

Finding Contractions and Induced Minors in Chordal Graphs via Disjoint Paths*

Rémy Belmonte¹, Petr A. Golovach², Pinar Heggernes¹, Pim van 't Hof¹,
Marcin Kamiński³, and Daniël Paulusma²

¹ Department of Informatics, University of Bergen, Norway

{remy.belmonte, pinar.heggenes, pim.vanthof}@ii.uib.no

² School of Engineering and Computing Sciences, Durham University, United Kingdom

{petr.golovach, daniel.paulusma}@durham.ac.uk

³ Département d'Informatique, Université Libre de Bruxelles, Belgium

marcin.kaminski@ulb.ac.be

Abstract. The k -DISJOINT PATHS problem, which takes as input a graph G and k pairs of specified vertices (s_i, t_i) , asks whether G contains k mutually vertex-disjoint paths P_i such that P_i connects s_i and t_i , for $i = 1, \dots, k$. We study a natural variant of this problem, where the vertices of P_i must belong to a specified vertex subset U_i for $i = 1, \dots, k$. In contrast to the original problem, which is polynomial-time solvable for any fixed integer k , we show that this variant is NP-complete even for $k = 2$. On the positive side, we prove that the problem becomes polynomial-time solvable for any fixed integer k if the input graph is chordal. We use this result to show that, for any fixed graph H , the problems H -CONTRACTIBILITY and H -INDUCED MINOR can be solved in polynomial time on chordal graphs. These problems are to decide whether an input graph G contains H as a contraction or as an induced minor, respectively.

1 Introduction

We study algorithmic problems that involve deciding whether the structure of a graph H appears as a pattern within the structure of another graph G . Table 1 shows seven graph containment relations that can be obtained by combining vertex deletions (VD), edge deletions (ED), and edge contractions (EC). For example, a graph H is an *induced minor* of a graph G if H can be obtained from G by a sequence of graph operations, including vertex deletions and edge contractions, but not including 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.

With the exception of GRAPH ISOMORPHISM, all problems in Table 1 are known to be NP-complete when G and H are both input (cf. [16]). In search of tractability, it is common to fix the graph H as a part of the problem definition, and to consider only the graph G as input. We indicate this by adding “ H -” in front of the names of the decision problems. A celebrated result by Robertson and Seymour [17] states

* This work is supported by EPSRC (EP/G043434/1) and Royal Society (JP100692), and by the Research Council of Norway (197548/F20).

Containment Relation	VD	ED	EC	Decision Problem
minor	yes	yes	yes	MINOR
induced minor	yes	no	yes	INDUCED MINOR
contraction	no	no	yes	CONTRACTIBILITY
subgraph	yes	yes	no	SUBGRAPH ISOMORPHISM
induced subgraph	yes	no	no	INDUCED SUBGRAPH ISOMORPHISM
spanning subgraph	no	yes	no	SPANNING SUBGRAPH ISOMORPHISM
isomorphism	no	no	no	GRAPH ISOMORPHISM

Table 1. Seven known containment relations obtained by graph operations. The missing combination “no yes yes” corresponds to the minor relation if we allow an extra operation that removes isolated vertices.

that H -MINOR can be solved in cubic time for any fixed graph H . The problems H -SUBGRAPH ISOMORPHISM, H -INDUCED SUBGRAPH ISOMORPHISM, H -SPANNING SUBGRAPH ISOMORPHISM and H -GRAPH ISOMORPHISM can readily be solved in polynomial time for any fixed graph H . In contrast, there exist graphs H such that H -INDUCED MINOR and H -CONTRACTIBILITY are NP-complete. Although a complete complexity classification of these two problems is still not known, there are many partial results.

Fellows, Kratochvíl, Middendorf, and Pfeiffer [5] gave both polynomial-time solvable and NP-complete cases for the H -INDUCED MINOR problem on general input graphs. The smallest known NP-complete case is a graph H on 68 vertices [5]. A number of polynomial-time solvable and NP-complete cases for the H -CONTRACTIBILITY problem on general input graphs can be found in a series of papers started by Brouwer and Veldman [3], followed by Levin, Paulusma and Woeginger [14, 15], and van ’t Hof et al. [10]. The smallest NP-complete cases are when H is a path or a cycle on 4 vertices [3]. When it comes to input graphs with a particular structure, both H -INDUCED MINOR [5] and H -CONTRACTIBILITY [12] can be solved in polynomial time on planar graphs for any fixed graph H . The same is true when the input graphs are split graphs, as shown by Belmonte, Heggernes, and van ’t Hof [1] and Golovach et al. [9]. Finally, Golovach, Kamiński and Paulusma [8] showed that H -CONTRACTIBILITY and H -INDUCED MINOR are polynomial-time solvable on chordal graphs whenever H is a fixed tree or a fixed split graph. For any fixed H that is neither a tree nor a split graph, the computational complexity of these two problems on chordal graphs was left open.

In this paper, we solve this open problem by showing that H -CONTRACTIBILITY and H -INDUCED MINOR can be solved in polynomial time on chordal graphs, for any fixed graph H . In fact, our result implies algorithms with running time $|V_G|^{O(|V_H|^2)}$ for CONTRACTIBILITY and INDUCED MINOR on input pairs (G, H) where G is chordal. Our result is best possible in the sense that both these problems are NP-complete already on input pairs (G, H) , where G is a split graph and H is a threshold⁴ graph [1]. Moreover, both problems are $W[1]$ -hard parameterized by $|V_H|$ when G and H are both split graphs [9]. This means that it is unlikely that these two problems can be solved in

⁴ Threshold graphs form a subset of split graphs, which form a subset of chordal graphs.

time $f(|V_H|) |V_G|^{O(1)}$ on input pairs (G, H) that are split graphs, such that the function f is independent of $|V_G|$.

Our approach differs from previous approaches [1, 8, 9], as it is based on an application of the following problem. Here, a *terminal pair* in a graph $G = (V, E)$ is a specified pair of vertices s_i and t_i called *terminals*, and the *domain* of a terminal pair (s_i, t_i) is a specified subset $U_i \subseteq V$ containing both s_i and t_i . We say that an (s_i, t_i) -path and an (s_j, t_j) -path are *vertex-disjoint* if they have no common vertices except the vertices in $\{s_i, t_i\} \cap \{s_j, t_j\}$.

SET-RESTRICTED k -DISJOINT PATHS

Instance: A graph G , terminal pairs $(s_1, t_1), \dots, (s_k, t_k)$, and domains U_1, \dots, U_k .

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

Observe that the domains U_1, \dots, U_k can have common vertices. Furthermore, if we let every domain contain all vertices of G , we obtain the well-known problem k -DISJOINT PATHS. This problem can be solved in cubic time for any fixed k , as was shown by Robertson and Seymour [17]. In contrast to this result, we show in Section 3 that SET-RESTRICTED k -DISJOINT PATHS is NP-complete even for $k = 2$. However, we show in the same section that on chordal graphs SET-RESTRICTED k -DISJOINT PATHS can be solved in polynomial time for any fixed integer k . If k is part of the input, then this problem is NP-complete for chordal graphs, as Kammer and Tholey [11] showed that its special case DISJOINT PATHS remains NP-complete when restricted to this graph class.

We apply the result of Section 3 as follows. First, in the same section we generalize our polynomial-time algorithm from disjoint paths to disjoint trees. Then, in Section 4, we show how we can use this algorithm as a subroutine for solving H -CONTRACTIBILITY and H -INDUCED MINOR on chordal graphs.

2 Preliminaries

Let $G = (V, E)$ be a graph. If the vertex set and edge set of G are not specified, then we use V_G and E_G to denote these sets, respectively. The number of vertices in an input graph is denoted by n . A subset $U \subseteq V$ is a *clique* if there is an edge in G between any two vertices of U , and U is an *independent set* if there is no edge in G between any two vertices of U . A vertex is called *simplicial* if its neighbors form a clique. We write $G[U]$ to denote the subgraph of G induced by $U \subseteq V$, i.e., the graph on vertex set U and an edge between any two vertices whenever there is an edge between them in G . Two sets $U, U' \subseteq V$ are called *adjacent* if there exist vertices $u \in U$ and $u' \in U'$ such that $uu' \in E$.

The *edge contraction* of an edge uv in G deletes the vertices u and v from G , and replaces them by a new vertex adjacent to precisely those vertices to which u or v were adjacent. A graph H is a *contraction* of a graph G if H can be obtained by performing a series of edge contractions in G . An *H -witness structure* \mathcal{W} is a partition of V_G 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 ;

- (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 .

Clearly, H is a contraction of G if and only if G has an H -witness structure satisfying conditions (i) and (ii); H can be obtained by contracting each witness set into a single vertex.

A *tree decomposition* of a graph G is a pair $(\mathcal{T}, \mathcal{X})$, where \mathcal{X} is a collection of subsets of V_G , 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. First, $\bigcup_{X \in \mathcal{X}} X = V_G$. Second, for each edge $uv \in E_G$, there is a bag $X \in \mathcal{X}$ with $u, v \in X$. Third, for each $x \in V_G$, the set of nodes containing x forms a subtree of \mathcal{T} .

A graph is *chordal* if it does not contain a chordless cycle on at least four vertices as an induced subgraph. It is easy to see that the class of chordal graphs is closed under edge contractions. Chordal graphs can be recognized in linear time [18]. Every chordal graph contains at most n maximal cliques, and, if it is not a complete graph, at least two non-adjacent simplicial vertices [4]. 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 [7]. Such a tree decomposition is called a *clique tree* and can be constructed in linear time [2].

3 Set-Restricted Disjoint Paths

We start with the following result, the proof of which can be found in Appendix A.

Theorem 1. *The SET-RESTRICTED 2-DISJOINT PATHS problem is NP-complete.*

We now apply dynamic programming to prove that SET-RESTRICTED k -DISJOINT PATHS can be solved in polynomial time on chordal graphs for any fixed integer k . The first key observation is that the existence of k disjoint paths is equivalent to the existence of k disjoint *induced* paths. The second key observation is that any induced path contains at most two vertices of any clique. Our algorithm can easily be modified to produce the required paths (if they exist).

Kloks [13] 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 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 $X = X' = X''$.

Since each chordal graph G has a clique tree, it also has a nice tree decomposition with the additional property that each bag is a (not necessary maximal) clique in G .

Now we are ready to describe our algorithm for SET-RESTRICTED k -DISJOINT PATHS. Let k be a positive integer, and let G be a chordal graph with k pairs of *terminal* vertices $(s_1, t_1), \dots, (s_k, t_k)$ that have domains U_1, \dots, U_k , respectively. If G is disconnected, 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 without loss of generality that G is connected. We then proceed as follows.

First, we construct in linear time a nice tree decomposition $(\mathcal{T}, \mathcal{X})$ with root X_r for G such that each bag is a clique in G . Next, we apply a dynamic programming algorithm over $(\mathcal{T}, \mathcal{X})$. We describe what we store in tables corresponding to the nodes of \mathcal{T} . For any node $X_i \in V_{\mathcal{T}}$, we denote by \mathcal{T}_i the subtree of \mathcal{T} with root X_i that is induced by the descendants of X_i . We also define the subgraph $G_i = G[\bigcup_{j \in V_{\mathcal{T}_i}} X_j]$. For

a node X_i , the table stores the records $\mathcal{R} = ((\text{State}_1, R_1), \dots, (\text{State}_k, R_k))$, where each State_j can have one of the four values:

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

and $R_1, \dots, R_k \subseteq X_i$ are ordered sets without common vertices except (possibly) terminals, $R_j \subseteq U_j$, and $0 \leq |R_j| \leq 2$ for $j \in \{1, \dots, k\}$. These records correspond to the partial solution of SET-RESTRICTED k -DISJOINT PATHS for G_i with the following properties.

- If $\text{State}_j = \text{Not initialized}$, 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 $\text{State}_j = \text{Started from s}$, 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 $\text{State}_j = \text{Started from t}$, 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 $\text{State}_j = \text{Completed}$, 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.

Observe that since we are solving the decision problem, we do not keep (s_j, t_j) -paths or their subpaths themselves. The details of how the tables are constructed and updated are given in Appendix B.

Theorem 2. *The SET-RESTRICTED k -DISJOINT PATHS problem can be solved on chordal graphs in time $n^{O(k)}$.*

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 any clique. For the estimation of the running time, it is sufficient 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 and hence tables, our algorithm runs in $n^{O(k)}$ time. \square

For our purposes, we need to generalize Theorem 2 from paths to trees. Instead of terminal pairs (s_i, t_i) we speak of *terminal sets* S_i , each contained in a subset U_i called the *domain* of S_i . Moreover, we say that a tree T_i containing S_i is an S_i -tree. Then an S_i -tree and an S_j -tree are *vertex-disjoint* if they have no common vertices except the vertices in $S_i \cap S_j$. The SET-RESTRICTED k -DISJOINT TREES problem takes as input a graph G , terminal sets S_1, \dots, S_k , and domains U_1, \dots, U_k , and asks whether G contains mutually vertex-disjoint trees T_1, \dots, T_k such that T_i is an S_i -tree containing only vertices from U_i for $i = 1, \dots, k$.

Corollary 1. *The SET-RESTRICTED k -DISJOINT TREES problem can be solved on chordal graphs in $n^{O(p)}$ time, where $p = \sum_{i=1}^k |S_i|$ is the total size of the terminal sets.*

Proof. Let G be a chordal graph on n vertices with terminal sets S_1, \dots, S_k and corresponding domains U_1, \dots, U_k . Let $p_i = |S_i|$ for $i = 1, \dots, k$. Observe that any 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 . By contracting all degree 2 vertices in T_i , we obtain a tree with at most $2p_i - 2$ vertices, and consequently, at most $2p_i - 3$ edges. Every such edge corresponds to a path in G . By this observation, we can do as follows.

We choose k mutually disjoint sets X_1, \dots, X_k with $X_i \subseteq U_i \setminus S_i$ and $|X_i| \leq p_i - 2$ for $i = 1, \dots, k$. Then, for each $S_i \cup X_i$, we choose a tree 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 this tree gives us a terminal pair (s, t) with domain U_i . Let $(s_1, t_1), \dots, (s_{k'}, t_{k'})$ be the total set of terminal pairs with domains $U_1, \dots, U_{k'}$ that is obtained after considering all sets $S_i \cup X_i$ in this way. We then solve the SET-RESTRICTED k' -DISJOINT PATHS problem for this instance. Note that $k' \leq \sum_{i=1}^k (2p_i - 3)$.

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, \dots, 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 $n^{O(p)} \cdot p^{2p_1 + \dots + 2p_k} \cdot n^{O(2p)} = n^{O(p)}$. \square

4 Contractions and Induced Minors in Chordal Graphs

First, in Section 4.1 below, we give a structural characterization of chordal graphs that contain a fixed graph H as a contraction. Then, in Section 4.2, we present our polynomial-time algorithm for solving H -CONTRACTIBILITY on chordal graphs for any fixed graph H , and show how it can be used to solve H -INDUCED MINOR as well.

4.1 Properties

Throughout Section 4.1, let G be a connected chordal graph, let \mathcal{T}_G be a clique tree of G , and let H be a graph with $V_H = \{x_1, \dots, x_k\}$. 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 any 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 any 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 \mathcal{W} of G . For each $i \in \{1, \dots, k\}$, we choose an arbitrary vertex $a_i \in W(x_i)$, and let $A = \{a_1, \dots, a_k\}$. 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. \square

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 . \square

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 mutually disjoint nonempty sets of vertices $S_1, \dots, S_k \subseteq V_G$, each of size at most l , such that

1. $V_G \subseteq I(S_1) \cup \dots \cup I(S_k)$;
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 mutually vertex-disjoint trees T_1, \dots, 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 $i = 1, \dots, k$, 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 $i = 1, \dots, k$. 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 $W(x_1), \dots, W(x_k)$ are mutually disjoint, the trees T_1, \dots, T_k are mutually vertex-disjoint. Moreover, as we already deduced, $S_i \subseteq V_{T_i} = W(x_i) \subseteq I(S_i)$ for $i = 1, \dots, k$.

Now suppose that there are mutually disjoint nonempty sets of vertices $S_1, \dots, S_k \subseteq V_G$, each of size at most l , that satisfy conditions 1–3 of the lemma. By condition 3, there exist mutually vertex-disjoint trees T_1, \dots, T_k with $S_i \subseteq V_{T_i} \subseteq I(S_i)$ for $i = 1, \dots, k$. By condition 1, we have $V_G \subseteq I(S_1) \cup \dots \cup I(S_k)$. This means that there is a partition X_1, \dots, 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 $i = 1, \dots, k$. Let $W(x_i) = V_{T_i} \cup X_i$ for $i = 1, \dots, k$. We claim that the sets $W(x_1), \dots, W(x_k)$ form an H -witness structure of G . By definition, $W(x_1), \dots, W(x_k)$ are mutually disjoint, nonempty, and $W(x_1) \cup \dots \cup W(x_k) = V_G$, 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 \mathcal{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. \square

4.2 The Algorithm

We are now ready to describe our algorithm for H -CONTRACTIBILITY on chordal graphs, for any fixed graph H . Although the presented algorithm solves the decision problem, it can be modified to produce an H -witness structure if one exists.

Theorem 3. *For any fixed graph H , the H -CONTRACTIBILITY problem can be solved in polynomial time on chordal graphs.*

Proof. Let G be a chordal graph on n vertices and let H be a graph on k vertices with $V_H = \{x_1, \dots, x_k\}$. 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, \dots, G_r and H_1, \dots, H_r , respectively. For each permutation $\langle i_1, \dots, i_r \rangle$ of the ordered set $\langle 1, \dots, r \rangle$, check whether H_{i_j} is a contraction of G_j for every $j \in \{1, \dots, r\}$. 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 mutually disjoint sets S_1, \dots, S_k , where each S_i is a subset of V_G with $1 \leq |S_i| \leq k$. For each collection S_1, \dots, S_k , construct the subtrees $\mathcal{T}_{S_1}, \dots, \mathcal{T}_{S_k}$ of \mathcal{T}_G and the sets $I(S_1), \dots, I(S_k)$, 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 $\langle 1, \dots, r \rangle$ 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, \dots, S_k with $|S_i| \leq k$ for $i = 1, \dots, k$ is $n^{O(k^2)}$. Constructing subtrees $\mathcal{T}_{S_1}, \dots, \mathcal{T}_{S_k}$ of \mathcal{T}_G and sets $I(S_1), \dots, I(S_k)$, 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 . \square

The algorithm for H -CONTRACTIBILITY can be modified for H -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 ([10]). *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. *For any fixed graph H , the H -INDUCED MINOR problem can be solved in polynomial time on chordal graphs.*

5 Concluding Remarks

Kammer and Tholey [11] improved the running time of DISJOINT PATHS from cubic for general graphs [17] to linear for chordal graphs. Is SET-RESTRICTED DISJOINT PATHS fixed-parameter tractable on chordal graphs, when parameterized by k ?

Recall that the problems CONTRACTIBILITY and INDUCED MINOR are $W[1]$ -hard for pairs (G, H) that are chordal graphs, when parameterized by $|V_H|$ [9]. Is either of these problems fixed-parameter tractable on pairs (G, H) that are interval graphs, which constitute an important subclass of chordal graphs?

References

1. R. Belmonte, P. Heggernes, and P. van 't Hof. Edge contractions in subclasses of chordal graphs. In: *Proceedings of TAMC 2011*, LNCS 6648, pp. 528–539, Springer, 2011.
2. J.R.S. Blair and B. Peyton. An introduction to chordal graphs and clique trees. *Graph Theory and Sparse Matrix Computation*, 56: 1–29, 1993.
3. A.E. Brouwer and H.J. Veldman. Contractibility and NP-completeness. *Journal of Graph Theory*, 11: 71–79, 1987.
4. G.A. Dirac. On rigid circuit graphs. *Ann. Math. Sem. Univ. Hamburg*, 25: 71–76, 1961.
5. M.R. Fellows, J. Kratochvíl, M. Middendorf, and F. Pfeiffer. The complexity of induced minors and related problems. *Algorithmica*, 13: 266–282, 1995.
6. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
7. F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16: 47–56, 1974.
8. 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, Springer, to appear.
9. P.A. Golovach, M. Kamiński, D. Paulusma and D.M. Thilikos. Containment relations in split graphs. Manuscript.
10. P. van 't Hof, M. Kamiński, D. Paulusma, S. Szeider and D.M. Thilikos. On graph contractions and induced minors. *Discrete Applied Mathematics*, to appear.
11. F. Kammer, T. Tholey. The k -disjoint paths problem on chordal graphs. In *Proceedings of WG 2009*, LNCS 5911, pp. 190–201, Springer, 2010.
12. M. Kamiński, D. Paulusma and D.M. Thilikos. Contractions of planar graphs in polynomial time. In: *Proceedings of ESA 2010*, LNCS 6346, pp. 122–133, Springer, 2010.
13. T. Kloks. *Treewidth, Computations and Approximations*. LNCS 842, Springer, Berlin, 1994.
14. 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.
15. 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.
16. J. Matoušek and R. Thomas. On the complexity of finding iso- and other morphisms for partial k -trees. *Discrete Mathematics*, 108: 343–364, 1992.
17. N. Robertson and P.D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63: 65–110, 1995.
18. 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: 66–79, 1984.

A The Proof of Theorem 1

Theorem 1. *The SET-RESTRICTED 2-DISJOINT PATHS problem is NP-complete.*

Proof. We reduce from the NP-complete 3-SATISFIABILITY problem [6]. 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-SATISFIABILITY with variables x_1, \dots, x_n and clauses C_1, \dots, 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, \dots, v_n and edges sv_0 and v_nt .
 - For $i = 1, \dots, n$, add vertices $x_i^{(1)}, x_i^{(2)}, \bar{x}_i^{(1)}, \bar{x}_i^{(2)}, y_i, \bar{y}_i$ and edges $v_{i-1}x_i^{(1)}, x_i^{(1)}y_i, y_ix_i^{(2)}, x_i^{(2)}v_i, v_{i-1}\bar{x}_i^{(1)}, \bar{x}_i^{(1)}\bar{y}_i, \bar{y}_i\bar{x}_i^{(2)}, \bar{x}_i^{(2)}v_i$.
Let $Q_i = v_{i-1}x_i^{(1)}y_ix_i^{(2)}v_i$ and $\bar{Q}_i = v_{i-1}\bar{x}_i^{(1)}\bar{y}_i\bar{x}_i^{(2)}v_i$.
 - Add $m + 1$ vertices u_0, \dots, 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 $z = \bar{x}_i$, then add edges $u_{j-1}\bar{x}_i^{(1)}, u_j\bar{x}_i^{(1)}$ if z is the first occurrence of the literal \bar{x}_i in Φ , and edges $u_{j-1}\bar{x}_i^{(2)}, u_j\bar{x}_i^{(2)}$ if z is the second occurrence.
- Let $R_j(z)$ be the obtained (u_{j-1}, u_j) -path of length two.
- Let $U_1 = V_G \setminus \{u_0, \dots, u_m\}$.
 - Let $U_2 = \{s, t\} \cup \{u_0, \dots, u_m\} \cup \{x_1^{(1)}, \dots, x_n^{(1)}\} \cup \{x_1^{(2)}, \dots, x_n^{(2)}\}$.

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, \dots, x_n have a truth assignment that satisfies Φ . We construct P_1 as follows. For each $i \in \{1, \dots, n\}$, we choose Q_i if $x_i = \text{false}$, and \bar{Q}_i if $x_i = \text{true}$. 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 $j \in \{1, \dots, m\}$, the clause C_j can be assumed to contain a literal $z = \text{true}$, 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 $x_i = \text{true}$ implying that \bar{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.

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 $i \in \{1, \dots, n\}$, either Q_i or \bar{Q}_i is a subpath of P_1 . If Q_i is a subpath of P_1 , then we set $x_i = \text{false}$, and $x_i = \text{true}$

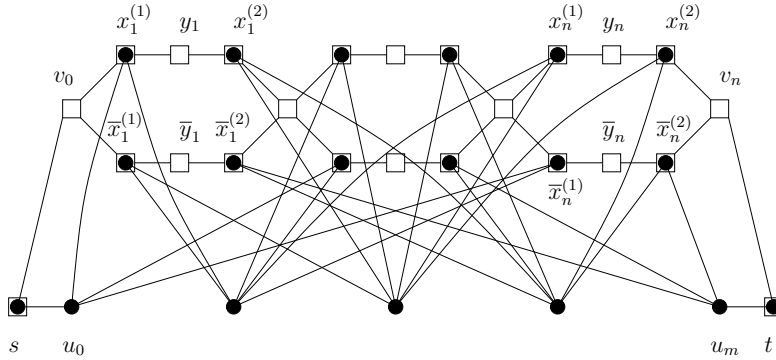


Fig. 1. The graph G ; the vertices of U_1 and U_2 are shown by squares and bullets, respectively.

otherwise. Note that for each $j \in \{1, \dots, m\}$, 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 \bar{Q}_i is a subpath of P_1 . Hence, $x_i = \text{true}$, and consequently $z = \text{true}$. Similarly, if $z = \bar{x}_i$ for some variable x_i , then Q_i is a subpath of P_1 and $x_i = \text{false}$, and consequently, $z = \text{true}$. Hence, each clause C_j is satisfied by this truth assignment, and $\Phi = \text{true}$, as desired. This completes the proof of Theorem 1. \square

B The details of the algorithm for SET-RESTRICTED DISJOINT PATHS on chordal graphs

The details of how the tables are constructed and updated are given below.

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

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

If u is not a terminal, then $State_j = \text{Not initialized}$ for $j = 1, \dots, k$, and either $R_j = \emptyset$ for all $j \in \{1, \dots, k\}$, 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 $\mathcal{R}' = ((State'_1, R'_1), \dots, (State'_k, R'_k))$ from the table for $X_{i'}$, we either modify \mathcal{R}' and include the modified record $\mathcal{R} = ((State_1, R_1), \dots, (State_k, R_k))$ 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 $State'_j = \text{Not initialized}$, then $State_j = \text{Started from } s$; 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 $State'_j = \text{Started from } t$, then $State_j = \text{Completed}$; if $R'_j = \langle z \rangle$, then $R_j = \langle z, u \rangle$;
- if $u = t_j$ and $State'_j = \text{Not initialized}$, then $State_j = \text{Started from } t$; 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 $State'_j = \text{Started from } s$, then $State_j = \text{Completed}$; 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 $j \in \{1, \dots, k\}$ with $u \in U_j$, we consider the records $\mathcal{R}' = ((State'_1, R'_1), \dots, (State'_k, R'_k))$ from the table for $X_{i'}$ with $|R'_j| \leq 1$, and do the following:

- if $State'_j = \text{Not initialized}$ 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 $l \in \{1, \dots, k\}, l \neq j$, $State_j = State'_j$, and $R_j = \langle z, u \rangle$ for the first record and $R_j = \langle u, z \rangle$ for the second;
- if $State'_j = \text{Not initialized}$ 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 $l \in \{1, \dots, k\}, l \neq j$, $State_j = State'_j$, and $R_j = \langle u \rangle$;

- if $State'_j = \text{Started from } s$ 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 $l \in \{1, \dots, k\}, l \neq j$, $State_j = State'_j$, and $R_j = \langle z, u \rangle$;
- if $State'_j = \text{Started from } t$ 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 $l \in \{1, \dots, k\}, l \neq j$, $State_j = State'_j$, 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 $\mathcal{R}' = ((State'_1, R'_1), \dots, (State'_k, R'_k))$ from the table for $X_{i'}$ with $u \notin R'_j$ for all $j \in \{1, \dots, k\}$ is included in the table for X_i . For each record $\mathcal{R}' = ((State'_1, R'_1), \dots, (State'_k, R'_k))$ from the table for $X_{i'}$ such that $u \in R'_j$ for some $j \in \{1, \dots, k\}$, we either modify it and include the modified record $\mathcal{R} = ((State_1, R_1), \dots, (State_k, R_k))$ in the table for X_i , or we discard \mathcal{R}' , using the following rules:

- for all $j \in \{1, \dots, k\}$, $State_j = State'_j$;
- for $j \in \{1, \dots, k\}$, if $u \notin R'_j$, then $R_j = R'_j$;
- for $j \in \{1, \dots, k\}$, if $u \in R'_j$ and $State_j = \text{Started from } s$, 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 $j \in \{1, \dots, k\}$, if $u \in R'_j$ and $State_j = \text{Started from } t$, 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 $j \in \{1, \dots, k\}$, if $u \in R'_j$ and $State_j = \text{Not initialized}$, then we discard the record;
- for $j \in \{1, \dots, k\}$, if $u \in R'_j$ and $State_j = \text{Completed}$, 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 $\mathcal{R}' = ((State'_1, R'_1), \dots, (State'_k, R'_k))$ and $\mathcal{R}'' = ((State''_1, R''_1), \dots, (State''_k, R''_k))$ from the tables for $X_{i'}$ and $X_{i''}$, respectively, such that $R'_j = R''_j$ for all $j \in \{1, \dots, k\}$, we construct the record $\mathcal{R} = ((State_1, R_1), \dots, (State_k, R_k))$ and include it in the table for X_i :

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

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 $\mathcal{R} = ((State_1, R_1), \dots, (State_k, R_k))$ with $State_1 = \dots = State_k = \text{Completed}$, and it returns No otherwise.