complete is the problem of deciding whether a graph contains an odd hole passing through a prescribed vertex. The following corollary, which clearly also holds in case we are looking for an *even* hole, shows that this problem becomes polynomially solvable when restricted to claw-free graphs.

Corollary 4. *It is possible to find an odd hole passing through a prescribed vertex of a claw-free graph, or conclude that such a hole does not exist, in $\mathcal{O}(n^7)$ time.*

Proof. Let G be a claw-free graph and let v be a vertex of G. We can find an odd hole of G passing through v, or conclude that such a hole does not exist, as follows. For each pair s, t of non-adjacent neighbors of v, let $G_{s,t}$ denote the (claw-free) graph obtained from G by removing v and all its neighbors, apart from s and t, from G. Clearly, G contains an odd hole through v if and only if the graph $G_{s,t}$ contains an odd induced path from s to t for some pair of non-adjacent neighbors s, t of v. We can find such a path, or conclude that such a path does not exist, in $\mathcal{O}(n^5)$ time by Theorem 7. The time complexity of $\mathcal{O}(n^7)$ follows from the fact that we have to perform our $\mathcal{O}(n^5)$ algorithm for $\mathcal{O}(n^2)$ pairs of non-adjacent neighbors of v. $\square$

4 Finding shortest induced paths of given parity

In this section we show that it is possible to find a *shortest* induced path of given parity between two specified vertices of a claw-free perfect graph in polynomial time, in case such a path exists. More specifically, we show that we can solve the following two problems in $\mathcal{O}(n^7)$ time for the class of claw-free perfect graphs.

Shortest Odd Induced Path
Instance: A graph G and two vertices s, t of G.
Task: Find a shortest odd induced path from s to t in G, or conclude that such a path does not exist.

Shortest Even Induced Path
Instance: A graph G and two vertices s, t of G.
Task: Find a shortest even induced path from s to t in G, or conclude that such a path does not exist.

Note that a shortest odd induced path between vertices s and t of a graph G is not necessarily a shortest odd path between s and t in G. For example, the shortest odd induced path from s to t in the graph in Figure 5 has length 5, whereas the shortest odd path from s to t has length 3.

Unlike the results in the previous section, we do not rely on the recognition algorithm for claw-free perfect graphs that was described in Section 2 to prove the main result of this section. Instead, we make use of the structural properties of claw-free perfect graphs that were presented by Chvátal and Sbihi in [12]. Recall that they showed that a claw-free perfect graph with no clique separator is

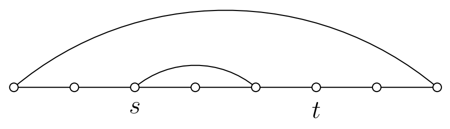

Fig. 5. Shortest odd path from s to t is not shortest odd induced path.

either elementary or peculiar (see Theorem 4). We first show in Section 4.1 that both the Shortest Odd Induced Path problem and the Shortest Even Induced Path problem can be solved in $\mathcal{O}(n^4)$ time for elementary graphs, and in $\mathcal{O}(n^3)$ time for peculiar graphs. In Section 4.2 we then present in detail Tarjan's clique separator decomposition algorithm that was already mentioned in Section 2. Finally, we prove in Section 4.3 that the Shortest Odd Induced Path and Shortest Even Induced Path problems can be solved in $\mathcal{O}(n^7)$ time for the class of claw-free perfect graphs.

4.1 Shortest induced paths in elementary and peculiar graphs

Let us start by showing how to find shortest induced paths of given parity in elementary graphs. Recall that a graph is elementary if and only if its edges can be colored with two colors such that every induced P_3 has both its edges colored differently.

Lemma 10. *Both the* Shortest Odd Induced Path *problem and the* Shortest Even Induced Path *problem can be solved in $\mathcal{O}(n^4)$ time for the class of elementary graphs.*

Proof. Let H be an elementary graph, and let u and v be two vertices of H. Note that we may assume, without loss of generality, that H is connected and that u and v are not adjacent. Suppose u and v have a common neighbor w. The even induced path uwv is the only induced path from u to v that contains w; in particular, w cannot lie on an odd induced path from u to v. Hence we may assume that u and v do not have a common neighbor.

We observe that any induced path from u to v in H contains exactly one vertex from $N_H(u)$ and exactly one vertex from $N_H(v)$. We also observe that in any elementary coloring of H, any two consecutive edges of any induced path will be colored differently. Hence if there exists an odd (respectively even) induced path from u to v, then the first and the last edge of that path have the same color (respectively different colors). Using these observations, we can find a shortest odd induced path from u to v in H as follows.

We first find an elementary coloring $\varphi : E(H) \to \{0, 1\}$ of H; we can find such a coloring φ in $\mathcal{O}(n^3)$ time by Lemma 1. For every pair $u' \in N_H(u)$ and $v' \in N_H(v)$ with $\varphi(uu') = \varphi(vv')$, we define $H_{u'v'}$ to be the graph obtained from H by deleting the set $(N_H[u] \cup N_H[v]) \setminus \{u', v'\}$. Note that $H_{u'v'}$ is well-defined, since u and v are not adjacent and have no common neighbors. We either find a shortest path P' from u' to v' in $H_{u'v'}$, or conclude that such a path does not exist. It is well-known that we can do this in $\mathcal{O}(n^2)$ time. If there exists a shortest path P' from u' to v', then this path P' is clearly an induced path in $H_{u'v'}$. We add the vertices u and v as well as the edges uu' and vv' to P', which yields an induced path P from u to v in H. Since P is induced and φ is an elementary coloring, the colors 0 and 1 alternate on P. Then P is an odd induced path from u to v in H, since $\varphi(uu') = \varphi(vv')$. By performing this procedure for all pairs u', v' with $\varphi(uu') = \varphi(vv')$, we either find a shortest odd induced path from u to v in H, or conclude that such a path does not exist. It is clear that the procedure can be executed in $\mathcal{O}(n^4)$ time.

To solve Shortest Even Induced Path, we perform the above procedure for all pairs u', v' with $\varphi(uu') \neq \varphi(vv')$ instead of $\varphi(uu') = \varphi(vv')$. □

It is clear from Definition 2 that the vertex set of every peculiar graph can be partitioned into nine disjoint cliques. Since every induced path contains at most two vertices of any clique, this immediately implies that every peculiar graph is P_{19}-free. A more careful analysis of the definition of a peculiar graph yields the following result.

Lemma 11. *Every peculiar graph is P_6-free but not P_5-free.*

Proof. Let H be a peculiar graph, and let A_i, B_i, D_i ($i = 1, 2, 3$) be a partition of $V(H)$ as mentioned in Definition 2. The set $V(H)$ can be partitioned into three cliques, namely $X_1 := A_2 \cup B_1 \cup B_2 \cup D_3$, $X_2 := D_1$ and $X_3 := A_1 \cup A_3 \cup B_3 \cup D_2$. This immediately implies that H is P_7-free, as any induced path in H contains at most two vertices of any clique. The P_6-freeness of H follows from the observation that for every pair $x, y \in X_2$ we have $N_H[x] = N_H[y]$, which implies that any induced path in H containing vertices of $X_1 \cup X_3$ can only contain at most one vertex from X_2.

Let $a_2 \in A_2$ and $b_3 \in B_3$ be a pair of non-adjacent vertices of H; note that such a pair exists by Definition 2. Since none of the sets D_1, D_2, D_3 is empty, H contains an induced path $d_3a_2d_1b_3d_2$, where $d_i \in D_i$, $i = 1, 2, 3$ (see also Figure 2). □

The observation that any induced path from s to t in a P_6-free graph H contains at most three other vertices of H immediately implies the following result.

Lemma 12. *Both the* Shortest Odd Induced Path *problem and the* Shortest Even Induced Path *problem can be solved in $\mathcal{O}(n^3)$ time for the class of P_6-free graphs.*

Lemma 11 and Lemma 12 together immediately yield the following.

Corollary 5. *Both the* Shortest Odd Induced Path *problem and the* Shortest Even Induced Path *problem can be solved in $\mathcal{O}(n^3)$ time for the class of peculiar graphs.*

4.2 A closer look at Tarjan's decomposition algorithm

We now take a closer look at Tarjan's [36] decomposition algorithm, mentioned in Section 2. Tarjan's algorithm runs in $\mathcal{O}(mn)$ time and produces a clique separator decomposition of a graph with at most $n-1$ atoms; see also Theorem 2. We prove some properties of the clique separator decomposition obtained by this algorithm, and use those properties in the proof of Theorem 8 in Section 4.3. We first introduce some additional terminology and describe an algorithm, called the Elimination Game, which is used as a subroutine in Tarjan's decomposition algorithm. The Elimination Game was first described by Parter [30] in 1961.

A graph is *chordal* if it does not contain an induced cycle of length at least 4. If a graph G is a subgraph of a chordal graph H, then H is called a *triangulation* of G. A triangulation H of a graph G is called *minimal* if none of the proper subgraphs of H is a triangulation of G.

Consider the following algorithm, known as the *Elimination Game*: given a graph G and an ordering $\pi = v_1, \ldots, v_{|V(G)|}$ of the vertices of G, repeatedly choose a vertex v_i with the lowest index, add edges in order to make the neighborhood of v_i into a clique, and remove v_i from the graph. The output G^+_π of the Elimination Game is a triangulation of the input graph G, and the set F_π of edges that are added during the Elimination Game are called *fill edges*. Note that $G^+_\pi = (V(G), E(G) \cup F_\pi)$.

The total number of fill edges depends on the order π in which the vertices are considered. If the number of fill edges is 0, then the order in which the vertices were considered is called a *perfect elimination ordering* of G. It is well-known that a graph has a perfect elimination ordering if and only if it is chordal [20]. An ordering π is called a *minimal elimination ordering* if G^+_π is a minimal triangulation of G. A minimal elimination ordering and a minimal triangulation of a graph can be found in $\mathcal{O}(mn)$ time, for example using an algorithm by Rose, Tarjan and Lueker [32].

Tarjan's clique separator decomposition algorithm takes as input a connected graph G, and starts by finding a minimal elimination ordering π of the vertices of G. The algorithm then calculates G^+_π by running the Elimination Game on G and π. For each vertex v of G, the algorithm then computes $X(v) := \{w \mid \pi(v) > \pi(w) \text{ and } vw \in E(G) \cup F_\pi\}$, i.e., the set of neighbors of v in the graph G^+_π that appear after v in the ordering π, where π is interpreted as a bijection from $V(G)$ to $\{1, \ldots, |V(G)|\}$. The algorithm repeats the following decomposition step for each vertex v in increasing order with respect to π.

Decomposition Step. Let A be the vertex set of the connected component of $G - X(v)$ containing v, and let $B := V(G) - (X(v) \cup A)$. If $X(v)$ is a clique of G and $B \neq \emptyset$, then decompose G into $G_A := G[A \cup X(v)]$ and $G_B := G[B \cup X(v)]$, and replace G by G_B.

If the set $X(v)$ in a decomposition step satisfies both conditions, i.e., $X(v)$ is a clique of G and $B \neq \emptyset$, then $X(v)$ is a clique separator, and the decomposition step is called *successful*. We observe the following. Firstly, G is only decomposed into G_A and G_B in successful decomposition steps. Secondly, because the number of vertices of G decreases in every successful decomposition step due to the operation that replaces G by G_B, a clique separator found in a successful decomposition step is a clique separator of a subgraph of G. However, it is not hard to see that every such clique separator is also a clique separator of the original input graph G; we refer to Gavril [21] for an explicit proof of this statement.

When Tarjan's algorithm is run on a graph G that is clean for two vertices s and t, then we can prove another property of a clique separator found in a successful decomposition step.

Lemma 13. *Let X be a clique separator of a graph G that is clean for two vertices s and t. Then $G-X$ consists of exactly two connected components, one containing s and the other containing t.*

Proof. By definition, $G-X$ has at least two connected components. We first show that s and t cannot belong to the same connected component. Suppose, for contradiction, that s and t belong to the same connected component D of $G-X$. Let D' be another connected component of $G-X$, and let $d' \in D'$. Because G is clean for s and t, there exists an induced path P from s to t containing d'. Because X is a clique separator of G and $D' \neq D$, both the path $s\overrightarrow{P}d'$ and the path $d'\overrightarrow{P}t$ must contain a vertex of X. However, then P is not an induced path, because X is a clique. This contradiction shows that d' is not contained in any induced path from s to t. By definition, this means that d' is irrelevant, contradicting the assumption that G is clean. Hence s and t must belong to two different connected components D_1 and D_2 of $G-X$, respectively.

Now suppose $G-X$ has another connected component D_3, and let d be a vertex of D_3. Since s, t and d are contained in three different connected components of $G-X$ and X is a clique of G, there exists no induced path from s to t containing d. This means that d is irrelevant, contradicting the assumption that G is clean. We conclude that $G-X$ consists of exactly two connected components, one containing s and the other containing t. □

Tarjan [36] showed that in every successful decomposition step the graph G_A is an atom of the input graph G. We say that the corresponding clique separator *created* G_A. The graph G_B in the last successful decomposition step is an atom of G. Hence, in this case, the corresponding clique separator created two atoms. Summarizing, in every successful decomposition step, a new atom of G is created, and in the last successful decomposition step, two new atoms of G are created.

We now describe a procedure that allows us to define an ordering of the atoms of a graph that is clean for a certain pair of vertices.

Let G be a graph that is clean for two vertices s and t. Let G_A be an atom of G created by a clique separator $X(v)$ found in a successful decomposition step. We define $G_i := G_A$, where the index i is determined as follows. First of all, we may assume that $i \in \{1, \dots, n-1\}$, because G has at most $n-1$ atoms due to Theorem 2. Now, by Lemma 13, $G - X(v)$ contains exactly two connected components D_1 and D_2, where $s \in V(D_1)$ and $t \in V(D_2)$. Note that $v \notin X(v)$, which means that v belongs to either D_1 or D_2. If v belongs to D_1, then we choose i to be the smallest integer from $\{1, \ldots, n-1\}$ that has not yet been used. Otherwise we choose i to be the largest integer from $\{1, \ldots, n-1\}$ that has not yet been used. We repeat this procedure for each atom G_A created in a successful decomposition step. Recall that the graph G_B in the last successful decomposition step is an atom of G, and we define $G_i := G_B$, where i is the smallest integer from $\{1, \ldots, n-1\}$ that has not yet been used.

The procedure described above yields a clique separator decomposition $\mathcal{C}$ of G, where $\mathcal{C} := \{G_1, \ldots, G_k, G_\ell, \ldots, G_{n-1}\}$ for some $k < \ell$. For convenience, we relabel the atoms in such a way that the atoms have consecutive indices, i.e., such that $\mathcal{C} := \{G_1, \ldots, G_p\}$, where $p = k + n - \ell$. By our procedure, s belongs to G_1 and t belongs to G_p. We call a clique separator decomposition $\mathcal{C}$ obtained from Tarjan's algorithm with indices as defined above an *ordered* clique separator decomposition (with respect to s and t). We say that $\mathcal{C}$ is *nontrivial* if $\mathcal{C}$ contains at least two atoms. In that case $p \geq 2$, and we can define $X_{i,i+1} := V(G_i) \cap V(G_{i+1})$ for $i = 1, \ldots, p-1$. We call $\mathcal{X} = \{X_{1,2}, \ldots, X_{p-1,p}\}$ the *intersection set* of $\mathcal{C}$.

Lemma 14. *Let $\mathcal{X}$ be the intersection set of a nontrivial ordered clique separator decomposition $\mathcal{C}$ of a graph G that is clean for two vertices s and t. Then every set in $\mathcal{X}$ is a clique separator of G.*

Proof. Suppose $\mathcal{X} = \{X_{1,2}, X_{2,3}, \ldots, X_{p-1,p}\}$ for some $p \geq 2$. Let $1 \leq i \leq p-1$. We show that $X_{i,i+1}$ is a clique separator of G.

Recall that $X_{i,i+1} = V(G_i) \cap V(G_{i+1})$. Let $X(v)$ be the clique separator that created G_i, and let $X(v')$ be the clique separator that created G_{i+1}. If $v = v'$, then $X(v) = X(v')$, and G_i and G_{i+1} are the last two atoms created by Tarjan's algorithm. Consequently, $X_{i,i+1} = X(v)$, and the statement of the lemma is satisfied.

Now suppose $v \neq v'$. Then one of the two atoms is created before the other. We first assume that G_i was created before G_{i+1}. By Lemma 13, the graph $G - X(v)$ consists of two connected components D_1 and D_2 with $s \in V(D_1)$ and $t \in V(D_2)$.

First suppose $V(G_i) \backslash X(v)$ belongs to D_1. We need the following claim.

Claim 1. $X_{i,i+1} \subseteq X(v)$.

We prove Claim 1 as follows. Because G_i is created before G_{i+1}, and $V(G_i) \backslash X(v)$ belongs to D_1, we find that $V(G_{i+1}) \backslash X(v)$ belongs to $V(D_2)$. This means that $(V(G_i) \backslash X(v)) \cap (V(G_{i+1}) \backslash X(v)) = \emptyset$. Consequently, $X_{i,i+1} \subseteq X(v)$, and Claim 1 is proven.

By Lemma 13, the graph $G - X(v')$ consists of two connected components D'_1 and D'_2 with $s \in V(D'_1)$ and $t \in V(D'_2)$. Because G_i is created before G_{i+1}, and $V(G_i) \backslash X(v)$ belongs to D_1, we deduce that $V(D_1) \subseteq V(D'_1)$ and $V(D'_2) \subseteq V(D_2) \cup X(v)$. Let $S = V(D'_1) \backslash V(D_1)$ and $S^* = S \cap X(v)$. We observe that $S \neq \emptyset$ by description of Tarjan's algorithm. Let $T = V(D'_2)$ and let $T^* = T \cap X(v)$. We consider four cases.

Case 1. $S^* \neq \emptyset$ and $T^* \neq \emptyset$.
Then there exists a vertex $s \in S^*$ and a vertex $t \in T^*$. Because s and t belong to the clique $X(v)$, we find that s and t are adjacent. However, s and t are in two different connected components D_1' and D_2' of $G - X(v')$. Hence, Case 1 is not possible.

Case 2. $S^* = T^* = \emptyset$.
Then $X(v) \subseteq X(v') \subset V(G_{i+1})$. Because $X(v) \subset V(G_i)$, this means that $X(v) \subseteq V(G_i) \cap V(G_{i+1}) = X_{i,i+1}$. By this and Claim 1, we then find that $X_{i,i+1} = X(v)$, and the statement of the lemma is true.

Case 3. $S^* = \emptyset$ and $T^* \neq \emptyset$.
Then s and S are in different connected components of $G - X(v')$. This contradicts to the fact that the vertices of S and s belong to the same connected component of $G - X(v')$, namely connected component D_1'. Hence, Case 3 is not possible.

Case 4. $S^* \neq \emptyset$ and $T^* = \emptyset$.
First suppose $V(G_{i+1}) \backslash X(v')$ belongs to $T = V(D_2')$, which contains t. Then the algorithm has chosen the largest available index from $\{1, \ldots, n-1\}$ for G_{i+1}. Because $V(G_i) \backslash X(v) \subset V(D_1)$ and $s \in V(D_1)$, the algorithm has chosen the smallest available index from $\{1, \ldots, n-1\}$ for G_i. Hence, the algorithm used all available $n-1$ indices. However, it still needs to process and index a subgraph of $G[S \cup X(v')]$. This is not possible (recall that, according to Theorem 2, G has at most $n-1$ atoms, and hence the set of indices $\{1, \ldots, n-1\}$ is large enough). This means that $V(G_{i+1}) \backslash X(v')$ does not belong to T. As a result, $V(G_{i+1}) \backslash X(v')$ belongs to S.

Suppose S^* contains a vertex $w \notin V(G_{i+1}) \backslash X(v')$. Because $S^* \cap X(v') = \emptyset$, we then find that $w \notin V(G_{i+1})$. By definition of S^*, we have $w \in X(v) \cap S \subset V(G_i) \cap S$.

We observe the following. After G_i was created the algorithm continued with a subgraph G_B of $G[V(D_2) \cup X(v)]$ that contains $X(v)$, because $G(V_i) \backslash X(v)$ belongs to D_1. Because $w \in X(v)$, we find that $w \in V(G_B)$. This means that w is in an atom G_j that is created after G_i.

After G_{i+1} was created, the algorithm continued with a subgraph $G_{B'}$ of $G[V(D_2') \cup X(v')]$, because $G(V_{i+1}) \backslash X(v')$ belongs to $S \subset V(D_1')$. Because $w \in S$ and $S \cap (V(D_2') \cup X(v')) = \emptyset$, we find that $w \notin V(G_{B'})$. Hence, G_j must have been created before G_{i+1}.

Let G_j be created by $X(v'')$. By Lemma 13, the graph $G - X(v'')$ consists of two connected components D_1'' and D_2'' with $s \in V(D_1'')$ and $t \in V(D_2'')$. Because G_j was created after G_i but before G_{i+1}, the algorithm would have chosen index $i+1$ as the smallest available index for G_j if $V(G_j) \backslash X(v'')$ was in D_1''. Hence, $V(G_j) \backslash X(v'')$ must belong to D_2''. Then, after creating G_j the algorithm continued with a subgraph $G_{B''}$ of $G[V(D_1'') \cup X(v'')]$. We note that $G_{B''}$ contains G_{i+1}, because G_{i+1} is created after G_j.

Recall that $w \in S \subset V(D_1')$. Hence, a path Q from w to t must contain a vertex from $X(v')$. Because $V(G_j) \backslash X(v'')$ belongs to $V(D_2'')$, there exists a path from any vertex in $V(G_j)\backslash X(v'')$ to t that uses no vertex from $X(v')$. Hence $w \in X(v'')$, and the set $V(G_j) \cap X(v')$ is a clique separator of G_j. This is not possible, because G_j is an atom. From this contradiction, we conclude that all vertices in S^* belong to $V(G_{i+1})\backslash X(v')$. Because $T^* = \emptyset$, we find that $X(v) \subset V(G_{i+1})$. Because $X(v) \subset V(G_i)$, this means that $X(v) \subseteq V(G_i) \cap V(G_{i+1}) = X_{i,i+1}$. By this and Claim 1, we then find that $X_{i,i+1} = X(v)$, and the statement of the lemma is true.

We can use the same arguments if $V(G_i)\backslash X(v)$ belongs to D_2. Finally, we can also use the same arguments if G_i was created after G_{i+1}. This finishes the proof of Lemma 14. ◻

We say that a path P in a graph $G = (V, E)$ *passes through* a set $X \subseteq V$ if P contains a vertex of X. If P contains k mutually vertex-disjoint subpaths Q_i with $V(Q_i) \subseteq X$ for $i = 1, \ldots, k$ such that there is no edge $uv \in E(P)$ between any two vertices u, v that are end vertices of two different subpaths Q_i, Q_j, then we say that P *passes through* X k *times.* We use this terminology in the lemma below.

Lemma 15. *Let* $\mathcal{C} := \{G_1, \ldots, G_p\}$ *be an ordered clique separator decomposition of a graph* G *that is clean for two vertices* s *and* t*. Then every induced path in* G *from* s *to* t *passes through each of the atoms* $G_1, \ldots, G_p$ *exactly once, and passes through them in increasing order, i.e., passes through* G_i *before passing through* G_j*, for every* $1 \leq i < j \leq p$.

Proof. Let P be an induced path in G from s to t. We observe that P can only pass each atom at most once; if P passes an atom G_i twice, then it must have passed through a clique separator twice. Consequently, P would not be induced. It remains to prove that P visits each atom in order of increasing index.

Because $s \in V(G_1)$, we find that P passes through atom G_1 first. Suppose P does not visit each atom in order of increasing index. Let G_i be the first atom that P "skips". Let G_j be the first atom that P passes through after leaving atom G_{i-1}. Because P passes through every atom at most once, we find that $j > i$. Recall that $X_{i,i+1} = V(G_i) \cap V(G_{i+1})$ is a clique separator due to Lemma 14. By Lemma 13, vertices s and t are in two different connected components of $G - X_{i,i+1}$. Hence P contains a vertex from $X_{i,i+1}$. Because $X_{i,i+1} \subset V(G_i)$, this means that P passes through G_i at least twice. We already deduced that this is not possible. This finishes the proof of Lemma 15. ◻

4.3 Shortest induced paths in claw-free perfect graphs

We are now ready to prove the main result of this section.

Theorem 8. *The* Shortest Odd Induced Path *problem and the* Shortest Even Induced Path *problem can each be solved in* $\mathcal{O}(n^7)$ *time for the class of claw-free perfect graphs.*

Proof. In order to prove Theorem 8, we present an algorithm that solves both the SHORTEST ODD INDUCED PATH problem and the SHORTEST EVEN INDUCED PATH problem on claw-free perfect graphs in $\mathcal{O}(n^7)$ time. The algorithm takes as input a claw-free perfect graph and two of its vertices s and t. We first preprocess this input graph by performing the two steps of the preprocessing procedure described in Section 3.1. This way we obtain a claw-free graph G, such that s and t are simplicial vertices of G, and G is clean for s and t. The preprocessing phase can be done in $\mathcal{O}(n^5)$ time by Lemma 5. As a result of Lemma 5, in order to prove Theorem 8 it suffices to show that we can solve the two problems on G in $\mathcal{O}(n^7)$ time.

Next we find an ordered clique separator decomposition $\mathcal{C} := \{G_1, \ldots, G_p\}$ of G with at most $n-1$ atoms using Tarjan's decomposition algorithm described in Section 4.2. We can find such a clique separator decomposition in $\mathcal{O}(nm)$ time by Theorem 2.

We first prove the following claim.

CLAIM 1. *Let $G_i \in \mathcal{C}$. We can solve* SHORTEST ODD INDUCED PATH *and* SHORTEST EVEN INDUCED PATH *in $\mathcal{O}(n^4)$ time for any induced subgraph of G_i.*

Let G' be an induced subgraph of one of the atoms $G_i \in \mathcal{C}$. The graph G_i is a claw-free perfect graph without a clique separator, so G_i is either elementary or peculiar by Theorem 4. By Lemma 1 and Lemma 2 we can decide in $\mathcal{O}(n^3)$ time whether G_i is elementary or peculiar. If G_i is peculiar, then G_i is P_6-free by Lemma 11. Since every induced subgraph of an elementary (respectively P_6-free) graph is elementary (respectively P_6-free), we can solve SHORTEST ODD INDUCED PATH and SHORTEST EVEN INDUCED PATH for the graph G' in $\mathcal{O}(n^4)$ time as a result of Lemma 10 and Lemma 12, respectively. This finishes the proof of Claim 1.

We observe that $\mathcal{C}$ is nontrivial, because $N_G(s)$ is a clique separator of G. Then the intersection set $\mathcal{X} = \{X_{1,2}, \ldots, X_{p-1,p}\}$ of $\mathcal{C}$ exists. Recall that every $X_{i,i+1}$ in $\mathcal{X}$ is a clique separator due to Lemma 14. Also recall that by our indexing procedure s belongs to G_1 and t belongs to G_p. Because G does not contain irrelevant vertices, we have $s \in V(G_1) \setminus X_{1,2}$ and $t \in V(G_p) \setminus X_{p-1,p}$.

Note that, in general, a set $X_{i-1,i}$ might share vertices with the set $X_{i,i+1}$, and possibly with other sets in $\mathcal{X}$. Let us for the moment assume that this is not the case, i.e., that the sets in $\mathcal{X}$ are pairwise disjoint. At the very end of this proof we will explain why we can make this assumption without loss of generality. Define $X_{0,1} := \emptyset$, $X_{p,p+1} := \emptyset$ and $V_i := V(G_i) \setminus (X_{i-1,i} \cup X_{i,i+1})$ for $i = 1, \ldots, p$. Let $W_i := V_1 \cup \cdots \cup V_i \cup X_{1,2} \cup \cdots \cup X_{i-1,i} = V(G_1) \cup \ldots \cup V(G_{i-1}) \cup (V(G_i) \setminus X_{i,i+1})$ for $i = 1, \ldots, p$. See Figure 6 for a schematic representation of graph G with respect to the clique separator decomposition $\mathcal{C}$ (under the assumption that the sets in $\mathcal{X}$ are pairwise disjoint).

We observe that any induced path from s to t contains either one vertex or two vertices of each $X_{i,i+1}$, since each set $X_{i,i+1}$ is a clique. We now restrict our attention to the graph G_1. We claim that all vertices of G_1 belong to the closed neighborhood of s. Suppose there is a vertex $v \in V(G_1) \setminus N_G[s]$. Then $N_G(s)$ is a

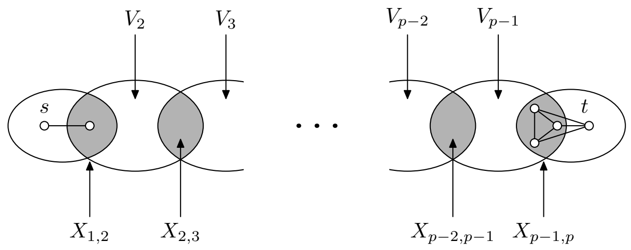

Fig. 6. Structure of the graph G with respect to the clique separator decomposition $\mathcal{C}$.

clique separator of G_1, contradicting the assumption that G_1 is an atom. Hence we know that for every vertex $x \in X_{1,2}$, there exists only one induced path from s to x, and it has length 1.

In order to find a shortest odd and a shortest even induced path from s to any vertex in $X_{p-1,p}$, we run the following algorithm for increasing $i = 2, \ldots, p-1$. For each vertex v in $X_{i,i+1}$ we perform the following two steps. We state the two steps first, before we describe how to perform them below.

Step 1. Find a shortest odd induced path from s to v in $G[W_i \cup \{v\}]$, or conclude that such a path does not exist, and find a shortest even induced path from s to v in $G[W_i \cup \{v\}]$, or conclude that such a path does not exist.

Step 2. For each $v' \in X_{i,i+1} \setminus \{v\}$, find a shortest odd induced path from s to v in $G[W_i \cup \{v, v'\}]$ using edge $v'v$, or conclude that such a path does not exist, and find a shortest even induced path from s to v in $G[W_i \cup \{v, v'\}]$ using edge $v'v$, or conclude that such a path does not exist.

To execute Step 1, we act as follows. For all $u \in X_{i-1,i}$, we find a shortest odd (even) induced path from u to v in the graph $G' := G[V_i \cup \{u, v\}]$, or conclude that such a path does not exist. Since G' is an induced subgraph of G_i, we can find a shortest odd (even) induced path from u to v in G' in $\mathcal{O}(n^4)$ time as a result of Claim 1. Combining those shortest induced paths of both parities with the shortest induced paths of both parities from s to u in $G[W_{i-1} \cup \{u\}]$ yields at most four induced paths from s to v in $G[W_i \cup \{v\}]$. To check whether there exists a shorter odd or even induced path from s to v, using *two* vertices of $X_{i-1,i}$, we act as follows. For each $u' \in X_{i-1,i} \setminus \{u\}$, we find a shortest odd (even) induced path from u to v in the graph $G[(V_i \setminus N_G(u')) \cup \{u, v\}]$. We combine those paths of both parities with the shortest induced paths of both parities from s to u, using edge $u'u$, in the graph $G[W_{i-1} \cup \{u, u'\}]$. This way we are guaranteed to find both a shortest odd and a shortest even induced path from s to v in $G[W_i \cup \{v\}]$, unless one of those paths does not exist. For step 2 we perform similar checks in $\mathcal{O}(n^4)$ time.

After we have completed both steps for $i = p-1$, we have found (if they exist) shortest odd and shortest even induced paths from s to every vertex in

$X_{p-1,p}$, both paths using one and paths using two vertices of $X_{p-1,p}$. Recall that $V(G_1) \subseteq N_G[s]$. Similarly, we have $V(G_p) \subseteq N_G[t]$, which means that there is exactly one induced path from any vertex in $X_{p-1,p}$ to t, and it has length 1. This way we find a shortest odd and a shortest even induced path from s to t, or conclude that such a path does not exist. Because we assume that the sets in $\mathcal{X}$ are pairwise disjoint, we have $\sum_{i=1}^{p-1} |X_{i,i+1}| \leq n$. This means that we only have to perform the $\mathcal{O}(n^4)$ procedure for finding shortest induced paths on $\mathcal{O}(n^3)$ induced subgraphs of G. Hence the overall time complexity is $\mathcal{O}(n^7)$.

What remains is to argue why we may assume that the sets in $\mathcal{X}$ are pairwise disjoint. Suppose that the algorithm has processed atoms $G_1, \ldots, G_{i-1}$, and is about to process atom G_i. The algorithm has found the shortest odd (even) induced paths from s to every vertex in $X_{i-1,i}$, using either one or two vertices of $X_{i-1,i}$. After processing atom G_i, the algorithm has extended those paths to shortest odd (even) induced paths from s to every vertex in $X_{i,i+1}$, using either one or two vertices of $X_{i,i+1}$. Suppose $X_{i-1,i}$ and $X_{i,i+1}$ overlap, and let x be in $X_{i-1,i} \cap X_{i,i+1}$. As a result of Lemma 15 and the observation that any induced path can pass through a clique only once, the paths found for x just before the algorithm starts processing atom G_i are exactly the same as the paths found after G_i is processed. In other words, we do not have to perform Steps 1 and 2 for any vertex in $X_{i-1,i} \cap X_{i,i+1}$, since it will make no difference to the final solution. That means we can redefine $X_{i,i+1}$ as follows: $X_{i,i+1} := X_{i,i+1} \setminus X_{i-1,i}$. Similar arguments imply that the same holds for any other set $X_{j,j+1} \in \mathcal{X}$ with $j > i$ that overlaps with $X_{i-1,i}$. Hence, after redefining the sets in $\mathcal{X}$ we can run the same algorithm. This completes the proof of Theorem 8. □

5 Conclusions and open problems

We have proved that both the Odd Induced Path problem and the Even Induced Path problem, and consequently the Parity Path problem, can be solved in $\mathcal{O}(n^5)$ time for the class of claw-free graphs. This immediately implies that we can also decide in polynomial time whether a claw-free graph contains an odd induced path between *every* pair of vertices. We also showed how we can find a shortest induced path of given parity between two specified vertices of a claw-free perfect graph in $\mathcal{O}(n^7)$ time. Does there exist a polynomial-time algorithm for the Shortest Odd Induced Path and Shortest Even Induced Path problems for *general* claw-free graphs? Another interesting question is whether or not there exists a polynomial-time algorithm for the Odd Induced Path and Even Induced Path problems for the class of planar graphs.

One of Bienstock's [6] NP-complete problems is to decide whether a graph contains an odd hole passing through a given vertex. We showed that this problem, as well as the variant where we want to find an even hole through a given vertex, can be solved in $\mathcal{O}(n^7)$ time for the class of claw-free graphs. Recently, Shrem et al. [35] obtained a polynomial-time algorithm for detecting a shortest odd hole in a claw-free graph. There are a number of problems in the literature related to finding holes in graphs. Checking if a graph has no hole is equivalent to deciding if the graph is chordal. It is well-known that this problem can be solved in linear time [32]. An interesting related problem is to decide if a graph has an *odd* hole. The computational complexity of this problem remains open, even though a seemingly similar problem —deciding if a graph has an *even* hole— can be solved in polynomial time [8].

Acknowledgements. The authors would like to thank Jiří Fiala, Benjamin Lévêque, Luke Mathieson, Nicolas Trotignon and two anonymous reviewers for helpful suggestions.

References

1. E.M. Arkin, C.H. Papadimitriou, and M. Yannakakis. Modularity of cycles and paths in graphs. *Journal of the ACM*, 38(2):255–274, 1991.
2. S.R. Arikati and U.N. Peled. A linear algorithm for the group path problem on chordal graphs. *Discrete Applied Mathematics*, 44(1-3):185–190, 1993.
3. S.R. Arikati and U.N. Peled. A polynomial algorithm for the parity path problem on perfectly orientable graphs. *Discrete Applied Mathematics*, 65(1):5–20, 1996.
4. S.R. Arikati, C.P. Rangan, and G.K. Manacher. Efficient reduction for path problems on circular-arc graphs. *BIT*, 31(2):182–193, 1991.
5. C. Berge. Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind. *Wissenschaftliche Zeitschrift der Martin-Luther-Universität Halle-Wittenberg, Mathematisch-Naturwissenschaftliche Reihe*, 10: 114, 1961.
6. D. Bienstock. On the complexity of testing for odd holes and induced odd paths. *Discrete Mathematics*, 90(1):85–92, 1991.
7. M. Chudnovsky, G. Cornuéjols, X. Liu, P.D. Seymour and K. Vušković. Recognizing Berge Graphs. *Combinatorica*, 25(2):143–186, 2005.
8. M. Chudnovsky, K. Kawarabayashi and P.D. Seymour. Detecting even holes. *Journal of Graph Theory*, 48(2):85–111, 2005.
9. M. Chudnovsky, N. Robertson, P.D. Seymour, and R. Thomas. The strong perfect graph theorem. *Annals of Mathematics*, 164:51–229, 2006.
10. M. Chudnovsky and P.D. Seymour. The three-in-a-tree problem. *Combinatorica*, to appear, manuscript at `www.columbia.edu/∼mc2775/threeinatree.pdf`.
11. M. Chudnovsky and P.D. Seymour. Three-colourable perfect graphs without even pairs. Submitted, manuscript at `www.columbia.edu/∼mc2775/K4evenpairs.ps`.
12. V. Chvátal and N. Sbihi. Recognizing claw-free perfect graphs. *Journal of Combinatorial Theory, Series B*, 44:154–176, 1988.
13. D.G. Corneil and J. Fonlupt. Stable set bonding in perfect graphs and parity graphs. *Journal of Combinatorial Theory, Series B*, 59:1–14, 1993.
14. N. Derhy and C. Picouleau. Finding induced trees. *Discrete Applied Mathematics*, 157: 3552–3557, 2009.
15. R. Diestel. *Graph Theory.* (3rd Edition). Springer-Verlag Heidelberg, 2005.
16. H. Everett, C.M.H. de Figueiredo, C.L. Sales, F. Maffray, O. Porto, and B.A. Reed. Path parity and perfection. *Discrete Mathematics*, 165-166:233–252, 1997.
17. H. Everett, C.M.H. de Figueiredo, C. Linhares Sales, F. Maffray, O. Porto, B.A. Reed. Even pairs. In: L. Ramirez-Alfonsin and B.A. Reed (Eds.), *Perfect Graphs*, Wiley, 67–92, 2001.

18. C.M.H. de Figueiredo, J.G. Gimbel, C.P. Mello, and J.L. Szwarcfiter. Even and odd pairs in comparability and in P_4-comparability graphs. *Discrete Applied Mathematics*, 91(1-3):293–297, 1999.
19. J. Fonlupt and J.P. Uhry. Transformations which preserve perfectness and H-perfectness of graphs. *Annals of Discrete Mathematics*, 16:83–85, 1982.
20. D.R. Fulkerson and O.A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15:835–855, 1965.
21. F. Gavril. Algorithms on clique separable graphs. *Discrete Mathematics*, 19:159–165, 1977.
22. C.T. Hoàng and V.B. Le. Recognizing perfect 2-split graphs. *SIAM Journal on Discrete Mathematics*, 13(1):48–55, 2000.
23. W.-L. Hsu. Recognizing planar perfect graphs. *Journal of the ACM*, 34(2):255–288, 1987.
24. A.S. LaPaugh and C.H. Papadimitriou. The even-path problem for graphs and digraphs. *Networks*, 14:507–513, 1984.
25. B. Lévêque, D.Y. Lin, F. Maffray, and N. Trotignon. Detecting induced subgraphs. *Discrete Applied Mathematics*, 157: 3540–3551, 2009.
26. X. Li and W. Zang. A combinatorial algorithm for minimum weighted colorings of claw-free perfect graphs. *Journal of Combinatorial Optimization*, 9(4):331–347, 2005.
27. C. Linhares Sales and F. Maffray. Even pairs in claw-free perfect graphs. *Journal of Combinatorial Theory, Series B*, 74:169–191, 1998.
28. F. Maffray and B.A. Reed. A description of claw-free perfect graphs. *Journal of Combinatorial Theory, Series B*, 75:134–156, 1999.
29. H. Meyniel. A new property of critical imperfect graphs and some consequences. *European Journal of Combinatorics*, 8:313–316, 1987.
30. S. Parter. The use of linear graphs in Gauss elimination. *SIAM Review*, 3(2):119–130, 1961.
31. N.D. Roussopoulos. A $\max\{m, n\}$ algorithm for determining the graph H from its line graph G. *Information Processing Letters 2*, 108–112, 1973.
32. D.J. Rose, R.E. Tarjan and G.S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976.
33. R.M. Sampaio and C.L. Sales. On the complexity of finding even pairs in planar perfect graphs. In: Brazilian Symposium on Graphs, Algorithms and Combinatorics 2001, Fortaleza, *Electronic Notes in Discrete Mathematics*, 7:186–189, 2001.
34. C.R. Satyan and C.P. Rangan. The parity path problem on some subclasses of perfect graphs. *Discrete Applied Mathematics*, 68(3):293–302, 1996.
35. S. Shrem, M. Stern and M.C. Golumbic. Smallest odd holes in claw-free graphs. In: Proceedings of the 35th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2009), *Lecture Notes in Computer Science*, 5911: 329–340, 2009.
36. R.E. Tarjan. Decomposition by clique separators. *Discrete Mathematics*, 55:221–232, 1985.
37. N. Trotignon. *Graphes parfaits: Structure et algorithmes.* PhD Thesis, Université Joseph Fourier - Grenoble I (in French), 2004.
38. S.H. Whitesides. A method for solving certain graph recognition and optimization problems, with applications to perfect graphs. *Annals of Discrete Mathematics*, 21:281–297, 1984.