Additional information:

This work is supported by EPSRC (EP/G043434/1) and Royal Society (JP100692), and by the Research Council of Norway (197548/F20). Preliminary versions of different parts of the paper were presented at MFCS 2011 [18] and ISAAC 2011 [1].

Detecting Fixed Patterns in Chordal Graphs in Polynomial Time*

Rémy Belmonte[1], Petr A. Golovach[2], Pinar Heggernes[1], Pim van 't Hof[1], Marcin Kamiński[3], and Daniël Paulusma[2]

[1] Department of Informatics, University of Bergen, Norway
{remy.belmonte,pinar.heggernes,pim.vanthof}@ii.uib.no
[2] School of Engineering and Computing Sciences, Durham University, UK
{petr.golovach,daniel.paulusma}@durham.ac.uk
[3] Département d'Informatique, Université Libre de Bruxelles, Belgium
marcin.kaminski@ulb.ac.be

Abstract. The Contractibility problem takes as input two graphs G and H, and the task is to decide whether H can be obtained from G by a sequence of edge contractions. The Induced Minor and Induced Topological Minor problems are similar, but the first allows both edge contractions and vertex deletions, whereas the latter allows only vertex deletions and vertex dissolutions. All three problems are NP-complete, even for certain *fixed* graphs H. We show that these problems can be solved in polynomial time for every fixed H when the input graph G is chordal. Our results can be considered tight, since these problems are known to be W[1]-hard on chordal graphs when parameterized by the size of H. To solve Contractibility and Induced Minor, we define and use a generalization of the classic Disjoint Paths problem, where we require the vertices of each of the k paths to be chosen from a specified set. We prove that this variant is NP-complete even when $k = 2$, but that it is polynomial-time solvable on chordal graphs for every fixed k. Our algorithm for Induced Topological Minor is based on another generalization of Disjoint Paths called Induced Disjoint Paths, where the vertices from different paths may no longer be adjacent. We show that this problem, which is known to be NP-complete when $k = 2$, can be solved in polynomial time on chordal graphs even when k is part of the input. Our results fit into the general framework of graph containment problems, where the aim is to decide whether a graph can be modified into another graph by a sequence of specified graph operations. Allowing combinations of the four well-known operations edge deletion, edge contraction, vertex deletion, and vertex dissolution results in the following ten containment relations: (induced) minor, (induced) topological minor, (induced) subgraph, (induced) spanning subgraph, dissolution, and contraction. Our results, combined with existing results, settle the complexity of each of the ten corresponding containment problems on chordal graphs.

1 Introduction

We study algorithmic problems that aim to decide whether the structure of a graph H appears as a "pattern" within the structure of another graph G. The exact definition of "pattern" depends on the graph operations that are allowed when modifying G into H. We consider the following four elementary graph operations. The operations *vertex deletion* (VD) and *edge deletion* (ED) simply remove a vertex or an edge, respectively,

* This work is supported by EPSRC (EP/G043434/1) and Royal Society (JP100692), and by the Research Council of Norway (197548/F20). Preliminary versions of different parts of the paper were presented at MFCS 2011 [18] and ISAAC 2011 [1].

from the graph. The *edge contraction* (EC) operation, when applied to an edge uv, deletes the vertices u and v from the graph, and replaces them by a new vertex that is made adjacent to precisely those vertices to which u or v were adjacent. The *vertex dissolution* (VDi) operation can be applied to a vertex v of degree 2 whose two neighbors are not adjacent; it contracts one of the two edges incident with v. Table 1 shows ten graph containment relations obtained by combining these four operations. For example, a graph H is an induced minor of a graph G if H can be obtained from G by a sequence of vertex deletions and edge contractions (and consequently also vertex dissolutions), but not edge deletions. The corresponding decision problem, in which G and H form the ordered input pair (G, H), is called Induced Minor. The other rows in Table 1 are to be interpreted similarly.

Containment Relation	VD	ED	EC	VDi	Decision Problem
Minor	yes	yes	yes	yes	Minor
Induced minor	yes	no	yes	yes	Induced Minor
Topological minor	yes	yes	no	yes	Topological Minor
Induced topological minor	yes	no	no	yes	Induced Topological Minor
Contraction	no	no	yes	yes	Contractibility
Dissolution	no	no	no	yes	Dissolution
Subgraph	yes	yes	no	no	Subgraph Isomorphism
Induced subgraph	yes	no	no	no	Induced Subgraph Isomorphism
Spanning subgraph	no	yes	no	no	Spanning Subgraph Isomorphism
Isomorphism	no	no	no	no	Graph Isomorphism

Table 1. Ten known containment relations in terms of the four mentioned graph operations. The missing two combinations "no yes yes yes", and "no yes no yes" correspond to the minor and topological minor relations, respectively, if we allow an extra operation that removes isolated vertices.

With the exception of Graph Isomorphism, all problems in Table 1 are known to be NP-complete (cf. [29]). By the results of Robertson and Seymour [33], Grohe et al. [21], and Golovach et al. [19], Minor, Topological Minor, and Dissolution are in FPT with parameter $|V_H|$. The problems Spanning Subgraph Isomorphism and Graph Isomorphism require the input graphs G and H to have the same size, and hence they are trivially in FPT with parameter $|V_H|$. The problems Subgraph Isomorphism and Induced Subgraph Isomorphism are trivially in XP, as they can be solved by brute force checking all vertex subsets of G of size $|V_H|$. However, both problems are W[1]-hard with parameter $|V_H|$, as they both generalize the well-known Clique problem, which is known to be W[1]-complete when parameterized by the size of the desired clique [10]. In summary, all of the above problems are polynomial-time solvable for every fixed graph H.

In contrast, the problems that we focus on in this paper are harder. In particular, there exist graphs H such that H-Contractibility, H-Induced Minor, and H-Induced Topological Minor are NP-complete, where the "H-" in front of the problem names indicates the variant of the problems where the graph H is fixed and only G is part of the input. Moreover, the problems Contractibility, Induced Minor, and Induced Topological Minor are W[1]-hard with parameter $|V_H|$ even on chordal graphs. We

prove that these problems are in XP with parameter $|V_H|$ on chordal graphs. This implies that they can be solved in polynomial time on chordal graphs for every fixed graph H, as the class of chordal graphs is closed under edge contractions. Before we explain our results in more detail in the next section, we give an overview of known results on these problems.

For H-Contractibility, polynomial-time solvable and NP-complete cases, depending on H, can be found in a series of papers started by Brouwer and Veldman [7], followed by Levin et al. [27, 28], and Van 't Hof et al. [22]. The smallest NP-complete cases are when H is a path or a cycle on 4 vertices [7]. Fellows et al. [13] gave polynomial-time solvable and NP-complete cases for H-Induced Minor. The smallest known NP-complete case is a graph H on 68 vertices [13]. Even the question whether this problem is polynomial-time solvable for every fixed tree H is still open. Lévêque et al. [26] gave both polynomial-time solvable and NP-complete cases for H-Induced Topological Minor. This problem is NP-complete when H is a complete graph on 5 vertices, but its complexity is open when H is a complete graph on 4 vertices.

The following results, where $|V_H|$ is the parameter, are known for the case where the input graph G has a particular structure. The problems Contractibility and Induced Minor are in FPT on planar graphs by the respective results of Kamiński and Thilikos [24], and Fellows et al. [13]. Fiala et al. [11] showed that Induced Topological Minor is in XP on claw-free graphs, whereas Contractibility remains NP-complete on claw-free graphs even when H is a path on seven vertices [12]. Belmonte et al. [2] and Golovach et al. [19] independently proved that Contractibility is in XP on split graphs, which form a proper subclass of chordal graphs. In fact, all ten problems of Table 1 can be solved in polynomial time on split graphs for every fixed H [19]. However, whereas six of these problems are in FPT on split graphs, our three problems Contractibility, Induced Minor, Induced Topological Minor, as well as the problem Induced Subgraph Isomorphism, are W[1]-hard on split graphs, and hence on chordal graphs [19]. This motivates our study of these three problems on chordal graphs with respect to XP algorithms.

Chordal graphs are the graphs in which every cycle of length at least four contains a chord. Chordal graphs constitute one of the most famous and well-studied graph classes, as they have a large number of practical applications in fields like sparse matrix computations, computational biology, computer vision, and VLSI design (cf. [17, 20, 34]). This graph class properly contains other well-known graph classes, like forests, interval graphs, and split graphs (cf. [6, 32]).

2 Our Results and Methodology

Table 2 gives a survey on the classical and the parameterized complexity of the ten containment problems from Table 1 on chordal graphs. We replaced "chordal" by "general" or "split" wherever possible in order to present the results in their strongest form. All the results on chordal graphs in Table 2 are new, and we prove them in the remainder of this paper. The remaining results in the table, namely the ones on split graphs and general graphs, are known and have already been mentioned. Section 3 contains the necessary additional terminology and a (straightforward) proof showing that Minor, Topological

Minor, and Subgraph Isomorphism are in linear-time FPT with parameter $|V_H|$ on chordal graphs.

Containment Problem	Parameter: $\|V_H\|$
Minor	linear-time FPT on chordal; cubic-time FPT in general
Induced Minor	XP on chordal; W[1]-hard on split
Topological Minor	linear-time FPT on chordal; cubic-time FPT in general
Induced Topological Minor	XP on chordal; W[1]-hard on split
Contractibility	XP on chordal; W[1]-hard on split
Dissolution	linear-time FPT in general
Subgraph Isomorphism	linear-time FPT on chordal; cubic-time FPT in general
Induced Subgraph Isomorphism	XP in general; W[1]-hard on split
Spanning Subgraph Isomorphism	constant-time FPT in general
Graph Isomorphism	constant-time FPT in general

Table 2. The parameterized complexity of the ten problems from Table 1 on general graphs, chordal graphs and split graphs. All the results on chordal graphs are new.

The three results from Table 2 that are left to prove are that Contractibility, Induced Minor, and Induced Topological Minor are in XP with parameter $|V_H|$ on chordal graphs. In order to obtain these results, we design algorithms for solving two generalizations of the classical Disjoint Paths problem on chordal graphs; these might be of interest independently of the studied graph containment problems.

In order to solve Contractibility and Induced Minor, we first need to solve some other problems. In Section 4, we introduce the following generalization of the Disjoint Paths problem. A *terminal pair* in a graph $G = (V, E)$ is a specified pair of vertices s and t called *terminals*, and the *domain* of a terminal pair (s, t) is a specified subset $U \subseteq V$ containing both s and t. We say that two paths, each of which is between some terminal pair, are *vertex-disjoint* if they have no common vertices except possibly the vertices of the terminal pairs. This leads to the following decision problem.

Set-Restricted Disjoint Paths

Instance: A graph G, k terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$ in G, and their respective domains $U_1, \ldots, U_k$.

Question: Does G contain k pairwise vertex-disjoint paths $P_1, \ldots, P_k$ such that P_i is a path between s_i and t_i using only vertices from U_i, for $i = 1, \ldots, k$?

Note that the domains $U_1, \ldots, U_k$ are not necessarily pairwise disjoint. If we let every domain contain all vertices of G, we obtain exactly the Disjoint Paths problem.

The Set-Restricted Disjoint Paths problem is NP-complete on chordal, even interval, graphs, since Disjoint Paths is NP-complete on interval graphs [30]. Robertson and Seymour [33] showed that Disjoint Paths is in FPT with parameter k; their algorithm runs in $O(|V_G|^3)$ time for every fixed k. In contrast, we show that the more general Set-Restricted Disjoint Paths problem is NP-complete even when $k = 2$. We prove that on chordal graphs Set-Restricted Disjoint Paths is in XP with parameter k.

We then consider disjoint trees, or equivalently, disjoint connected subgraphs, instead of disjoint paths. This leads to the following problem.

Set-Restricted Disjoint Connected Subgraphs
Instance: A graph G, k pairwise disjoint nonempty vertex subsets $S_1, \ldots, S_k$ of G, and their respective domains $U_1, \ldots, U_k$.
Question: Does G contain k pairwise vertex-disjoint connected subgraphs $G_1, \ldots, G_k$ such that $S_i \subseteq V_{G_i} \subseteq U_i$, for $1 \leq i \leq k$?

Choosing each domains U_i to be V_G yields the Disjoint Connected Subgraphs problem, which was introduced by Robertson and Seymour [33]. This problem is NP-complete even when $k = 2$ and $\min\{|S_1|, |S_2|\} = 2$ [23]. Moreover, it is NP-complete on split graphs, and hence on chordal graphs, when $k = 2$. Robertson and Seymour [33] showed that it is in FPT with parameter $|S_1| + |S_2| + \cdots + |S_k|$. We show that the more general Set-Restricted Disjoint Connected Subgraphs problem is in XP with this parameter when restricted to chordal graphs.

In Section 5, we show how to use our XP algorithm for Set-Restricted Disjoint Connected Subgraphs as a subroutine for solving Contractibility and Induced Minor on chordal graphs in time $|V_G|^{O(|V_H|^2)}$.

In Section 6, we turn our attention to Induced Topological Minor. To solve this problem, we use another generalization of Disjoint Paths. We say that two paths P_1 and P_2 between two terminal pairs in a graph $G = (V, E)$ are *mutually induced* if they are vertex-disjoint and no vertex of P_1 is adjacent to a vertex of P_2, except possibly the terminal vertices. Note that each path is not necessarily induced (chordless), but we may assume this without loss of generality. This leads to the following decision problem.

Induced Disjoint Paths
Instance: A graph G and k terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$ in G.
Question: Does G contain k mutually induced paths $P_1, \ldots, P_k$ such that P_i is a path between s_i and t_i, for $i = 1, \ldots, k$?

The Induced Disjoint Paths problem is already NP-complete for $k = 2$, due to a result of Bienstock [3]. Hence, on general graphs this problem is harder than Disjoint Paths, which is in FPT with parameter k as mentioned earlier. Note that on general graphs, Induced Disjoint Paths generalizes Disjoint Paths; subdividing the edges of an input graph of the latter problem yields an instance of the former problem. However, this is not true on chordal graphs, as subdividing the edges of a chordal graph might result in a graph that is not chordal. Interestingly, on chordal graphs the two problems completely swap complexity: although Disjoint Paths is NP-complete on chordal graphs as mentioned above, we show that Induced Disjoint Paths is polynomial-time solvable on chordal graphs. We use the corresponding algorithm as a subroutine in our algorithm for solving Induced Topological Minor on chordal graphs in time $O(|E_H| \cdot |V_G|^{|V_H|+3})$.

In Section 7, we give another application of our algorithm for solving Set-Restricted Disjoint Paths by using it as a subroutine to solve the Set-Restricted Disjoint Connected Subgraphs problem on interval graphs. We conclude the paper by mentioning some open problems.

3 Preliminaries

All graphs considered in this paper are finite, undirected, and have neither self-loops nor multiple edges. All the problems in this paper have a graph G as a part of the input. Throughout the paper, we use n and m to denote the number of vertices and edges of this input graph G, respectively.

Let $G = (V, E)$ be a graph. If the vertex and edge sets of a graph G are not specified, we use V_G and E_G to denote these sets, respectively. A subset $U \subseteq V$ is a *clique* if every pair of vertices in U are adjacent. A vertex is *simplicial* if its neighbors form a clique. We write $G[U]$ to denote the subgraph of G induced by $U \subseteq V$. Two sets $U, U' \subseteq V$ are called *adjacent* if there exist vertices $u \in U$ and $u' \in U'$ such that $uu' \in E$. A path between vertices u and v is called a (u, v)-path. The set of vertices of a path P is denoted by V_P.

The *subdivision* of an edge $e = uw$ in a graph removes e, adds a new vertex v and two new edges uv and vw. We say that a graph G is a *subdivision* of a graph H if G can be obtained from H by a sequence of edge subdivisions. We observe that an edge subdivision is the reverse operation of a vertex dissolution. Hence, H is an (induced) topological minor of G if and only if an (induced) subgraph of G is a subdivision of H.

An *H-witness structure* of G is a partition of V into $|V_H|$ nonempty sets $W(x)$, one set for each $x \in V_H$, called *H-witness sets*, such that

(i) each $W(x)$ induces a connected subgraph of G; and
(ii) for all $x, y \in V_H$ with $x \neq y$, sets $W(x)$ and $W(y)$ are adjacent in G if and only if x and y are adjacent in H.

Observe that H is a contraction of G if and only if G has an H-witness structure: H can be obtained from G by contracting the edges in each H-witness set until a single vertex remains in each of them. This view provides an intuition on the hardness of the Contractibility problem: it is a partition problem rather than a subset problem.

A *tree decomposition* of G is a pair $(\mathcal{T}, \mathcal{X})$, where $\mathcal{X}$ is a collection of subsets of V, called *bags*, and $\mathcal{T}$ is a tree whose vertices, called *nodes*, are the sets of $\mathcal{X}$, such that the following three properties are satisfied.

- $\bigcup_{X \in \mathcal{X}} X = V$,
- for each edge $uv \in E$, there is a bag $X \in \mathcal{X}$ with $u, v \in X$,
- for each $x \in V$, the set of nodes containing x forms a connected subtree of $\mathcal{T}$.

The *width* of a tree decomposition $(\mathcal{T}, \mathcal{X})$ is the size of a largest bag in $\mathcal{X}$ minus 1. The *treewidth* of G is the minimum width over all possible tree decompositions of G.

A *chord* of a path or a cycle is an edge between two non-consecutive vertices of the path or the cycle. A graph is *chordal* if every cycle of it on at least four vertices has a chord. It is not difficult to see that the class of chordal graphs is closed under vertex deletions and edge contractions but not under edge deletions. Chordal graphs can be recognized in linear time [35]. Every chordal graph contains at most n maximal cliques, and, if it is not a complete graph, at least two non-adjacent simplicial vertices [9]. A graph G is chordal if and only if it has a tree decomposition whose set of bags is exactly the set of maximal

cliques of G [16]. Such a tree decomposition is called a *clique tree* and can be constructed in linear time [4].

The complexity classes XP and FPT are defined in the framework of parameterized complexity. A parameterized problem Q belongs to the class XP if for each instance (I, k) it can be decided in $|I|^{f(k)}$ time whether $(I, k) \in Q$, where f is a function that depends only on the *parameter* k, and $|I|$ denotes the size of I. If a problem belongs to XP, then it can be solved in polynomial time for every fixed k. Hence, if a problem is NP-complete for some fixed value of k, then it is unlikely to belong to XP. If a parameterized problem can be solved by an algorithm with running time $f(k)\,|I|^{O(1)}$, then the problem belongs to the class FPT. A problem is in cubic-time FPT, linear-time FPT, or constant-time FPT if it can be solved in time $f(k)\,|I|^3$, in time $f(k)\,|I|$ or in time $f(k)$, respectively. Between FPT and XP is a hierarchy of parameterized complexity classes, $\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{W}[\mathsf{P}] \subseteq \mathsf{XP}$, where hardness for one of the W-classes is considered to be strong evidence of intractability with respect to the class FPT. For formal background on parameterized complexity, we refer to the textbooks by Downey and Fellows [10], Flum and Grohe [14], and Niedermeier [31].

Let u, v, w be three distinct vertices in a graph such that uv and vw are edges. The operation that removes the edges uv and vw, and adds the edge uv in the case u and w are not adjacent, is called a *lift*. A graph G contains H as a *immersion* if H can be obtained from G by a sequence of vertex deletions, edge deletions, and lifts. The corresponding decision problem is called Immersion. Grohe et al. [21] showed that Immersion is in FPT with parameter $|V_H|$; their running time is $O(n^3)$ for every fixed H. Proposition 1 below proves that this problem can be solved in linear time on chordal graphs, and also proves the linear-time FPT results on Minor, Topological Minor, and Subgraph Isomorphism that were stated in Table 2.

Proposition 1. *The problems* Minor, Topological Minor, Subgraph Isomorphism, *and* Immersion *can be solved in* $f(|V_H|)\cdot(n+m)$ *time on chordal graphs for some function* f *that depends only on* $|V_H|$.

Proof. Let G be a chordal input graph. A clique C of maximum size in G can be found in $O(n + m)$ time [35]. If $|C| \geq |V_H|$, then $G[C]$, and hence G, contains H as a subgraph and thus as a minor, topological minor, and immersion. If $|C| < |V_H|$, then the treewidth of G is $|C| - 1 < |V_H| - 1$ by the definition of a clique tree. Because H is fixed, the treewidth of G is bounded by a constant. A seminal result of Courcelle [8] states that on every class of graphs of bounded treewidth, every problem expressible in monadic second-order logic, i.e., the fragment of second-order logic where quantified relation symbols must have arity 1, can be solved in $O(n)$ time. For fixed H, it is known that the problems H-Subgraph Isomorphism, H-Minor, H-Topological Minor, and H-Immersion can all be expressed in monadic second-order logic; in particular we refer to Grohe et al. [21] for the case of immersions. Since a clique tree can be constructed in $O(n + m)$ time, this completes the proof of Proposition 1. □

4 Set-Restricted Disjoint Paths

We start with the following result.

Theorem 1. Set-Restricted Disjoint Paths *is* NP*-complete even when* $k = 2$.

Proof. We reduce from the NP-complete 3-Sat problem [15]. It is well known that this problem remains NP-complete when each Boolean variable occurs at most twice as a positive literal and at most twice as a negative literal. We use this variant for our reduction. Let Φ be an instance of 3-Sat with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$. We construct a graph G as follows; see also Figure 1.

- Add two vertices s and t.
- Add $n+1$ vertices $v_0, \ldots, v_n$ and edges sv_0 and v_nt.
- For $i = 1, \ldots, n$, add vertices $x_i^{(1)}, x_i^{(2)}, \overline{x}_i^{(1)}, \overline{x}_i^{(2)}, y_i, \overline{y}_i$ and edges $v_{i-1}x_i^{(1)}$, $x_i^{(1)}y_i$, $y_ix_i^{(2)}$, $x_i^{(2)}v_i$, $v_{i-1}\overline{x}_i^{(1)}$, $\overline{x}_i^{(1)}\overline{y}_i$, $\overline{y}_i\overline{x}_i^{(2)}$, $\overline{x}_i^{(2)}v_i$.

  Let $Q_i = v_{i-1}x_i^{(1)}y_ix_i^{(2)}v_i$ and $\overline{Q}_i = v_{i-1}\overline{x}_i^{(1)}\overline{y}_i\overline{x}_i^{(2)}v_i$.
- Add $m+1$ vertices $u_0, \ldots, u_m$ and edges su_0 and u_mt.
- For each clause C_j and each literal z in C_j:
  - if $z = x_i$, then add edges $u_{j-1}x_i^{(1)}$, $u_jx_i^{(1)}$ if z is the first occurrence of the literal x_i in Φ, and edges $u_{j-1}x_i^{(2)}$, $u_jx_i^{(2)}$ if z is the second occurrence.
  - if $z = \overline{x}_i$, then add edges $u_{j-1}\overline{x}_i^{(1)}$, $u_j\overline{x}_i^{(1)}$ if z is the first occurrence of the literal $\overline{x}_i$ in Φ, and edges $u_{j-1}\overline{x}_i^{(2)}$, $u_j\overline{x}_i^{(2)}$ if z is the second occurrence.

  Let $R_j(z)$ be the obtained (u_{j-1}, u_j)-path of length two.
- Let $U_1 = V_G \setminus \{u_0, \ldots, u_m\}$.
- Let $U_2 = \{s,t\} \cup \{u_0, \ldots, u_m\} \cup \{x_1^{(1)}, \ldots, x_n^{(1)}\} \cup \{x_1^{(2)}, \ldots, x_n^{(2)}\}$.

We prove that Φ can be satisfied if and only if there are two vertex-disjoint (s,t)-paths P_1 and P_2 in G such that $V_{P_1} \subseteq U_1$ and $V_{P_2} \subseteq U_2$; recall that we allow such paths to have common end-vertices, as is the case here.

First suppose that the variables $x_1, \ldots, x_n$ have a truth assignment that satisfies Φ. We construct P_1 as follows. For each $i \in \{1, \ldots, n\}$, we choose Q_i if $x_i = \texttt{false}$, and $\overline{Q}_i$ if $x_i = \texttt{true}$. Afterwards, we concatenate the chosen paths. We get a (v_0, v_n)-path, and construct the (s,t)-path P_1 by adding the edges sv_0 and v_nt. By construction, $V_{P_1} \subseteq U_1$. We construct P_2 as follows. For each $j \in \{1, \ldots, m\}$, the clause C_j can be assumed to contain a literal $z = \texttt{true}$, and we select such a literal and the corresponding paths $R_j(z)$. Afterwards, we concatenate the chosen paths. We get a (u_0, u_m)-path, and construct the (s,t)-path P_2 by adding the edges su_0 and u_mt. By construction, $V_{P_2} \subseteq U_2$. If P_2 contains $R_j(z) = u_{j-1}x_i^{(1)}u_j$ or $R_j(z) = u_{j-1}x_i^{(2)}u_j$ as a subpath, then $x_i = \texttt{true}$ implying that $\overline{Q}_i$ is a subpath of P_1. Hence, $x_i^{(1)}, x_i^{(2)} \notin V_{P_1}$. We conclude that P_1 and P_2 are vertex-disjoint.

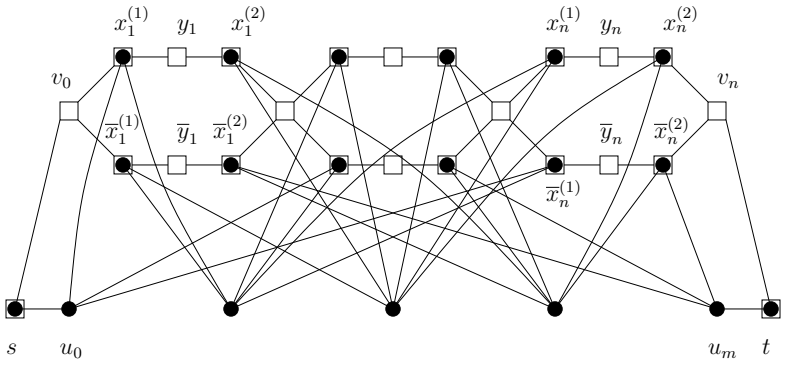

Fig. 1. The graph G; the vertices of U_1 and U_2 are shown by white squares and black circles, respectively. A white square with a black circle inside indicates that the vertex belongs to both sets.

Now suppose that there are vertex-disjoint (s,t)-paths P_1 and P_2 in G, such that $V_{P_1} \subseteq U_1$ and $V_{P_2} \subseteq U_2$. Note that for each $i \in \{1,\ldots,n\}$, either Q_i or $\overline{Q}_i$ is a subpath of P_1. If Q_i is a subpath of P_1, then we set $x_i = \texttt{false}$, and $x_i = \texttt{true}$ otherwise. Note that for each $j \in \{1,\ldots,m\}$, the path P_2 contains $R_j(z)$ as a subpath for some literal z in C_j. If $z = x_i$ for some variable x_i, then our assumption that P_1 and P_2 are vertex-disjoint implies that $\overline{Q}_i$ is a subpath of P_1. Hence, $x_i = \texttt{true}$, and consequently $z = \texttt{true}$. Similarly, if $z = \overline{x}_i$ for some variable x_i, then Q_i is a subpath of P_1 and $x_i = \texttt{false}$, and consequently, $z = \texttt{true}$. Hence, each clause C_j is satisfied by this truth assignment, and $\Phi = \texttt{true}$, as desired. This completes the proof of Theorem 1. □

We apply dynamic programming to prove that Set-Restricted Disjoint Paths can be solved in polynomial time on chordal graphs for every fixed integer k. The first key observation is that the existence of k disjoint paths is equivalent to the existence of k disjoint *induced*, i.e. chordless, paths. The second key observation is that every induced path contains at 0, 1, or 2 vertices from each clique. Our algorithm solves the decision problem, but it can easily be modified to produce the desired paths if they exist.

Kloks [25] showed that every tree decomposition of a graph can be converted in linear time to a *nice tree decomposition*, such that the size of the largest bag does not increase, and the total size of the tree is linear in the size of the original tree. A tree decomposition $(\mathcal{T}, \mathcal{X})$ is *nice* if $\mathcal{T}$ is a binary tree with root X_r such that the nodes of $\mathcal{T}$ are of the following four types:

1. a *leaf node* X is a leaf of $\mathcal{T}$ and has size $|X| = 1$;
2. an *introduce node* X has one child X' with $X = X' \cup \{v\}$ for some vertex $v \in V_G$;
3. a *forget node* X has one child X' with $X = X' \setminus \{v\}$ for some vertex $v \in V_G$;
4. a *join node* X has two children X' and X'' with $X = X' = X''$.

Applying the conversion algorithm of Kloks [25] on a clique tree of a chordal graph G leads to a nice tree decomposition of G with the additional property that each bag is a (not necessary maximal) clique in G. We use such nice tree decompositions of chordal graphs in our algorithm for Set-Restricted Disjoint Paths, which we describe next.

Let k be a positive integer, and let G be a chordal graph with k terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$ that have domains $U_1, \ldots, U_k$, respectively. If G is disconnected, we check for each terminal pair (s_i, t_i) whether s_i and t_i belong to the same connected component. If not, then we return `No`. Otherwise, we consider each connected component and its set of terminals separately. Consequently, we assume from now on that G is connected.

We construct a nice tree decomposition $(\mathcal{T}, \mathcal{X})$ of G with root X_r, such that each bag is a clique in G. For every node $X_i \in V_{\mathcal{T}}$, we denote by $\mathcal{T}_i$ the subtree of $\mathcal{T}$ with root X_i induced by X_i and all its descendants. We define $G_i = G[\bigcup_{j \in V_{\mathcal{T}_i}} X_j]$, i.e., the subgraph of G induced by the set of all vertices of G appearing in bags of $\mathcal{T}_i$.

Our dynamic programming algorithm keeps a table for each node of $\mathcal{T}$. For a node X_i, the table stores a collection of records

$$\mathcal{R} = ((State_1, R_1), \ldots, (State_k, R_k)),$$

where $R_1, \ldots, R_k \subseteq X_i$ are ordered sets without common vertices except (possibly) terminals, where $R_j \subseteq U_j$ and $0 \leq |R_j| \leq 2$ for $j \in \{1, \ldots, k\}$, and where each $State_j$ can have one of the following four values:

Not initialized, *Started from* s, *Started from* t, or *Completed*.

These records correspond to the partial solution of Set-Restricted Disjoint Paths for G_i with the following properties.

- If $State_j =$ *Not initialized*, then $s_j, t_j \notin V_{G_i}$. If $R_j = \emptyset$, then (s_j, t_j)-paths have no vertices in G_i in the partial solution. If $R_j = \langle z \rangle$, then z is the unique vertex of a (s_j, t_j)-path in G_i in the partial solution. If $R_j = \langle z_1, z_2 \rangle$, then z_1, z_2 are vertices in a (s_j, t_j)-path, z_1 is the predecessor of z_2 in the path, and this path has no other vertices in G_i.
- If $State_j =$ *Started from* s, then $s_j \in V_{G_i}$, $t_j \notin V_{G_i}$ and R_j contains either one or two vertices. If $R_j = \langle z \rangle$, then the partial solution contains an (s_j, z)-path with the unique vertex $z \in X_i$. If $R_j = \langle z_1, z_2 \rangle$, then the partial solution contains an (s_j, z_2)-path such that z_1 is the predecessor of z_2 with exactly two vertices $z_1, z_2 \in X_i$.
- If $State_j =$ *Started from* t, then $s_j \notin V_{G_i}$, $t_j \in V_{G_i}$ and R_j contains either one or two vertices. If $R_j = \langle z \rangle$, then the partial solution contains a (z, t_j)-path with the unique vertex $z \in X_i$. If $R_j = \langle z_1, z_2 \rangle$, then the partial solution contains an (z_1, t_j)-path such that z_2 is the successor of z_1 with exactly two vertices $z_1, z_2 \in X_i$.
- If $State_j =$ *Completed*, then $s_j \in V_{G_i}$, $t_j \in V_{G_i}$. The partial solution in this case contains an (s_j, t_j)-path, and R_j is the set of vertices of this path in X_i. If $R_j = \langle z_1, z_2 \rangle$, then z_1 is the predecessor of z_2 in the path.

The tables are constructed and updated as follows.

Leaf nodes. Let $X_i = \{u\}$ be a leaf node of $\mathcal{T}$. Then the table for X_i stores all records $\mathcal{R} = ((State_1, R_1), \ldots, (State_k, R_k))$ with the following properties. If u is a terminal, then for $j = 1, \ldots, k$,

- if $u = s_j$, then $State_j =$ *Started from* s and $R_j = \langle u \rangle$;

– if $u = t_j$, then $State_j = \textit{Started from } t$ and $R_j = \langle u \rangle$; and
– if $u \neq s_j$ and $u \neq t_j$, then $State_j = \textit{Not initialized}$ and $R_j = \emptyset$.

If u is not a terminal, then $State_j = \textit{Not initialized}$ for $j = 1, \ldots, k$, and either $R_j = \emptyset$ for all $j \in \{1, \ldots, k\}$, or exactly one set $R_j = \langle u \rangle$ if $u \in U_j$ and other sets are empty.

Introduce nodes. Let X_i be an introduce node with child $X_{i'}$, and let $X_i = X_{i'} \cup \{u\}$ for some $u \in V_G$. We consider two cases.

Case 1. u is a terminal.
For each record $\mathcal{R}' = ((State'_1, R'_1), \ldots, (State'_k, R'_k))$ from the table for $X_{i'}$, we either modify $\mathcal{R}'$ and include the modified record $\mathcal{R} = ((State_1, R_1), \ldots, (State_k, R_k))$ in the table for X_i, or we discard $\mathcal{R}'$. If there exists an index j such that $u \in \{s_j, t_j\}$ and $|R'_j| = 2$, then $\mathcal{R}'$ is discarded. Otherwise,

– if $u = s_j$ and $State'_j = \textit{Not initialized}$, then $State_j = \textit{Started from } s$; if $R'_j = \emptyset$, then $R_j = \langle u \rangle$, and if $R'_j = \langle z \rangle$, then $R_j = \langle u, z \rangle$;
– if $u = s_j$ and $State'_j = \textit{Started from } t$, then $State_j = \textit{Completed}$; if $R'_j = \langle z \rangle$, then $R_j = \langle z, u \rangle$;
– if $u = t_j$ and $State'_j = \textit{Not initialized}$, then $State_j = \textit{Started from } t$; if $R'_j = \emptyset$, then $R_j = \langle u \rangle$, and if $R'_j = \langle z \rangle$, then $R_j = \langle z, u \rangle$;
– if $u = t_j$ and $State'_j = \textit{Started from } s$, then $State_j = \textit{Completed}$; if $R'_j = \langle z \rangle$ then $R_j = \langle z, u \rangle$; and
– if $u \neq s_j$ and $u \neq t_j$, then $State_j = State'_j$ and $R_j = R'_j$.

Case 2. u is not a terminal.
We include all records from the table for $X_{i'}$ in the table for X_i. In addition, we add new records to the table for X_i according to the following rules. For each $j \in \{1, \ldots, k\}$ with $u \in U_j$, we consider the records $\mathcal{R}' = ((State'_1, R'_1), \ldots, (State'_k, R'_k))$ from the table for $X_{i'}$ with $|R'_j| \leq 1$, and do the following:

– if $State'_j = \textit{Not initialized}$ and $R'_j = \langle z \rangle$, then we add to the table for X_i two records such that $State_l = State'_l$ and $R_l = R'_l$ for $l \in \{1, \ldots, k\}$, $l \neq j$, $State_j = State'_j$, and $R_j = \langle z, u \rangle$ for the first record and $R_j = \langle u, z \rangle$ for the second;
– if $State'_j = \textit{Not initialized}$ and $R'_j = \emptyset$, then we include in the table for X_i the record such that $State_l = State'_l$ and $R_l = R'_l$ for $l \in \{1, \ldots, k\}$, $l \neq j$, $State_j = State'_j$, and $R_j = \langle u \rangle$;
– if $State'_j = \textit{Started from } s$ and $R'_j = \langle z \rangle$, then we add to the table for X_i the record such that $State_l = State'_l$ and $R_l = R'_l$ for $l \in \{1, \ldots, k\}$, $l \neq j$, $State_j = State'_j$, and $R_j = \langle z, u \rangle$;
– if $State'_j = \textit{Started from } t$ and $R'_j = \langle z \rangle$, then we add to the table for X_i the record such that $State_l = State'_l$ and $R_l = R'_l$ for $l \in \{1, \ldots, k\}$, $l \neq j$, $State_j = State'_j$, and $R_j = \langle u, z \rangle$.

Forget nodes. Let X_i be a forget node with child $X_{i'}$, and let $X_i = X_{i'} \setminus \{u\}$ for some $u \in V_G$. Every record $\mathcal{R}' = ((State'_1, R'_1), \ldots, (State'_k, R'_k))$ from the table for $X_{i'}$ with $u \notin R'_j$ for all $j \in \{1, \ldots, k\}$ is included in the table for X_i. For each record $\mathcal{R}' =$

$((State'_1, R'_1), \ldots, (State'_k, R'_k))$ from the table for $X_{i'}$ such that $u \in R'_j$ for some $j \in \{1, \ldots, k\}$, we either modify it and include the modified record $\mathcal{R} = ((State_1, R_1), \ldots, (State_k, R_k))$ in the table for X_i, or we discard $\mathcal{R}'$, using the following rules:

- for all $j \in \{1, \ldots, k\}$, $State_j = State'_j$;
- for $j \in \{1, \ldots, k\}$, if $u \notin R'_j$, then $R_j = R'_j$;
- for $j \in \{1, \ldots, k\}$, if $u \in R'_j$ and $State_j =$ *Started from* s, then we discard the record if $R'_j = \langle u \rangle$ or $R'_j = \langle z, u \rangle$, and we set $R_j = \langle z \rangle$ if $R'_j = \langle u, z \rangle$;
- for $j \in \{1, \ldots, k\}$, if $u \in R'_j$ and $State_j =$ *Started from* t, then we discard the record if $R'_j = \langle u \rangle$ or $R'_j = \langle u, z \rangle$, and we set $R_j = \langle z \rangle$ if $R'_j = \langle z, u \rangle$;
- for $j \in \{1, \ldots, k\}$, if $u \in R'_j$ and $State_j =$ *Not initialized*, then we discard the record;
- for $j \in \{1, \ldots, k\}$, if $u \in R'_j$ and $State_j =$ *Completed*, then $R_j = R'_j \setminus \langle u \rangle$.

Join nodes. Let X_i be a join node with children $X_{i'}$ and $X_{i''}$. For each pair of records $\mathcal{R}' = ((State'_1, R'_1), \ldots, (State'_k, R'_k))$ and $\mathcal{R}'' = ((State''_1, R''_1), \ldots, (State''_k, R''_k))$ from the tables for $X_{i'}$ and $X_{i''}$, respectively, such that $R'_j = R''_j$ for all $j \in \{1, \ldots, k\}$, we construct the record $\mathcal{R} = ((State_1, R_1), \ldots, (State_k, R_k))$ and include it in the table for X_i:

- for $j = 1, \ldots, k$, $R_j = R'_j = R''_j$;
- for $j = 1, \ldots, k$, if $State'_j = State''_j$, then $State_j = State'_j = State''_j$;
- for $j = 1, \ldots, k$, if $State'_j =$ *Not initialized*, then $State_j = State''_j$;
- for $j = 1, \ldots, k$, if $State''_j =$ *Not initialized*, then $State_j = State'_j$;
- for $j = 1, \ldots, k$, if $State'_j =$ *Completed* or $State''_j =$ *Completed*, then $State_j =$ *Completed*;
- for $j = 1, \ldots, k$, if $State'_j =$ *Started from* s and $State''_j =$ *Started from* t or $State'_j =$ *Started from* t and $State''_j =$ *Started from* s, then $State_j =$ *Completed*.

The algorithm computes these tables for all nodes of $\mathcal{T}$, starting from the leaves. Finally, the table for the root X_r is constructed. The algorithm returns `Yes` if the table for X_r contains a record $\mathcal{R} = ((State_1, R_1), \ldots, (State_k, R_k))$ with $State_1 = \ldots = State_k =$ *Completed*, and it returns `No` otherwise.

Theorem 2. *The* Set-Restricted Disjoint Paths *problem can be solved in* $n^{O(k)}$ *time on chordal graphs.*

Proof. The correctness of the algorithm follows from its description, keeping in mind that we are looking for k disjoint *induced* paths, each of which contains at most two vertices of every clique. For the running time, recall that a clique tree can be constructed in linear time [4] and that it can be converted in linear time to a nice tree decomposition in which each bag corresponds to a clique [25]. It remains to observe that each table contains at most $n^{O(k)}$ records, since each R_j has at most two elements. Since there are $O(n)$ nodes in $\mathcal{T}$ and hence $O(n)$ tables in total, our algorithm runs in $n^{O(k)}$ time. □

For our purposes, we need to generalize Theorem 2 in the following way.

Corollary 1. *The* Set-Restricted Disjoint Connected Subgraphs *problem can be solved in* $n^{O(p)}$ *time on chordal graphs, where* $p = \sum_{i=1}^{k} |S_i|$ *is the sum of the sizes of the terminal sets.*

Proof. Let $G = (V, E)$ be a chordal graph on n vertices with terminal sets $S_1, \ldots, S_k$ and corresponding domains $U_1, \ldots, U_k$. To solve Set-Restricted Disjoint Connected Subgraphs on G, we check whether G contains pairwise vertex-disjoint trees $T_1, \ldots, T_k$, such that T_i contains all vertices of S_i and such that all its other vertices are from U_i for $i = 1, \ldots, k$. For this purpose, we simply generate all collections of k subsets of V that can possibly give us the desired subtrees $T_1, \ldots, T_k$. We need to argue that it is sufficient to generate collections that consist of $O(p)$ vertices.

Let $p_i = |S_i|$ for $i = 1, \ldots, k$. Observe that every inclusion minimal subtree T_i of G with $S_i \subseteq V_{T_i}$ contains at most $p_i - 2$ vertices of degree at least 3 that are not in S_i. Repeatedly contracting every edge in T_i that is incident with a vertex of degree 2 in T_i, results in a reduced tree T_i' with at most $2p_i - 2$ vertices, and consequently, at most $2p_i - 3$ edges. Every edge in T_i' corresponds to a path in G. Hence we can guess the $2p_i - 2$ vertices of each possible reduced tree T_i', and then expand each edge of the tree to a path in G if possible, to obtain every possible tree T_i, using our algorithm for Set-Restricted Disjoint Paths. Consequently, we proceed as follows.

We iterate over every collection of k pairwise disjoint sets $X_1, \ldots, X_k$ with $X_i \subseteq U_i \setminus S_i$ and $|X_i| \leq p_i - 2$ for $i = 1, \ldots, k$. In each collection, for each $S_i \cup X_i$, we generate the set $\mathcal{T}_i$ of all possible trees with vertex set $S_i \cup X_i$ such that the vertices of X_i are of degree at least 3. Every edge st in a tree from $\mathcal{T}_i$ gives us a terminal pair (s', t') with domain U_i. We now pick one tree T_i' for $i = 1, \ldots, k$. This leads to a choice of trees $T_1', \ldots, T_k'$, where each T_i' corresponds to a set of terminal pairs with domain U_i as we explained above. We call such a choice a *T'-combination*. Let $(s_1', t_1'), \ldots, (s_{k'}', t_{k'}')$ denote the union of these k sets of terminal pairs and denote the domain of (s_h', t_h') by U_h' for $h = 1, \ldots, k'$. We note that $U_h' = U_i$ if and only if (s_h', t_h') is a terminal pair corresponding to an edge in T_i'. We also note that $k' \leq \sum_{i=1}^{k}(2p_i - 3)$, because each T_i' has at most $2p_i - 3$ edges. We now solve the Set-Restricted Disjoint Paths problem for this instance. If we find a solution, then we return `Yes`. Otherwise, we choose another T'-combination and solve the resulting instance, until we have considered all T'-combinations.

In order to prove that the above procedure leads to a total running time of $n^{O(p)}$, we observe that there are $n^{O(p)}$ possibilities to choose the sets $X_1, \ldots, X_k$. Moreover, by Cayley's formula, we have at most $(2p_i - 2)^{2p_i - 4} \leq p^{2p_i}$ different possibilities to join the vertices of $S_i \cup X_i$ by paths to obtain a tree. For each choice, we check the existence of at most $\sum_{i=1}^{k}(2p_i - 3) \leq 2p$ disjoint paths, which can be done in $n^{O(2p)}$ time by Theorem 2. Hence, the total running time is $n^{O(p)} \cdot p^{2p_1 + \cdots + 2p_k} \cdot n^{O(2p)} = n^{O(p)}$. This completes our proof. □

5 Contractions and Induced Minors in Chordal Graphs

First, in Section 5.1 below, we give a structural characterization of chordal graphs that contain a fixed graph H as a contraction. Then, in Section 5.2, we present our XP algorithm for solving Contractibility on chordal graphs, and show how it can be used to solve Induced Minor as well.

5.1 Properties

Throughout Section 5.1, let G be a connected chordal graph, let $\mathcal{T}_G$ be a clique tree of G, and let H be a graph with $V_H = \{x_1, \ldots, x_k\}$. For a set of vertices $A \subseteq V_G$, we let $G(A)$ denote the induced subgraph of G obtained by recursively deleting simplicial vertices that are not in A. Since every leaf in every clique tree contains at least one simplicial vertex, we immediately obtain Lemma 1 below. This lemma, in combination with Lemma 2, is crucial for the running time of our algorithm.

Lemma 1. *For every set $A \subseteq V_G$, every clique tree of $G(A)$ has at most $|A|$ leaves.*

Lemma 2. *The graph H is a contraction of G if and only if there is a set $A \subseteq V_G$ such that $|A| = k$ and H is a contraction of $G(A)$.*

Proof. First suppose that H is a contraction of G. Let $\mathcal{W}$ be an H-witness structure of G. For each $i \in \{1, \ldots, k\}$, we choose an arbitrary vertex $a_i \in W(x_i)$, and let $A = \{a_1, \ldots, a_k\}$. Suppose that G has a simplicial vertex $v \notin A$, and assume without loss of generality that $v \in W(x_1)$. Because $v \neq a_1$ and $a_1 \in W(x_1)$, we find that $|W(x_1)| \geq 2$. Hence, $W(x_1)$ contains a vertex u adjacent to v. The graph G', obtained from G by deleting v, is isomorphic to the graph obtained from G by contracting uv, since v is simplicial. Because u and v belong to the same witness set, namely $W(x_1)$, this implies that H is a contraction of G'. Using these arguments inductively, we find that H is a contraction of $G(A)$.

Now suppose that A is a subset of V_G with $|A| = k$, and that H is a contraction of $G(A)$. Deleting a simplicial vertex v in a graph is equivalent to contracting an edge incident with v. This means that $G(A)$ is a contraction of G. Because H is a contraction of $G(A)$ and contractibility is a transitive relation, we conclude that H is a contraction of G as well. □

For a subtree $\mathcal{T}$ of $\mathcal{T}_G$, we say that a vertex $v \in V_G$ is an *inner* vertex for $\mathcal{T}$ if v only appears in the maximal cliques of G that are nodes of $\mathcal{T}$. By $I(\mathcal{T}) \subseteq V_G$ we denote the set of all inner vertices for $\mathcal{T}$. For a subset $S \subseteq V_G$, let $\mathcal{T}_S$ be the unique minimal subtree of $\mathcal{T}_G$ that contains all maximal cliques of G that have at least one vertex of S; we say that a vertex v is an *inner* vertex for S if $v \in I(\mathcal{T}_S)$, and we set $I(S) = I(\mathcal{T}_S)$. Lemma 4 below provides an alternative and useful structural description of G if it contains H as a contraction. We need the following lemma to prove Lemma 4.

Lemma 3. *Let $S \subseteq V_G$ and let T be a subgraph of G that is a tree such that $S \subseteq V_T \subseteq I(S)$. Then $K \cap V_T \neq \emptyset$ for each node K of $\mathcal{T}_S$.*

Proof. Let K be a node of $\mathcal{T}_S$. If $K \cap S \neq \emptyset$, then clearly $K \cap V_T \neq \emptyset$. Suppose that $K \cap S = \emptyset$. Because $\mathcal{T}_S$ is the unique minimal subtree of $\mathcal{T}_G$ that contains all maximal cliques of G that have at least one vertex of S, we find that K separates two nodes K_1 and K_2 in $\mathcal{T}_S$ for which $K_1 \cap S \neq \emptyset$ and $K_2 \cap S \neq \emptyset$. This means that K separates two vertices $u \in K_1 \cap S$ and $v \in K_2 \cap S$ in G. Since T is a tree and $u, v \in V_T$, at least one vertex of T must be in K. □

Let l denote the number of leaves in $\mathcal{T}_G$; if $\mathcal{T}_G$ consists of one node, then we say that this node is a leaf of $\mathcal{T}_G$.

Lemma 4. *The graph H is a contraction of G if and only if there are pairwise disjoint nonempty sets of vertices $S_1, \ldots, S_k \subseteq V_G$, each of size at most l, such that*

1. *$V_G \subseteq I(S_1) \cup \ldots \cup I(S_k)$;*
2. *$V_{\mathcal{T}_{S_i}} \cap V_{\mathcal{T}_{S_j}} \neq \emptyset$ if and only if $x_i x_j \in E_H$ for $1 \leq i < j \leq k$;*
3. *G has pairwise vertex-disjoint trees $T_1, \ldots, T_k$ with $S_i \subseteq V_{T_i} \subseteq I(S_i)$ for $1 \leq i \leq k$.*

Proof. First suppose that H is a contraction of G. Consider a corresponding H-witness structure $\mathcal{W}$ of G. For $i = 1, \ldots, k$, let $\mathcal{T}_i$ be the subgraph of $\mathcal{T}_G$ induced by the maximal cliques of G that contain one or more vertices of $W(x_i)$. Because each $W(x_i)$ induces a connected subgraph of G, each $\mathcal{T}_i$ is connected. This means that $\mathcal{T}_i$ is a subtree of $\mathcal{T}_G$, i.e., $\mathcal{T}_i = \mathcal{T}_{W(x_i)}$. We construct S_i as follows. For each leaf K of $\mathcal{T}_i$, we choose a vertex of $W(x_i) \cap K$ and include it in the set S_i. Because $\mathcal{T}_G$ has l leaves, each $\mathcal{T}_i$ has at most l leaves. Hence, $|S_i| \leq l$ for $i = 1, \ldots, k$. We now check conditions 1–3 of the lemma.

1. By construction, we have $\mathcal{T}_i = \mathcal{T}_{S_i}$. All vertices of $W(x_i)$ are inner vertices for $\mathcal{T}_i$, so $W(x_i) \subseteq I(\mathcal{T}_i) = I(\mathcal{T}_{S_i}) = I(S_i)$. Hence, $V_G = \bigcup_{i=1}^k W(x_i) \subseteq \bigcup_{i=1}^k I(S_i)$.

2. Any two vertices $u, v \in V_G$ are adjacent if and only if there is a maximal clique K in G containing u and v. Hence, two witness sets $W(x_i)$ and $W(x_j)$ are adjacent if and only if there is a maximal clique K in G such that $K \cap W(x_i) \neq \emptyset$ and $K \cap W(x_j) \neq \emptyset$. This means that $W(x_i)$ and $W(x_j)$ are adjacent if and only if $V_{\mathcal{T}_i} \cap V_{\mathcal{T}_j} \neq \emptyset$. It remains to recall that $\mathcal{T}_i = \mathcal{T}_{S_i}$ and $\mathcal{T}_j = \mathcal{T}_{S_j}$, and that two witness sets $W(x_i)$ and $W(x_j)$ are adjacent if and only if $x_i x_j \in E_H$.

3. Every $G[W(x_i)]$ is a connected graph. Hence, every $G[W(x_i)]$ contains a spanning tree T_i. Because the sets $W(x_1), \ldots, W(x_k)$ are pairwise disjoint, the trees T_1, …, T_k are pairwise vertex-disjoint. Moreover, as we already deduced, $S_i \subseteq V_{T_i} = W(x_i) \subseteq I(S_i)$ for $i = 1, \ldots, k$.

Now suppose that there are pairwise disjoint nonempty sets of vertices $S_1, \ldots, S_k \subseteq V_G$, each of size at most l, that satisfy conditions 1–3 of the lemma. By condition 3, there exist pairwise vertex-disjoint trees $T_1, \ldots, T_k$ with $S_i \subseteq V_{T_i} \subseteq I(S_i)$ for $i = 1, \ldots, k$. By condition 1, we have $V_G \subseteq I(S_1) \cup \ldots \cup I(S_k)$. This means that there is a partition $X_1, \ldots, X_k$ of $V_G \setminus \bigcup_{i=1}^k V_{T_i}$, where some of the sets X_i can be empty, such that $X_i \subseteq I(S_i)$ for $i = 1, \ldots, k$. Let $W(x_i) = V_{T_i} \cup X_i$ for $i = 1, \ldots, k$. We claim that the sets $W(x_1), \ldots, W(x_k)$ form an H-witness structure of G. By definition, $W(x_1), \ldots, W(x_k)$ are pairwise disjoint, nonempty, and $W(x_1) \cup \ldots \cup W(x_k) = V_G$, i.e., they form a partition of V_G. It remains to show that these sets satisfy conditions (i) and (ii) of the definition of an H-witness structure.

(i) By definition, each tree T_i is connected. By Lemma 3, each node K of $\mathcal{T}_{S_i}$ contains a vertex of T_i. By definition, $X_i \subseteq I(S_i)$, which implies that for each $v \in X_i$, there is a node K in T_{S_i} such that $v \in K$. Because K is a clique in G, we then find that v is adjacent to at least one vertex of T_i. Therefore, each $W(x_i)$ induces a connected subgraph of G.

(ii) For the forward direction, suppose that $W(x_i)$ and $W(x_j)$ are two adjacent witness sets. Then there exist two vertices $u \in W(x_i)$ and $v \in W(x_j)$ such that $uv \in E_G$. Let K be a maximal clique that contains both u and v. Because $u \in W(x_i)$ and $v \in W(x_j)$, we find that K is a node of $\mathcal{T}_{S_i}$ and of $\mathcal{T}_{S_j}$, respectively. Hence, $V_{\mathcal{T}_{S_i}} \cap V_{\mathcal{T}_{S_j}} \neq \emptyset$, which means that $x_i x_j \in E_H$ by condition 2.

For the reverse direction, let x_i and x_j be two adjacent vertices in H. By condition 2, we find that $V_{\mathcal{T}_{S_i}} \cap V_{\mathcal{T}_{S_j}} \neq \emptyset$. Hence, there is a node $K \in V_{\mathcal{T}_{S_i}} \cap V_{\mathcal{T}_{S_j}}$. By Lemma 3, we deduce that K contains a vertex $u \in V_{T_i}$ and a vertex $v \in V_{T_j}$. Because K is a clique in G, this means that u and v are adjacent. Because $V_{T_i} \subseteq W(x_i)$ and $V_{T_j} \subseteq W(x_j)$, we obtain $u \in W(x_i)$ and $v \in W(x_j)$, respectively. Hence, $W(x_i)$ and $W(x_j)$ are adjacent. □

5.2 The Algorithm

We are now ready to describe our algorithm for Contractibility on chordal graphs.

Theorem 3. Contractibility *can be solved in* $n^{O(|V_H|^2)}$ *time on chordal graphs.*

Proof. Let G be a chordal graph on n vertices, and let H be a graph on k vertices with $V_H = \{x_1, \ldots, x_k\}$. If $k > n$ or the number of connected components of G and H are different, then return `No`. Suppose that G and H have $r > 1$ connected components $G_1, \ldots, G_r$ and $H_1, \ldots, H_r$, respectively. For each permutation $\langle i_1, \ldots, i_r \rangle$ of the ordered set $\langle 1, \ldots, r \rangle$, check whether H_{i_j} is a contraction of G_j for every $j \in \{1, \ldots, r\}$. Return `Yes` if this is the case for some permutation, and `No` otherwise. Hence, we may assume that G and H are connected.

Construct a clique tree $\mathcal{T}_G$ of G. If $\mathcal{T}_G$ has at least $k+1$ leaves, then consider each set $A \subseteq V_G$ with $|A| = k$, and continue with $G(A)$ instead of G. This is allowed due to Lemma 2. Note that a clique tree of $G(A)$ has at most $|A| = k$ leaves due to Lemma 1. Hence, we may assume that $\mathcal{T}_G$ has at most k leaves.

Consider each collection of pairwise disjoint sets $S_1, \ldots, S_k$, where each S_i is a subset of V_G with $1 \leq |S_i| \leq k$. For each collection $S_1, \ldots, S_k$, construct the subtrees $\mathcal{T}_{S_1}, \ldots, \mathcal{T}_{S_k}$ of $\mathcal{T}_G$ and the sets $I(S_1), \ldots, I(S_k)$, and test whether conditions 1–3 of Lemma 4 are satisfied. If so, the algorithm returns `Yes`; otherwise, it returns `No`. Correctness of this algorithm is an immediate consequence of Lemma 4.

We now analyze the running time. The number of permutations of the ordered set $\langle 1, \ldots, r \rangle$ is at most $r! \leq k!$. Constructing $\mathcal{T}_G$ takes linear time. The number of k-element subsets A of V_G is $n^{O(k)}$. The number of collections of sets $S_1, \ldots, S_k$ with $|S_i| \leq k$ for $i = 1, \ldots, k$ is $n^{O(k^2)}$. Constructing subtrees $\mathcal{T}_{S_1}, \ldots, \mathcal{T}_{S_k}$ of $\mathcal{T}_G$ and sets $I(S_1), \ldots, I(S_k)$, and testing if conditions 1–2 of Lemma 4 are satisfied, takes $n^{O(1)}$ time. As a result of Corollary 1, testing whether condition 3 of Lemma 4 is satisfied takes $n^{O(k^2)}$ time. Hence, the total running time is $n^{O(k^2)}$. Consequently, we can test in this running time whether a given chordal graph G contains a given graph H as a contraction. When the graph H, and hence k, is fixed, the running time is polynomial in n. □

The algorithm for Contractibility can be modified for Induced Minor, but it is easier to use the following observation. Let $P_1 \bowtie G$ denote the graph obtained from a graph G by adding a new vertex and making it adjacent to every vertex of G.

Lemma 5 ([22]). *Let G and H be two arbitrary graphs. Then G contains H as an induced minor if and only if $(P_1 \bowtie G)$ contains $(P_1 \bowtie H)$ as a contraction.*

Since $P_1 \bowtie G$ is chordal whenever G is chordal, we can combine Lemma 5 with Theorem 3 to obtain the following result, with the same running time as in the proof of Theorem 3.

Corollary 2. Induced Minor *can be solved in $n^{O(|V_H|^2)}$ time on chordal graphs.*

6 Induced Disjoint Paths and Induced Topological Minors

We start this section by showing that the Induced Disjoint Paths problem is polynomial-time solvable on chordal graphs. Recall that Disjoint Paths is NP-complete even on interval graphs [30], and that Induced Disjoint Paths is NP-complete on general graphs already when $k = 2$.

Let G be a graph that, together with terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$, constitutes an instance of Induced Disjoint Paths. We say that a set of paths $P_1, \ldots, P_k$ forms a *solution* of this instance if $P_1, \ldots, P_k$ are mutually induced and P_i is an (s_i, t_i)-path for $i = 1, \ldots, k$. Recall that it is sufficient to consider solutions where each path P_i is an induced path in G. Moreover, two different paths P_i and P_j can only intersect in terminals, and every edge in $G[\bigcup_{i=1}^{k} V(P_i)]$ belongs to some path P_i or is an edge between two terminal vertices belonging to different terminal pairs.

Theorem 4. *The* Induced Disjoint Paths *problem can be solved in $O(kn^3)$ time on chordal graphs.*

Proof. We apply dynamic programming to solve Induced Disjoint Paths on chordal graphs. Because our approach is similar to the one we used for Set-Restricted Disjoint Paths, we mainly explain the differences.

Let G be a chordal graph that together with terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$ forms an instance of Induced Disjoint Paths. If G is disconnected, then we check for each pair of terminals (s_i, t_i) whether s_i and t_i belong to the same connected component. If not, then we return `No`. Otherwise, we consider each connected component and its set of terminals separately. Hence, we may assume that G is connected.

As before, we construct in linear time a nice tree decomposition $(\mathcal{T}, \mathcal{X})$ of G with root X_r, such that each bag is a clique in G. We then apply our dynamic programming algorithm on $(\mathcal{T}, \mathcal{X})$. Recall that for a node $X_i \in V_{\mathcal{T}}$, we let $\mathcal{T}_i$ denote the subtree of $\mathcal{T}$ with root X_i that is induced by X_i and all its descendants, and we write $G_i = G[\bigcup_{j \in V_{\mathcal{T}_i}} X_j]$.

For each node X_i, we construct a table that stores a collection of records

$$\mathcal{R} = ((State_1, R_1), \ldots, (State_k, R_k)),$$

where $R_1, \ldots, R_k \subseteq X_i$ are ordered sets without common vertices except (possibly) terminals and $0 \leq |R_j| \leq 2$ for $j \in \{1, \ldots, k\}$, and where each $State_j$ can have one of the following four values:

Not initialized, *Started from* s, *Started from* t, or *Completed*.

These records correspond to the partial solution of Induced Disjoint Paths for G_i with exactly the same properties as the records stored in the tables for Set-Restricted Disjoint Paths. In particular, each R_j is the set of vertices of the (s_j, t_j)-path in X_i.

The tables are constructed and updated in a straightforward way using the following two observations: (i) every induced path contains at most two vertices of each clique, and (ii) every clique contains internal vertices of at most one path. The algorithm computes these tables for all nodes of $\mathcal{T}$, starting from the leaves. Finally, the table for the root X_r is constructed. The algorithm returns `Yes` if the table for X_r contains a record $\mathcal{R} = ((State_1, R_1), \ldots, (State_k, R_k))$ with $State_1 = \ldots = State_k = Completed$, and it returns `No` otherwise.

For the running time, it is sufficient to note that by observations (i) and (ii), there are at most kn^2 possibilities to include in a partial solution internal vertices of the paths that go through the vertices of X_i. Because $\mathcal{T}$ contains $O(n)$ nodes, this means that the total running time is $O(kn^3)$. □

Corollary 3. Induced Topological Minor *can be solved in* $O(|E_H| \cdot n^{|V_H|+3})$ *time on chordal graphs.*

Proof. Let G be a chordal graph on n vertices and H be a graph whose vertices are ordered as $x_1, \ldots, x_{|V_H|}$. Recall that G contains H as an induced topological minor if and only if G contains an induced subgraph isomorphic to a subdivision of H. In G we choose $|V_H|$ vertices that we order, say as $u_1, \ldots, u_{|V_H|}$. For each edge $x_i x_j \in E_H$ we define a terminal pair (u_i, u_j). This leads to a set of terminal pairs $T = \{(s_1, t_1), \ldots, (s_\ell, t_\ell)\}$ where $\ell = |E_H|$. Then G contains an induced subgraph isomorphic to a subdivision of H such that the isomorphism maps u_i to x_i for $i = 1, \ldots, |V_H|$ if and only if G contains a set of ℓ mutually induced paths $P_1, \ldots, P_\ell$, such that P_j has end-vertices s_j and t_j for $j = 1, \ldots, \ell$. Hence, we may apply Theorem 4 to solve the latter problem in $O(\ell n^3)$ time. If this does not yield a solution, then we choose another ordered set of u-vertices until we have considered them all. Because the number of such choices is $O(n^{|V_H|})$ and $\ell = |E_H|$, the result follows. □

7 Concluding Remarks

In this section, we give another application of the Set-Restricted Disjoint Paths problem. We show that Set-Restricted Disjoint Connected Subgraphs can be solved in polynomial time on interval graphs for every fixed k. A graph is an *interval graph* if intervals of the real line can be associated with its vertices in such a way that two vertices are adjacent if and only if their corresponding intervals overlap. Interval graphs constitute another important subclass of chordal graphs besides split graphs.

Proposition 2. *The* Set-Restricted Disjoint Connected Subgraphs *problem can be solved in* $n^{O(k)}$ *time on interval graphs.*

Proof. Let $G = (V, E)$ be an interval graph that together with k pairwise disjoint nonempty vertex subsets $S_1, \ldots, S_k$ with corresponding domains $U_1, \ldots, U_k$ for some $k \geq 1$ forms an instance of Set-Restricted Disjoint Connected Subgraphs.

We construct an interval representation of G; this can be done in linear time as shown by Booth and Lueker [5]. Given this representation, we say that a vertex u in a subset $V' \subseteq V$ is a *leftmost* vertex of V' if there is no vertex in V' whose associated interval contains a point placed on the real line before the interval of u starts. We define the notion of a *rightmost* vertex analogously. For $i = 1, \ldots, k$, let s_i and t_i be a leftmost and rightmost vertex of S_i, respectively. We define $U'_i = U_i \setminus \bigcup_{h \neq i} S_h$ for $i = 1, \ldots, k$.

We make the following observation. Let $1 \leq i \leq k$, and let P be an arbitrary path from s_i to t_i. Then, by our choice of s_i and t_i, we find that P *dominates* S_i, i.e., every vertex of S_i that is not on P is adjacent to a vertex of P. Moreover, our choice of s_i and t_i also implies that every connected subgraph that contains S_i contains a path from s_i to t_i that dominates S_i. Hence, the terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$ with domains $U'_1, \ldots, U'_k$, respectively, form an instance of Set-Restricted Disjoint Paths that is a `Yes`-instance of this problem if and only if $(G, S_1, \ldots, S_k, U_1, \ldots, U_k)$ is a `Yes`-instance of Set-Restricted Disjoint Connected Subgraphs. Because we can solve the first problem by Theorem 2 in $n^{O(k)}$ time, the result follows. □

We conclude with some open questions. Recall that Contractibility and Induced Minor are W[1]-hard with parameter $|V_H|$ on chordal graphs [19]. Is either of these problems in FPT on interval graphs? Is Set-Restricted Disjoint Connected Subgraphs in FPT with parameter $|S_1| + |S_2| + \cdots + |S_k|$ on chordal graphs? Even though an affirmative answer to the last question would not improve our results for Contractibility, Induced Minor and Induced Topological Minor, the question might be interesting in its own right.

References

1. R. Belmonte, P. A. Golovach, P. Heggernes, P. van 't Hof, M. Kamiński, and D. Paulusma. Finding contractions and induced minors in chordal graphs via disjoint paths. In: *Proceedings of ISAAC 2011*, LNCS 7074: 110–119, Springer.
2. R. Belmonte, P. Heggernes, and P. van 't Hof. Edge contractions in subclasses of chordal graphs. In: *Proceedings of TAMC 2011*, LNCS 6648: 528–539, Springer, 2011.
3. D. Bienstock. On the complexity of testing for odd holes and induced odd paths. *Discrete Mathematics*, 90: 85–92, 1991. See also Corrigendum. *Discrete Mathematics*, 102: 109, 1992.
4. J. R. S. Blair and B. W. Peyton. An introduction to chordal graphs and clique trees. In *Graph Theory and Sparse Matrix Computations*, IMA Volumes in Mathematics and its Applications 56: 1–29, Springer 1993.
5. K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13: 335–379, 1976.
6. A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph classes: a survey*. SIAM, 1999.
7. A. E. Brouwer and H. J.Veldman. Contractibility and NP-completeness. *Journal of Graph Theory*, 11: 71–79, 1987.
8. B. Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.
9. G. A. Dirac. On rigid circuit graphs. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 25: 71–76, 1961.

10. R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, 141:109–131, 1995.
11. J. Fiala, M. Kamiński, B. Lidický, and D. Paulusma. The k-in-a-path problem for claw-free graphs. *Algorithmica*, 62: 499-519, 2012.
12. J. Fiala, M. Kamiński and D. Paulusma. A note on contracting claw-free graphs. Manuscript, 2011.
13. M. R. Fellows, J. Kratochvíl, M. Middendorf, and F. Pfeiffer. The complexity of induced minors and related problems. *Algorithmica*, 13: 266–282, 1995.
14. J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer-Verlag, 2006.
15. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.
16. F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16: 47–56, 1974.
17. A. George, and J. W Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice Hall Professional Technical Reference (1981)
18. P. A. Golovach, M. Kamiński, and D. Paulusma, Contracting a chordal graph to a split graph or a tree. In: *Proceedings of MFCS 2011*, LNCS 6907: 339–350, Springer, 2011.
19. P. A. Golovach, M. Kamiński, D. Paulusma and D. M. Thilikos. Containment relations in split graphs. *Discrete Applied Mathematics*, 160: 155–163, 2012.
20. M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Annals of Discrete Mathematics 57, Elsevier (2004)
21. M. Grohe, K. Kawarabayashi, D. Marx, and P. Wollan. Finding topological subgraphs is fixed-parameter tractable. In: *Proceedings of STOC 2011*, 479–488.
22. P. van 't Hof, M. Kamiński, D. Paulusma, S. Szeider and D. M. Thilikos. On graph contractions and induced minors. *Discrete Applied Mathematics*, 160: 799–809, 2012.
23. P. van 't Hof, D. Paulusma,and G. J. Woeginger. Partitioning graphs in connected parts. *Theoretical Computer Science*, 410: 4834–4843, 2009.
24. M. Kamiński and D. M. Thilikos. Contraction checking in graphs on surfaces. In: *Proceedings of STACS 2012*, to appear.
25. T. Kloks. *Treewidth, Computations and Approximations.* LNCS 842, Springer, 1994.
26. B. Lévêque, D. Y. Lin, F. Maffray, and N. Trotignon. Detecting induced subgraphs. *Discrete Applied Mathematics*, 157: 3540–3551, 2009.
27. A. Levin, D. Paulusma, and G. J. Woeginger. The computational complexity of graph contractions I: polynomially solvable and NP-complete cases. *Networks*, 51: 178–189, 2008.
28. A. Levin, D. Paulusma, and G. J. Woeginger. The computational complexity of graph contractions II: two tough polynomially solvable cases. *Networks* 52: 32–56, 2008.
29. J. Matoušek and R. Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108: 343–364, 1992.
30. S. Natarajan and A. P. Sprague. Disjoint paths in circular arc graphs. *Nordic Journal on Computing* 3: 256–270, 1996.
31. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*, Oxford University Press, 2006.
32. H. N. de Ridder et al. Information System on Graph Classes and their Inclusions (ISGCI). `http://www.graphclasses.org`, 2001-2012.
33. N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63: 65–110, 1995.
34. C. Semple and M. Steel. *Phylogenetics.* Oxford University Press graduate series Mathematics and its Applications, 2003.
35. R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13: 66–579, 1984.