

Edge Contractions in Subclasses of Chordal Graphs^{*}

Rémy Belmonte, Pinar Heggenes, and Pim van 't Hof

Department of Informatics, University of Bergen,
P.O. Box 7803, N-5020 Bergen, Norway.
{remy.belmonte,pinar.heggenes,pim.vanthof}@ii.uib.no

Abstract. Modifying a given graph to obtain another graph is a well-studied problem with applications in many fields. Given two input graphs G and H , the CONTRACTIBILITY problem is to decide whether H can be obtained from G by a sequence of edge contractions. This problem is known to be NP-complete already when both input graphs are trees of bounded diameter. We prove that CONTRACTIBILITY can be solved in polynomial time when G is a trivially perfect graph and H is a threshold graph, thereby giving the first classes of graphs of unbounded treewidth and unbounded degree on which the problem can be solved in polynomial time. We show that this polynomial-time result is in a sense tight, by proving that CONTRACTIBILITY is NP-complete when G and H are both trivially perfect graphs, and when G is a split graph and H is a threshold graph. If the graph H is fixed and only G is given as input, then the problem is called H -CONTRACTIBILITY. This problem is known to be NP-complete on general graphs already when H is a path on four vertices. We show that, for any fixed graph H , the H -CONTRACTIBILITY problem can be solved in polynomial time if the input graph G is a split graph.

1 Introduction

The problem of deciding whether a given graph can be obtained from another given graph by contracting edges is motivated from Hamiltonian graph theory and graph minor theory, and it has applications in computer graphics and cluster analysis [12]. This problem has recently attracted increasing interest, in particular when restrictions are imposed on the input graphs [3, 10–13]. We continue this line of research with new polynomial-time and NP-completeness results.

For a fixed graph H , the H -CONTRACTIBILITY problem is to decide whether H can be obtained from an input graph G by a sequence of edge contractions. This problem is closely related to the well-known H -MINOR CONTAINMENT problem, which is the problem of deciding whether H can be obtained from a *subgraph* of G by contracting edges. A celebrated result by Robertson and Seymour [17] shows that H -MINOR CONTAINMENT can be solved in polynomial time on general graphs for any fixed H . As a contrast, H -CONTRACTIBILITY is

^{*} This work has been supported by the Research Council of Norway.

NP-complete already for very simple fixed graphs H , such as a path or a cycle on four vertices [3]. The version of the problem where both graphs are given as input, called CONTRACTIBILITY, is NP-complete on trees of bounded diameter, as well as on trees all whose vertices but one have degree at most 5 [16].

Given these hardness results, it is perhaps not surprising that hardly any positive results are known on the CONTRACTIBILITY problem. So far, CONTRACTIBILITY is known to be solvable in polynomial time only when G has bounded treewidth and H has bounded degree [16]. A few more positive results are known on the H -CONTRACTIBILITY problem. For example, for every fixed graph H on at most 5 vertices, H -CONTRACTIBILITY can be solved on general graphs in polynomial time when H has a dominating vertex, and it is NP-complete otherwise [12, 13]. However, it is known that for larger fixed graphs H , the presence of a dominating vertex in H is not a guarantee for polynomial-time solvability of the problem [10]. Very recently, Kamiński, Paulusma and Thilikos [11] showed that H -CONTRACTIBILITY can be solved in polynomial time on planar input graphs for every fixed H .

In this paper, we study the CONTRACTIBILITY and H -CONTRACTIBILITY problems on subclasses of chordal graphs. Chordal graphs constitute one of the most famous graph classes, with a large number of practical applications (see e.g., [6, 7, 18]). Edge contractions preserve the property of being chordal. Since trees are chordal graphs, it follows from the above-mentioned hardness result on trees that CONTRACTIBILITY is NP-complete when G and H are both chordal. We show that the problem remains NP-complete even when G and H are both trivially perfect graphs or both split graphs. Note that trees are neither trivially perfect nor split. Trivially perfect graphs and split graphs are two unrelated subclasses of chordal graphs, and both classes are well-studied with several theoretical applications [2, 7]. These two classes share a common subclass called threshold graphs, which is another well-known subclass of chordal graphs [15]. We prove that CONTRACTIBILITY remains NP-complete even when G is split and H is threshold. On the positive side, we show that CONTRACTIBILITY can be solved in polynomial time when G is trivially perfect and H is threshold. This result can be considered tight by the above-mentioned hardness results. For H -CONTRACTIBILITY, we give a polynomial-time algorithm when G is a split graph and H is an arbitrary fixed graph. The results of this paper are summarized in Table 1.

On the way to obtain our results, we show that the problems CONTRACTIBILITY and INDUCED SUBGRAPH ISOMORPHISM are equivalent on connected trivially perfect graphs. Hence our results imply that the latter problem is NP-complete on connected trivially perfect graphs, and that this problem can be solved in polynomial time when G is trivially perfect and H is threshold. We would like to mention that INDUCED SUBGRAPH ISOMORPHISM is known to be NP-complete on split graphs and on cographs [4]. Trivially perfect graphs constitute a subclass of cographs, and threshold graphs are both cographs and split graphs. Hence our results tighten previously known hardness results on IN-

G	H	Complexity
Trivially perfect (i)	Trivially perfect (i)	NP-complete
Trivially perfect (i)	Threshold (i)	Polynomial
Trivially perfect (i)	Trivially perfect (f)	Polynomial
Threshold (i)	Arbitrary (i)	Linear
Split (i)	Threshold (i)	NP-complete
Split (i)	Arbitrary (f)	Polynomial

Table 1. The complexity of deciding whether G can be contracted to H , according to our results; (i) stands for “part of the input”, (f) stands for “fixed”.

DUCEDED SUBGRAPH ISOMORPHISM. The relationships between the graph classes mentioned in this paper are given in Figure 1.

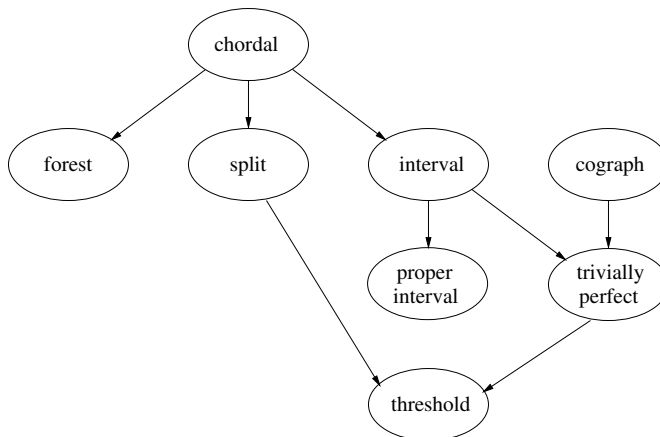


Fig. 1. The graph classes mentioned in this paper, where \rightarrow represents the \supset relation.

2 Preliminaries

All graphs considered in this paper are undirected, finite and simple. For a graph G , we use $V(G)$ and $E(G)$ to denote the set of vertices and set of edges of G , respectively. Let G be a graph, and let $V = V(G)$ and $E = E(G)$. For a vertex v in G , the set $N_G(v) = \{w \in V \mid vw \in E\}$, consisting of all the neighbors of v in G , is called the *neighborhood* of v . The set $N_G[v] = N_G(v) \cup \{v\}$ is the *closed neighborhood* of v . We omit subscripts when there is no ambiguity. The *degree* of a vertex v is $d(v) = |N(v)|$. An ordering $\alpha = (v_1, v_2, \dots, v_n)$ of the vertices of a graph G is called a *non-increasing degree ordering* of G if $d(v_1) \geq d(v_2) \geq \dots \geq d(v_n)$. A vertex is called *isolated* if its degree is 0. If

$N[v] = V$, then we say that v is a *universal vertex* of G . A *path* in G is a sequence of distinct vertices $P = u_1u_2 \cdots u_p$, where u_iu_{i+1} is an edge of G for every $i = 1, \dots, p-1$. We say that P is a path *between* u_1 and u_p , which are called the *end vertices* of P . If u_1u_p is an edge as well we obtain a *cycle*. A *forest* is a graph without cycles, and a *tree* is a connected forest. A vertex in a tree is called a *leaf* if it has degree 1. A *rooted tree* is a tree in which a vertex gets the special role called the *root*.

A graph is *connected* if there is a path between every pair of vertices. A maximal connected subgraph of a graph is called a *connected component*. A connected component of a graph is called *nontrivial* if it contains at least one edge. For any set $S \subseteq V$, we write $G[S]$ to denote the subgraph of G *induced* by S . We write $G-v$ to denote the graph $G[V \setminus \{v\}]$. The set S is said to be *connected* if $G[S]$ is connected. We say that two disjoint sets $S, S' \subseteq V$ are *adjacent* if there exist vertices $s \in S$ and $s' \in S'$ that are adjacent. A subset $S \subseteq V$ is a *clique* if all vertices in S are pairwise adjacent, and S is an *independent set* if no two vertices of S are adjacent. An *isomorphism* from a graph G to a graph H is a bijection $\varphi : V(G) \rightarrow V(H)$ such that $uv \in E(G)$ if and only if $\varphi(u)\varphi(v) \in E(H)$. We say that G is *isomorphic* to H if there exists an isomorphism from G to H . The INDUCED SUBGRAPH ISOMORPHISM problem is to decide, given two graphs G and H , whether G has an induced subgraph that is isomorphic to H . We say that two rooted trees T_1 and T_2 are isomorphic if there is an isomorphism from T_1 to T_2 that maps the root of T_1 to the root of T_2 .

The *contraction* of edge uv in G removes u and v from G , and replaces them by a new vertex, which is made adjacent to precisely those vertices that were adjacent to at least one of the vertices u and v . Instead of speaking of the contraction of edge uv , we sometimes say that a vertex u is *contracted onto* v if the new vertex resulting from the contraction is still called v . We write G/uv to denote the graph obtained from G by contracting the edge uv . We say that a graph G can be *contracted* to a graph H , or is *H -contractible*, if H is isomorphic to a graph that can be obtained from G by a sequence of edge contractions. Let $S \subseteq V(G)$ be a connected set. If we repeatedly contract edges in $G[S]$ until only one vertex of $G[S]$ remains, we say that we *contract S into a single vertex*. Let H be a graph with vertex set $\{h_1, \dots, h_{|V(H)|}\}$. Saying that a graph G can be contracted to H is equivalent to saying that G has a so-called *H -witness structure* \mathcal{W} , which is a partition of $V(G)$ into *witness sets* $W(h_1), \dots, W(h_{|V(H)|})$, such that each witness set induces a connected subgraph of G , and such that for every two vertices $h_i, h_j \in V(H)$, the corresponding witness sets $W(h_i)$ and $W(h_j)$ are adjacent in G if and only if h_i and h_j are adjacent in H . By contracting each of the witness sets into a single vertex, we obtain a graph which is isomorphic to H . See Figure 2 for an example that shows that, in general, an H -witness structure of G is not uniquely defined. For any subset $S \subseteq V(H)$, we write $W(S)$ to denote the set of vertices of G that are contained in a witness set $W(v)$ for some $v \in S$, i.e., $W(S) = \cup_{v \in S} W(v)$.

Cographs are the graphs that do not contain a path on four vertices as an induced subgraph. *Interval graphs* are the intersection graphs of intervals of a

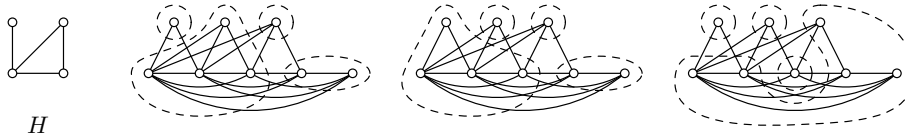


Fig. 2. Three different H -witness structures of a threshold graph.

line, and they form a subclass of chordal graphs. *Chordal graphs* are the graphs without induced cycles of length more than 3.

Trivially perfect graphs have various characterizations [2, 7, 8, 21]. For our purposes, it is convenient to use the following characterization as a definition. A graph G is *trivially perfect* if and only if each connected induced subgraph of G contains a universal vertex [19, 20]. Let $\alpha = (v_1, v_2, \dots, v_n)$ be an ordering of the vertices of a trivially perfect graph G . If α has the property that v_i is universal in a connected component of $G[\{v_i, v_{i+1}, \dots, v_n\}]$ for $i = 1, \dots, n$, then α is called a *universal-in-a-component ordering (uco)*. A graph is trivially perfect if and only if it has a uco, and if and only if every non-increasing degree ordering is a uco [8, 21]. Consequently, for every edge uv in a trivially perfect graph, either $N[u] \subseteq N[v]$ or $N[v] \subseteq N[u]$ [21].

Every rooted tree T defines a connected trivially perfect graph, which is obtained by adding tree edges to T so that every path between the root and a leaf becomes a clique. In fact, all connected trivially perfect graphs can be created this way, and there is a bijection between rooted trees and connected trivially perfect graphs [21]. Given a connected trivially perfect graph G , a rooted tree T_G corresponding to G , which we call a *uco-tree* of G , can be obtained in the following way. If G is a single vertex, then T_G is this vertex. Otherwise, take a universal vertex v of G , make it the root of T_G , and delete it from G . In the remaining graph, for each connected component G' , build a uco-tree $T_{G'}$ of G' recursively and make v the parent of the root of $T_{G'}$. All rooted trees that can be obtained from a connected trivially perfect graph in this way are isomorphic, and hence T_G is unique for every connected trivially perfect graph G . If G is disconnected, then it has a *uco-forest*, which is the disjoint union of the uco-trees of the connected components of G .

A graph G is a *split graph* if its vertex set can be partitioned into a clique C and an independent set I , where (C, I) is called a *split partition* of G . If C is not a maximum clique, then there is a vertex $v \in I$ that is adjacent to every vertex of C . In this case, $C' = C \cup \{v\}$ is a maximum clique, and $(C', I \setminus \{v\})$ is also a split partition of G . In this paper, unless otherwise stated, we assume that the clique C of a split partition (C, I) is maximum. This implies that none of the vertices in I is adjacent to every vertex of C . Split graphs form a subclass of chordal graphs.

Threshold graphs constitute a subclass of both trivially perfect graphs and split graphs. Threshold graphs have several characterizations [2, 7, 15], and we use the following one as a definition. A graph G is a *threshold graph* if and only

if it is a split graph and, for any split partition (C, I) of G , there is an ordering (v_1, v_2, \dots, v_k) of the vertices of C such that $N[v_1] \supseteq N[v_2] \supseteq \dots \supseteq N[v_k]$, and there is an ordering $(u_1, u_2, \dots, u_\ell)$ of the vertices of I such that $N(u_1) \subseteq N(u_2) \subseteq \dots \subseteq N(u_\ell)$ [15]. In that case, $(v_1, v_2, \dots, v_k, u_\ell, \dots, u_2, u_1)$ is a non-increasing degree ordering, and hence a uco, of G . Every connected threshold graph has a universal vertex, e.g., vertex v_1 in the ordering given above. Since we assume the clique of any split partition to be maximum, a vertex of C of smallest degree, e.g., vertex v_k in the ordering given above, has no neighbors in I . If a threshold graph is disconnected, then it has at most one nontrivial connected component; all other connected components are isolated vertices.

Split graphs, trivially perfect graphs, and threshold graphs are *hereditary* graph classes, meaning that the property of belonging to each of these classes is closed under taking induced subgraphs. These graph classes can be recognized in linear time; split partitions and uco-trees can also be obtained in linear time [2, 7, 8, 21].

3 Contractions and Induced Subgraph Isomorphisms of Trivially Perfect Graphs

In this section, we will give results on the computational complexity of CONTRACTIBILITY on trivially perfect graphs, corresponding to the first four rows of Table 1. The first theorem reveals the equivalence of the problems CONTRACTIBILITY and INDUCED SUBGRAPH ISOMORPHISM on the class of connected trivially perfect graphs.

Theorem 1. *For any two connected trivially perfect graphs G and H , the following three statements are equivalent:*

- (i) G can be contracted to H ;
- (ii) G contains an induced subgraph isomorphic to H ;
- (iii) T_G can be contracted to T_H .

Proof. First we prove the equivalence between (i) and (ii). Suppose G is H -contractible, and let uv be one of the edges of G that were contracted to obtain a graph isomorphic to H . Since G is trivially perfect, we have either $N_G[u] \subseteq N_G[v]$ or $N_G[v] \subseteq N_G[u]$. Without loss of generality, assume that $N_G[u] \subseteq N_G[v]$. Then contracting edge uv in G is equivalent to deleting vertex u from G . We can repeat this argument for every edge that was contracted, and conclude that G has an induced subgraph isomorphic to H .

For the opposite direction, suppose G' is an induced subgraph of G isomorphic to H . Let x be a universal vertex of G . We claim that G has an induced subgraph G'' isomorphic to H such that G'' contains x . If G' already contains x , then we can take $G'' = G'$. Suppose $x \notin V(G')$. Since G' is a connected trivially perfect graph, it has a universal vertex x' . Since x is a universal vertex in G , we have $N_G(x') \subseteq N_G[x]$. Hence the graph $G'' = G[(V(G') \setminus \{x'\}) \cup \{x\}]$ is isomorphic to G' , and is therefore also isomorphic to H . Now let $y \neq x$ be one of the

vertices that has to be deleted from G to obtain its induced subgraph G'' , i.e., $y \in V(G) \setminus V(G'')$. Since x is a universal vertex, we know that $N_G(y) \subseteq N_G[x]$. Then deleting vertex y from G is equivalent to contracting edge xy in G . Since $x \in V(G'')$, we can repeat this argument for every vertex of $V(G) \setminus V(G'')$, and conclude that G is H -contractible.

Next we prove the equivalence between (ii) and (iii). Suppose G contains an induced subgraph G' isomorphic to H , and let y be one of the vertices of G that has to be deleted to obtain G' . As argued above, we can assume that G' contains a universal vertex $x \neq y$ of G , which we can assume to be the root of T_G . This means in particular that $G - y$ is connected. Let x be the parent of y in T_G , and let T' be the tree obtained from T_G by contracting y onto x . This makes x the parent in T' of all children of y in T_G . Other than this, all parent-children relations are the same in T' as they were in T_G . Since x was already adjacent in G to all the vertices in the subtree of T_G rooted at y , we see that T' is indeed a uco-tree of $G - y$, and hence T' is isomorphic to T_{G-y} . Now we can repeat this argument for every vertex of $V(G) \setminus V(G')$, and conclude that T_G is T_H -contractible.

For the opposite direction, suppose T_G is T_H -contractible, and let xy be one of the edges of T_G that were contracted to obtain a tree isomorphic to T_H . Let $T' = T_G/xy$, and assume without loss of generality that x is the parent of y in T_G and that y is contracted onto x . Let G' be the trivially perfect graph having T' as a uco-tree. Note that a vertex u belongs to the subtree rooted at a vertex v in T' if and only if it already belonged to the subtree of T_G rooted at v . Therefore, for every pair of vertices $u, v \in V(G) \setminus \{y\}$, $uv \in E(G')$ if and only if $uv \in E(G)$, and hence G' is isomorphic to $G - y$. Now we can repeat this argument for every edge of T_G that was contracted, and conclude that G has an induced subgraph isomorphic to H . \square

Theorem 1 immediately gives us the result mentioned in the third row of Table 1, since checking whether a *fixed* graph H appears as an induced subgraph of an input graph G can be done trivially in polynomial time. Since Matoušek and Thomas [16] implicitly proved CONTRACTIBILITY to be NP-complete on rooted trees, Theorem 1 also implies the following result.

Corollary 1. *Both CONTRACTIBILITY and INDUCED SUBGRAPH ISOMORPHISM are NP-complete on connected trivially perfect graphs.*

Proof. Let T_1 and T_2 be two rooted trees given as input to CONTRACTIBILITY. Let G be the trivially perfect graph having T_1 as a uco-tree, and let H be the trivially perfect graph having T_2 as a uco-tree. By Theorem 1, G is H -contractible if and only if G has an induced subgraph isomorphic to H if and only if T_1 is contractible to T_2 . The corollary now follows from the result by Matoušek and Thomas [16], stating that CONTRACTIBILITY is NP-complete on rooted trees. \square

The results below show that both problems can be solved in polynomial time when G is a trivially perfect graph and H is a threshold graph, even if both G and

H are disconnected. Observe that these results are tight in light of Corollary 1. The following lemma is used in the proof of Theorem 2 below.

Lemma 1. *A connected trivially perfect graph G is a threshold graph if and only if every vertex in T_G has at most one child that is not a leaf.*

Proof. Recall that every threshold graph is trivially perfect. Let G be a threshold graph, and assume for a contradiction that there is a vertex x in T_G with two children u and v such that both u and v have children. This means that u and v are not adjacent in G , $N_G(u) \not\subseteq N_G(v)$, and $N_G(v) \not\subseteq N_G(u)$, which contradicts the assumption that G is a threshold graph. For the other direction, assume that G is trivially perfect and that every vertex in T_G has at most one child that is not a leaf. Let $P = p_1 p_2 \dots p_n$ be the unique path in T_G consisting of all the vertices that have at least one child, where p_1 is the root of T_G . Observe that $\{p_1, p_2, \dots, p_n\}$ is a clique in G . Since every vertex x of T_G is adjacent in G to all the vertices in the subtree of T_G rooted at x , the vertices of P satisfy $N_G[p_1] \supseteq N_G[p_2] \supseteq \dots \supseteq N_G[p_n]$. The leaves of T_G form an independent set in G . A leaf of T_G is adjacent in G to exactly those vertices that are ancestors of it in T_G . Since the leaves of T_G are only adjacent to vertices of P , there is also an ordering of them such that their neighborhoods are ordered by the subset relation. By the definition of threshold graphs, we can conclude that G is threshold. \square

Theorem 2. *Given a threshold graph G and an arbitrary graph H , it can be decided in linear time whether G can be contracted to H .*

Proof. First we check if H has at most as many vertices and edges as G , and output a negative answer if not. Since threshold graphs are hereditary, G is H -contractible only if H is a threshold graph. We can check in linear time whether this is the case, and output a negative answer if not. Suppose H is a threshold graph. Since edge contractions preserve connectivity, we can immediately output a negative answer if G and H do not have the same number of connected components. Suppose G and H have the same number of connected components. We trivially output “yes” if H contains no edges. Assume that both G and H contain at least one edge. Recall that any threshold graph contains at most one nontrivial connected component. Now the problem is equivalent to deciding whether the only nontrivial connected component of G can be contracted to the only nontrivial connected component of H . Hence for the rest of the proof we can assume G and H to be connected threshold graphs. By Theorem 1, our remaining task is equivalent to deciding whether the uco-tree T_G of G can be contracted to the uco-tree T_H of H .

We compute the uco-trees T_G and T_H in linear time. Since G is a threshold graph, we know by the proof of Lemma 1 that T_G has a unique path containing all the vertices that have at least one child. Let $S = s_1 s_2 \dots s_g$ be this path, where s_1 is the root of T_G . Let $T = t_1 t_2 \dots t_h$ be an analogous path in T_H . Let $l(v)$ be the number of leaves adjacent to a vertex v of T or S . We describe an algorithm that either finds a T_H -witness structure \mathcal{W} of T_G , or concludes that

T_G is not T_H -contractible. First we partition the vertices of S into witness sets according to the following simple procedure. Initially, we set $W(x) = \emptyset$ for each vertex x of T_H .

```

j = 1;
for i = 1 to h do
  ℓ = 0;
  repeat
    W(ti) = W(ti) ∪ {sj};
    ℓ = ℓ + l(sj);
    j = j + 1;
  until (ℓ ≥ l(ti) or j > g);
  if j > g and (ℓ < l(ti) or i < h) then
    stop;
end for;
if j ≤ g then
  W(th) = W(th) ∪ {sj, ..., sg};

```

If this procedure runs until the end without being terminated by the **stop** command, then for $i = 1, \dots, h$ we know that T_G has at least $l(t_i)$ leaves adjacent to vertices that we placed in $W(t_i)$. For each leaf t of T_H adjacent to t_i , we take a different leaf s of T_G adjacent to a vertex of $W(t_i)$, and we let $W(t) = \{s\}$. If T_G has any leaves that are adjacent to $W(t_i)$ but have not been assigned to witness sets of cardinality 1 like this, then we add all those leaves to $W(t_i)$. We repeat this for each i . Since the above procedure places each vertex of S in a witness set, this partitions all vertices of T_G into witness sets.

We first remark that the number of witness sets adjacent to each $W(t_i)$ is equal to the degree of t_i , and therefore \mathcal{W} is a T_H -witness structure of G . Hence, if the above procedure runs until the end without being terminated by the **stop** command, then it produces a T_H -witness structure of T_G , and hence T_G is T_H -contractible.

Now we prove that if the procedure is terminated by the **stop** command before reaching the end of the **for**-loop, then we can conclude that T_G is not T_H -contractible. Assume for a contradiction that T_G is T_H -contractible, but that the procedure is terminated by the **stop** command. Let $W(t_1), \dots, W(t_p)$ be the witness sets that the procedure generated before it terminated. Let \mathcal{W}' be a correct T_H -witness structure of T_G . Observe that $W'(t_1), \dots, W'(t_h)$ partitions S into exactly h subpaths, and hence each of these witness sets contains consecutive vertices of S . In particular, $W'(t_1)$ contains s_1 . Now let k be the smallest integer such that $W'(t_k)$ differs from $W(t_k)$; note that $k \leq p$, but not necessarily $k = p$. Since k is chosen to be smallest, the vertex of S with the smallest index in $W'(t_k)$ is the same as the vertex of S with smallest index in $W(t_k)$. Observe that the number of leaves of T_G adjacent to the vertices of $W'(t_k)$ is at least $l(t_k)$. The **repeat**-loop for building $W(t_k)$ stops as soon as this number is reached, and hence $W(t_k)$ does not contain more vertices of S than $W'(t_k)$. Since the two sets are different, we conclude that $W(t_k)$ contains fewer vertices of S than $W'(t_k)$,

meaning that $W(t_k) \subset W'(t_k)$. Consequently, k was not the step at which the above procedure stopped. Furthermore, the vertex of S with the smallest index in $W(t_{k+1})$ has a smaller index than the vertex of S with the smallest index in $W'(t_{k+1})$, and by the same arguments, the vertex of S with the largest index in $W(t_{k+1})$ has no larger index than the vertex of S with the largest index in $W'(t_{k+1})$. Now we can repeat the same arguments to conclude that the vertex of S with the largest index in $W(t_i)$ has no larger index than the vertex of S with the largest index than $W'(t_i)$, for $i = k + 2, \dots, p$, which contradicts the assumption that the procedure terminated after generating the set $W(t_p)$.

All the described steps can clearly be completed within a running time of $O(|V(G)| + |E(G)|)$ if G and H are given by their adjacency lists. If G and H are already recognized as threshold graphs before testing contractibility, and if they are provided to us in a compact representation, like a non-increasing degree order, then our running time becomes $O(|V(G)|)$. \square

Theorem 3. *Given a trivially perfect graph G and a threshold graph H , it can be decided in polynomial time whether G can be contracted to H .*

Proof. If G has less vertices or edges than H , then we return a negative answer. If G or H is disconnected, we first check whether G and H have the same number of connected components. If not, then we return a negative answer, since edge contractions preserve connectivity. Otherwise, for every connected component G' of G and the unique nontrivial connected component H' of H , we check if G' is H' -contractible. By Theorem 1, this is equivalent to testing whether $T_{G'}$ can be contracted to $T_{H'}$. The total running time is no worse than $O(|V(G)|)$ times the running time of checking contractibility on a pair of connected components. Hence for the rest of the proof we assume that input graphs G and H are connected.

We compute the uco-trees T_G and T_H in linear time. Let S be any path in T_G between the root and a leaf. We define $C(S)$ to be the graph obtained by contracting every edge of T_G , apart from the edges that have both endpoints in S or that are incident to a leaf. By Theorem 1, $C(S)$ is the uco-tree of an induced subgraph G_S of G , and by Lemma 1, G_S is a threshold graph. Note also that $C(S)$ has as many leaves as G . We claim that G is H -contractible if and only if there is a path S in T_G such that $C(S)$ is T_H -contractible. Clearly, if such a path S exists, then T_G is T_H -contractible and hence G is H -contractible.

We now prove that if G is H -contractible, then such a path S exists in T_G . Assume that G is H -contractible. Then we know by Theorem 1 and its proof that G has an induced subgraph G' isomorphic to H such that G' contains the root of T_G . Hence we can assume that T_G and $T_{G'}$ have the same root. Since G' is a threshold graph, by Lemma 1, $T_{G'}$ has the property that there is a unique maximal path from the root every vertex of which has at least one child in $T_{G'}$. Let $T = t_1 t_2 \cdots t_h$ be such a path in $T_{G'}$. Hence t_1 is the root of $T_{G'}$, and t_h is the lowest vertex that is not a leaf. By Theorem 1 and its proof, we know that T_G is contractible to $T_{G'}$ in such a way that the witness set of t_1 contains the root of T_G . Let \mathcal{W} be such a $T_{G'}$ -witness structure of T_G . This clearly implies the

existence of a path S in T_G from the root to a leaf such that $W(t_1), \dots, W(t_h)$ partition S into subpaths. For each $i \in \{1, \dots, h\}$, let T_i be the subgraph of T_G obtained by deleting the vertices belonging to $W(t_j)$ for all $j \neq i$. The connected component of T_i containing $W(t_i)$ contains at least as many leaves of T_G as the number of leaves which are neighbors of t_i in $T_{G'}$. Hence $C(S)$ is a graph that is $T_{G'}$ -contractible. Since G' is isomorphic to H , $T_{G'}$ is isomorphic to T_H , and consequently S is exactly the path whose existence in T_G we wanted to prove.

The algorithm is now clear from the above discussion. For each distinct maximal path S of T_G from the root containing only vertices that have at least one child, we check whether $C(S)$ is contractible to T_H using the linear-time procedure described in the proof of Theorem 2. Since the number of distinct paths S is $O(|V(G)|)$, the total running time is polynomial. \square

By Theorem 1, INDUCED SUBGRAPH ISOMORPHISM is equivalent to CONTRACTIBILITY on connected trivially perfect graphs. Hence the only difference between the proofs of the following results and those of the two previous theorems is in the connectivity arguments.

Theorem 4. *Given a trivially perfect graph G and a threshold graph H , it can be decided in polynomial time whether G contains an induced subgraph isomorphic to H .*

Proof. Let G be a trivially perfect graph and let H be a threshold graph. We construct a graph G' from G by adding a new vertex x and making it adjacent to all vertices of G . Note that G' is a connected trivially perfect graph. Let H' be the connected threshold graph obtained from H by adding a new vertex y and making it adjacent to all vertices of H . We claim that G has an induced subgraph isomorphic to H if and only if G' has an induced subgraph isomorphic to H' . Assume that G' has an induced subgraph G'' isomorphic to H' . By the arguments used in the proof of Theorem 1, we can assume that G'' contains x . Consequently, $G'' - x$ is an induced subgraph of G . Since $G'' - x$ is isomorphic to H , this direction of the claim follows. For the other direction, assume that there exists a subset $U \subseteq V(G)$ such that $G[U]$ is isomorphic to H . Then the subgraph of G' induced by $U \cup \{x\}$ is isomorphic to H' . Hence, in order to prove Theorem 4, it suffices to show that we can decide in polynomial time whether a connected trivially perfect graph G' can be contracted to a connected threshold graph H' . This follows from Theorems 1 and 3. \square

Theorem 5. *Given a threshold graph G and an arbitrary graph H , it can be decided in linear time whether G has an induced subgraph isomorphic to H .*

Proof. Since threshold graphs are trivially perfect, we can use the same arguments as in the proof of Theorem 4 to conclude that it is enough to consider connected input graphs. Now the result follows from Theorems 1 and 2. \square

4 Contracting Split Graphs

In the previous section, we showed that deciding whether a threshold graph G can be contracted to an arbitrary graph H can be done in linear time. The next

theorem shows that this result is not likely to be extendable to split graphs. A *hypergraph* F is a pair (Q, \mathcal{S}) consisting of a set $Q = \{q_1, \dots, q_k\}$, called the *vertices* of F , and a set $\mathcal{S} = \{S_1, \dots, S_\ell\}$ of nonempty subsets of Q , called the *hyperedges* of F . A *2-coloring* of a hypergraph $F = (Q, \mathcal{S})$ is a partition (Q_1, Q_2) of Q such that $Q_1 \cap S_j \neq \emptyset$ and $Q_2 \cap S_j \neq \emptyset$ for $j = 1, \dots, \ell$.

Theorem 6. CONTRACTIBILITY is NP-complete on input pairs (G, H) where G is a connected split graph and H is a connected threshold graph.

Proof. We use a reduction from HYPERGRAPH 2-COLORABILITY, which is the problem of deciding whether a given hypergraph has a 2-coloring. This problem, also known as SET SPLITTING, is NP-complete [14]. The problem remains NP-complete when restricted to hypergraphs in which every vertex is contained in at least two hyperedges.

Let $F = (Q, \mathcal{S})$ be a hypergraph with $Q = \{q_1, \dots, q_k\}$ and $\mathcal{S} = \{S_1, \dots, S_\ell\}$ such that every vertex of Q appears in at least two hyperedges. We construct a split graph G as follows. We start with a clique $A = \{a_1, \dots, a_k\}$, where the vertex $a_i \in A$ corresponds to the vertex $q_i \in Q$ for $i = 1, \dots, k$. We add an independent set $B = \{b_1, \dots, b_\ell\}$, where the vertex $b_i \in B$ corresponds to the hyperedge $S_i \in \mathcal{S}$ for $i = 1, \dots, \ell$. Finally, for $i = 1, \dots, k$ and $j = 1, \dots, \ell$, we add an edge between a_i and b_j in G if and only if $q_i \in S_j$. We also construct a threshold graph H from a single edge x_1x_2 by adding an independent set $Y = \{y_1, \dots, y_\ell\}$ on ℓ vertices, and making each vertex of Y adjacent to both x_1 and x_2 . We claim that G can be contracted to H if and only if F has a 2-coloring.

Suppose F has a 2-coloring, and let (Q_1, Q_2) be a 2-coloring of F . Let (A_1, A_2) be the partition of A corresponding to this 2-coloring of F . Note that A_1 and A_2 both form a connected set in G , since the vertices of A form a clique in G . We contract A_1 into a single vertex p_1 , and we contract A_2 into a single vertex p_2 . Let G' denote the resulting graph. Since (Q_1, Q_2) is a 2-coloring of F , every vertex in B is adjacent to at least one vertex of A_1 and at least one vertex of A_2 in the graph G . As a result, every vertex in B is adjacent to both p_1 and p_2 in G' . Hence G' is isomorphic to H , which means that G can be contracted to H .

Now suppose G can be contracted to H , and let \mathcal{W} be an H -witness structure of G . Since we assumed that every vertex of F appears in at least two hyperedges, every vertex in A has at least two neighbors in B . This means that B is the only independent set of size ℓ in G . Since Y is an independent set of size ℓ in H , the witness sets $W(y_1), \dots, W(y_\ell)$ each must contain exactly one vertex of B . In fact, since every vertex of A has at least two neighbors in B , we have $W(Y) = B$. This means that the two witness sets $W(x_1)$ and $W(x_2)$ form a partition of the vertices of A . By the definition of an H -witness structure and the construction of H , each witness set $W(y_i)$ is adjacent to both $W(x_1)$ and $W(x_2)$. Hence the partition $(W(x_1), W(x_2))$ of A corresponds to a 2-coloring of F . \square

Although the problem of deciding whether a split graph G can be contracted to a split graph H is NP-complete when both G and H are given as input, we will show in the remainder of this section that the problem can be solved in polynomial time when H is fixed.

Definition 1. Let G and H be two split graphs with split partitions (C_G, I_G) and (C_H, I_H) , respectively. A set $U \subseteq I_G$ with $|U| = |I_H|$ is called H -compatible if G has an H -witness structure \mathcal{W} such that $W(I_H) = U$.

Lemma 2. Let G and H be two split graphs. Then G is H -contractible if and only if G contains an H -compatible set.

Proof. If G has an H -compatible set, then G is H -contractible by Definition 1. For the reverse direction, assume that G is H -contractible, and let \mathcal{W} be an H -witness structure of G . If I_H is empty, then $U = \emptyset$ is an H -compatible set of G by Definition 1, since the H -witness structure \mathcal{W} satisfies $W(I_H) = U = \emptyset$. Suppose I_H is not empty. Since I_H is an independent set in H , there can be at most one vertex $v \in I_H$ such that $W(v)$ contains a vertex of C_G . Note that this implies that G is not H -contractible if $|I_G| \leq |I_H| - 1$. Suppose there is a witness set $W(v)$ such that $W(v) \cap C_G \neq \emptyset$. Then for each $v' \in I_H \setminus \{v\}$, the witness set $W(v')$ contains only vertices of I_G , i.e., $W(I_H \setminus \{v\}) \subseteq I_G$. Since I_G is an independent set in G and every witness set is connected, $|W(v')| = 1$ for every $v' \in I_H \setminus \{v\}$. Recall that C_H is assumed to be a maximum clique of H . Hence there is a vertex x of C_H that is not adjacent to v in H , and therefore witness set $W(x)$ is not adjacent to $W(v)$ in G . Since $W(v)$ contains at least one vertex of C_G and is not adjacent to $W(x)$, $W(x)$ only contains vertices of I_G . Since I_G is an independent set and $W(x)$ is connected, we must have $W(x) = \{a\}$ for some vertex $a \in I_G$. This implies that $W(v)$ is adjacent to witness set $W(x')$ for every $x' \in C_H \setminus \{x\}$. Moreover, since for every $v' \in I_H \setminus \{v\}$ the witness set $W(v')$ consists of a single vertex from I_G , $W(x)$ is not adjacent to $W(v')$ for any $v' \in I_H \setminus \{v\}$. Therefore, we can define another H -witness structure \mathcal{W}' of G by setting $W'(v) = W(x)$, $W'(x) = W(v)$, and $W'(y) = W(y)$ for every $y \in V(H) \setminus \{v, x\}$. Now \mathcal{W}' has the property that $|W'(y)| = 1$ for every vertex $y \in I_H$. Consequently, $U = W'(I_H)$ is an H -compatible set of G by Definition 1. \square

If U is an H -compatible set of G , then, by Definition 1, G has an H -witness structure \mathcal{W} such that $W(I_H) = U$. The next technical lemma shows that each of the witness sets of \mathcal{W} contains a small subset, bounded in size by a function of $|V(H)|$ only, such that the collection of these subsets provide all the necessary adjacencies between the witness sets of \mathcal{W} .

Lemma 3. Let G and H be two connected split graphs with split partitions (C_G, I_G) and (C_H, I_H) , respectively. Let $C_H = \{x_1, \dots, x_k\}$. A set $U \subseteq I_G$ with $|U| = |I_H|$ is H -compatible if and only if there exists a collection \mathcal{M} of pairwise disjoint subsets $M(x_1), \dots, M(x_k)$ of $V(G) \setminus U$ satisfying the following properties:

- (i) at most one set of \mathcal{M} contains a vertex of I_G , and such a set has cardinality 1 if it exists;
- (ii) for every subset $X \subseteq U$, $M(x_i)$ contains at most two vertices a and b such that $N_G(a) \cap U = N_G(b) \cap U = X$, for $i = 1, \dots, k$;

- (iii) $\bigcup_{i=1}^k |M(x_i)| \leq |C_H| \cdot 2^{|I_H|+1}$;
- (iv) for every $v \in V(G) \setminus (U \cup \bigcup_{i=1}^k M(x_i))$, there is a set in \mathcal{M} that is adjacent to every vertex in $N_G(v) \cap U$;
- (v) the graph $G' = G[U \cup \bigcup_{i=1}^k M(x_i)]$ has an H -witness structure \mathcal{W}' such that $W'(I_H) = U$ and $W'(x_i) = M(x_i)$ for $i = 1, \dots, k$.

Proof. Let U be a subset of I_G of cardinality $|I_H|$. Suppose there exists a collection $\mathcal{M} = \{M(x_1), \dots, M(x_k)\}$ that has properties (i)–(v). Let $M = \bigcup_{i=1}^k M(x_i)$, and let $G' = G[U \cup M]$. By property (v), there exists an H -witness structure \mathcal{W}' of G' such that $W'(I_H) = U$ and $W'(x_i) = M(x_i)$ for $i = 1, \dots, k$. We will show that \mathcal{W}' can be extended to an H -witness structure \mathcal{W} of G with $W(I_H) = U$. Let $v \in C_G \setminus M$. By property (iv), there is a set $W'(x_i) \in \mathcal{W}'$ that is adjacent to every vertex of $N_G(v) \cap U$. Add v to $W'(x_i)$. Note that adding v to $W'(x_i)$ does not change the adjacencies between $W'(x_i)$ and the other witness sets of \mathcal{W}' . Repeat this until all vertices of $C_G \setminus M$ have been added to sets of \mathcal{W}' . Let $w \in I_G$. Since every vertex of C_G now belongs to a set of \mathcal{W}' , there exists a set $W'(x_j) \in \mathcal{W}'$ that is adjacent to w . Add w to $W'(x_j)$. It is clear that adding w to $W'(x_j)$ does not change the adjacencies between $W'(x_j)$ and the other witness sets of \mathcal{W}' . Repeat this until all vertices of $I_G \setminus U$ have been added to sets of \mathcal{W}' . We end up with an H -witness structure \mathcal{W} of G with $W(I_H) = U$, which means that U is H -compatible by Definition 1.

For the reverse direction, suppose that U is an H -compatible set of G . Then, by definition, G has an H -witness structure \mathcal{W} such that $W(I_H) = U$. Suppose there exist a vertex $x_i \in C_H$ whose witness set $W(x_i)$ contains only vertices of I_G . Note that this means that x_i is not adjacent to any vertex in I_H , as no vertex of I_G has a neighbor in U . Since every witness set is a connected set, $W(x_i) = \{p\}$ for some vertex $p \in I_G$. Suppose there is another set $W(x_j)$ that contains only vertices of I_G . Then $W(x_j) = \{q\}$ for some $q \in I_G \setminus \{p\}$. Since x_i and x_j are adjacent in H , the witness sets $W(x_i)$ and $W(x_j)$ must be adjacent in G . This contradicts the fact that p and q , both belonging to the independent set I_G , are not adjacent. This implies that there is at most one vertex $x_i \in C_H$ such that $W(x_i) = \{p\}$ for some $p \in I_G$. Moreover, if such a witness set exists, then p has at least one neighbor in the witness set $W(x_j)$ for every $x_j \in C_H$, since \mathcal{W} is an H -witness structure and C_H is a clique in H .

We now show how to construct the collection \mathcal{M} from \mathcal{W} . For $i = 1, \dots, k$, the set $M(x_i)$ is a subset of the witness set $W(x_i)$, and $M(x_i)$ can be obtained from $W(x_i)$ as follows. We first partition the vertices of $W(x_i)$ into sets in such a way, that two vertices a and b of $W(x_i)$ belong to the same partition set if and only if they are adjacent to the same vertices in U , i.e., if $N_G(a) \cap U = N_G(b) \cap U$. Let $S_i \subseteq W(x_i)$ be the partition set whose vertices have no neighbor in U . From each non-empty partition set other than S_i , we arbitrarily choose one vertex and add it to $M(x_i)$. If $S_i \neq W(x_i)$, then no vertex of S_i is added to $M(x_i)$. If $S_i = W(x_i)$ and S_i contains at least one vertex of C_G , then we arbitrarily choose one of the vertices of $S_i \cap C_G$ and add it to $M(x_i)$. If $S_i = W(x_i)$ and S_i contains no vertices of C_G but contains a vertex of I_G , then we add that vertex to $M(x_i)$; recall that in this case S_i contains exactly one vertex, and that this

case occurs at most once. After we have generated all the sets $M(x_i)$ this way, we check if there is a set $M(x_j) = \{p\}$ for some $p \in I_G$. If so, then we check, for every $x_i \in C_H \setminus \{x_j\}$, whether the set $M(x_i)$ contains at least one neighbor of p . If not, then we arbitrarily choose a neighbor p' of p in $W(x_i)$ and add it to $M(x_i)$. As we argued before, such a neighbor p' always exists. Note that adding p' to $M(x_i)$ does not change the adjacencies between $M(x_i)$ and U , since $M(x_i)$ already contained one vertex from every non-empty partition set of $W(x_i)$.

Let \mathcal{M} be the collection of sets $M(x_i)$ that are obtained this way from the witness sets $W(x_i)$, for every $x_i \in C_H$. For every $x_i \in C_H$, every vertex of $W(x_i) \setminus S_i$ belongs to C_G , since no vertex of I_G has a neighbor in U . The only time a vertex of I_G is added to a set $M(x_i)$ is when $S_i = W(x_i)$ and $W(x_i)$ does not contain a vertex of C_G . As we argued above, this situation occurs at most once, so \mathcal{M} satisfies property (i). For every witness set $W(x_i)$, there are at most $2^{|I_H|}$ non-empty partition sets, since U is H -compatible and thus has cardinality $|I_H|$. The set $M(x_i)$ contains one vertex from each non-empty partition set, and possibly one extra vertex a which is adjacent to the only set in \mathcal{M} of the form $M(x_j) = \{p\}$ for some $p \in I_G$. If such a vertex a exists, then this is the only vertex of $M(x_i)$ for which there exists another vertex $b \in M(x_i)$ with $N_G(a) \cap U = N_G(b) \cap U$. Hence every set $M(x_i)$ contains at most two vertices a and b such that $N_G(a) \cap U = N_G(b) \cap U$, for $i = 1, \dots, k$, and therefore certainly satisfies property (ii). The reason we write “for every subset $X \subseteq U$ ” instead of “for at most one subset $X \subseteq U$ ” in property (ii) will become clear from the description of the algorithm in Case 2 in the proof of Lemma 4. The number of non-empty partition sets is at most $2^{|I_H|}$, so $|M(x_i)| \leq 2^{|I_H|} + 1 < 2^{|I_H|+1}$. This, together with the fact that $k = |C_H|$, implies property (iii). Again, the reason for not formulating property (iii) in the strongest possible way will become clear in the proof of Lemma 4. Let $v \in V(G) \setminus (U \cup \bigcup_{i=1}^k M(x_i))$. Since $v \notin U$, v belongs to a witness set $W(x_i)$ for some $x_i \in C_H$. Consider the set $M(x_i)$. By construction, there exists a vertex $w \in M(x_i)$ such that $N_G(v) \cap U = N_G(w) \cap U$, as otherwise v would have been added to $M(x_i)$. Hence $M(x_i)$ is adjacent to all vertices in $N_G(v) \cap U$, and property (iv) holds.

It remains to show \mathcal{M} satisfies property (v). For every $x_i \in C_H$, the set $M(x_i)$ is adjacent to exactly the same vertices in U as the set $W(x_i)$, since $M(x_i)$ contains a vertex from every partition class of $W(x_i)$. If every set in \mathcal{M} contains at least one vertex of C_G , then the fact that C_G is a clique in G implies that the sets of \mathcal{M} are pairwise adjacent. Hence property (v) holds in this case. Suppose \mathcal{M} contains a set of the form $M(x_j) = \{p\}$ for some $p \in I_G$. Since \mathcal{M} satisfies property (i), every set in $\mathcal{M} \setminus M(x_j)$ contains only vertices from C_G , which means that those sets are pairwise adjacent. The last step in the construction of \mathcal{M} ensures that p is adjacent to every set in $\mathcal{M} \setminus M(x_j)$. Hence property (v) also holds in this case. \square

We call the collection \mathcal{M} in Lemma 3 an *essential collection* for U , and the sets $M(x_i)$ are called *essential sets*. The fact that the total size of an essential collection does not depend on the size of G plays a crucial role in the proof of the following lemma.

Lemma 4. *Let G and H be two split graphs with split partitions (C_G, I_G) and (C_H, I_H) , respectively. Given a set $U \subseteq I_G$ with $|U| = |I_H|$, it can be decided in $f(|V(H)|) \cdot |V(G)|^3$ time whether U is H -compatible, where the function f depends only on H and not on G .*

Proof. Let U be a subset of I_G with $|U| = |I_H|$, and let $C_H = \{x_1, \dots, x_k\}$. Throughout the proof, we use k to represent the number of vertices in C_H . We present an algorithm that checks whether or not there exists an essential collection for U . By Lemma 3, U is H -compatible if and only if such a collection exists. We distinguish two cases, depending on whether or not every vertex of C_H has at least one neighbor in I_H .

Case 1. Every vertex of C_H has at least one neighbor in I_H .

For every subset $X \subseteq U$, we define the set $Z_X = \{v \in V(G) \setminus U \mid N_G(v) \cap U = X\}$. Note that there are at most $2^{|U|}$ non-empty sets Z_X , and that these sets form a partition of $V(G) \setminus U$. Let $\mathcal{Z} = \{Z_X \mid X \subseteq U\}$ be the collection of these sets Z_X . Let \mathcal{A} be the power set of \mathcal{Z} , i.e., \mathcal{A} is the set consisting of all possible subsets of \mathcal{Z} . For every element $A \in \mathcal{A}$, we have $A = \{Z_{X_1}, \dots, Z_{X_\ell}\}$ for some $1 \leq \ell \leq 2^{|U|}$, where $X_i \subseteq U$ for $i = 1, \dots, \ell$ and $X_i \neq X_j$ whenever $i \neq j$. Finally, let \mathcal{B} be the set of all ordered k -tuples of elements in \mathcal{A} , where elements of \mathcal{A} may appear more than once in an element $B \in \mathcal{B}$. For any element $B \in \mathcal{B}$, we have $B = (A_1, A_2, \dots, A_k)$, where $A_i \in \mathcal{A}$ for $i = 1, \dots, k$.

For every $B = (A_1, A_2, \dots, A_k) \in \mathcal{B}$, we generate a “candidate” essential set $M(x_i)$ for every vertex $x_i \in C_H$ as follows. At the start, all the vertices of C_G are unmarked, and all the vertices of $I_G \setminus U$ are marked. Of every set in A_1 that contains at least one unmarked vertex, we add one unmarked vertex to $M(x_1)$. We mark all the vertices that are added to $M(x_1)$. We then generate a candidate essential set $M(x_2)$ as before, adding an unmarked vertex from every set in A_2 that contains such a vertex to $M(x_2)$, and marking all the vertices added to $M(x_2)$. After we have generated a candidate essential set $M(x_i)$ for every vertex $x_i \in C_H$ in the way described, we define $M = \bigcup_{i=1}^k M(x_i)$, i.e., M is the set of marked vertices of C_G . Let \mathcal{M} denote the collection of all candidate essential sets $M(x_i)$. Note that the sets of \mathcal{M} are pairwise disjoint subsets of C_G . It is clear that, by construction, \mathcal{M} satisfies properties (i), (ii), and (iii) of Lemma 3.

We now check whether \mathcal{M} satisfies properties (iv) and (v). In order to check property (iv), we determine for every vertex $v \in V(G) \setminus (U \cup M)$ whether \mathcal{M} contains a candidate essential set that is adjacent to every vertex in $N_G(v) \cap U$. \mathcal{M} satisfies property (iv) if and only if such a set exists for every vertex of $V(G) \setminus (U \cup M)$. In order to check property (v), we first delete all the vertices in $V(G) \setminus (U \cup M)$, and then contract each of the candidate essential sets $M(x_i)$ into a single vertex. \mathcal{M} satisfies property (v) if and only if the obtained graph is isomorphic to H . If \mathcal{M} satisfies properties (iv) and (v), then \mathcal{M} is an essential collection for U , and the algorithm concludes that U is H -compatible. If \mathcal{M} does not satisfy properties (iv) and (v), then we unmark all vertices of C_G (the vertices of $I_G \setminus U$ remain marked) and repeat the procedure on the next element

of \mathcal{B} . If we have processed all elements of \mathcal{B} without finding an essential collection for U , then we conclude that U is not H -compatible due to Lemma 3.

Before we consider Case 2 below, we first prove why the algorithm for Case 1 is correct. If the algorithm finds a collection \mathcal{M} that satisfies properties (i)–(v), then \mathcal{M} is an essential collection for U by definition. Hence, by Lemma 3, the algorithm correctly concludes that U is H -compatible in this case. It remains to prove that if U is H -compatible, then our algorithm will find an essential set \mathcal{M} for U .

Suppose U is H -compatible. Then, by definition, G has an H -witness structure \mathcal{W}' such that $W'(I_H) = U$. Let \mathcal{M}' be an essential collection for U , obtained from \mathcal{W}' in the way described in the proof of Lemma 3. Since every vertex of C_H has at least one neighbor in I_H , every set $M'(x_i) \in \mathcal{M}'$ satisfies the following two properties by construction: $M'(x_i)$ contains no vertex of I_G , and $M'(x_i)$ does not contain two vertices a and b such that $N_G(a) \cap U = N_G(b) \cap U$, i.e., $M'(x_i)$ contains at most one vertex from every set Z_X , for every subset $X \subseteq U$. Note that the sets of \mathcal{M}' are pairwise disjoint, which means that, for every set Z_X , the number of vertices in Z_X is at least as big as the number of sets of \mathcal{M}' that contain a vertex of Z_X . Consider the set $M'(x_1)$. Let $A_1 = \{Z_{X_1}, \dots, Z_{X_\ell}\}$ be the collection of sets in \mathcal{Z} such that $M'(x_1) \cap Z_{X_i} \neq \emptyset$ for $i = 1, \dots, \ell$. Let A_2, \dots, A_k be defined similarly for the sets $M'(x_2), \dots, M'(x_k)$, respectively. Let $B = (A_1, \dots, A_k)$. Since our algorithm processes every element in \mathcal{B} if necessary, it will consider B at some stage, unless it found an essential collection for U before and correctly concluded that U is H -compatible. When processing B , the algorithm will create candidate essential sets $M(x_1), \dots, M(x_k)$ such that, for $i = 1, \dots, k$, the set $M(x_i)$ contains a vertex from exactly those sets in \mathcal{Z} that $M'(x_i)$ contains a vertex from. Just like the sets $M'(x_i)$, the sets $M(x_i)$ are pairwise disjoint subsets of C_G . Since $N_G(a) \cap U = N_G(b) \cap U$ for every two vertices a, b that belong to the same set of \mathcal{Z} , the collection $\mathcal{M} = \{M(x_1), \dots, M(x_k)\}$ satisfies properties (i)–(v) of Lemma 3, and thus is an essential collection for U .

Case 2. At least one vertex of C_H has no neighbor in I_H .

Let S be the set of vertices of C_H that do not have any neighbors in I_H . Assume, without loss of generality, that $x_k \in S$. We first run the algorithm for Case 1 on U . If we find an essential set for U this way, then we conclude that U is H -compatible. Suppose we do not find an essential set for U using the algorithm for Case 1. We then run the following algorithm for every vertex $p \in I_G \setminus U$.

We first define a candidate essential set for x_k by setting $M(x_k) = \{p\}$. For every subset $X \subseteq U \cup \{p\}$, we define the set $Z_X = \{v \in V(G) \setminus (U \cup \{p\}) \mid N_G(v) \cap (U \cup \{p\}) = X\}$. Let $\mathcal{Z} = \{Z_X \mid X \subseteq (U \cup \{p\})\}$ be the collection of these sets Z_X , and let \mathcal{A} be the power set of \mathcal{Z} . Let \mathcal{B} be the set of all ordered $(k-1)$ -tuples of elements in \mathcal{A} , where elements of \mathcal{A} may appear more than once in an element $B \in \mathcal{B}$. Then, for every $B = (A_1, \dots, A_{k-1}) \in \mathcal{B}$, we act as follows. We mark all the vertices of vertices of $I_G \setminus U$, and leave the vertices of C_G unmarked. In particular, p is marked. For every i from 1 to $k-1$, we generate a candidate essential set $M(x_i)$ as we did in Case 1: from every set in A_i that contains at

least one unmarked vertex, we add one unmarked vertex to $M(x_i)$. We mark all the vertices that are added to $M(x_i)$.

Let \mathcal{M} be the collection of candidate essential sets generated this way, and let $M = \bigcup_{i=1}^k M(x_i)$ be the set of all marked vertices, including p . Since we marked all the vertices of $I_G \setminus U$ at the start of the algorithm, only the candidate essential set $M(x_k) = \{p\}$ contains a vertex from I_G . Hence \mathcal{M} satisfies property (i) of Lemma 3. Every set $M(x_i)$ contains at most one vertex from every set in \mathcal{Z} , and therefore never contains two vertices with exactly the same neighbors in $U \cup \{p\}$. It is however possible, for every $X \subseteq U$, that $M(x_i)$ contains two vertices a and b such that $N_G(a) \cap U = N_G(b) \cap U = X$, in which case exactly one of these two vertices is adjacent to p . This implies that \mathcal{M} satisfies property (ii). Property (iii) follows from the fact that $|U \cup \{p\}| = |I_H| + 1$, so \mathcal{Z} contains at most $2^{|I_H|+1}$ non-empty sets Z_X , each of which contributes at most one vertex to every set $M(x_i)$. Checking whether \mathcal{M} also satisfies properties (iv) and (v) is done in exactly the same way as in Case 1. If \mathcal{M} satisfies properties (iv) and (v), then \mathcal{M} is an essential collection for U , and the algorithm concludes that U is H -compatible. If \mathcal{M} does not satisfy both properties (iv) and (v), then we unmark all the vertices in C_G and repeat the procedure on the next element of \mathcal{B} . If none of the elements of \mathcal{B} yields an essential set for U , then the algorithm is repeated with another vertex $p' \in I_G \setminus U$ playing the role of p . If, for all the vertices of $I_G \setminus U$, none of the elements of \mathcal{B} yields an essential collection for U , then the algorithm concludes that U is not H -compatible.

Let us argue why the algorithm for Case 2 is correct. If the algorithm finds a collection \mathcal{M} that satisfies properties (i)–(v) of Lemma 3, then clearly \mathcal{M} is an essential collection for U . Hence, by Lemma 3, our algorithm correctly concludes that U is H -compatible in this case. We now show that if U is H -compatible, then our algorithm will find an essential collection for U . Suppose U is H -compatible. By Lemma 3, there exists an essential collection \mathcal{M}' for U . If there exists an essential collection \mathcal{M}'' for U such that every $M''(x_i) \in \mathcal{M}''$ contains at least one vertex of C_G , then the algorithm for Case 1 will find an essential collection for U by the arguments used in the correctness proof of Case 1. Suppose such a collection does not exist. Then, by property (i) of Lemma 3, we know that \mathcal{M}' contains exactly one set $M'(x_j)$ such that $M'(x_j) = \{p\}$ for some $p \in I_G \setminus U$. All other sets of \mathcal{M}' contain only vertices of C_G . Since p is not adjacent to any vertex of U , x_j must be a vertex of C_H that does not have any neighbors in I_H , i.e., $x_j \in S$. Recall that $x_k \in S$, and note that all the vertices of S have the same closed neighborhood in H . Hence we can assume that $x_j = x_k$, since we can swap the indices of x_j and x_k otherwise. Recall that the collection \mathcal{Z} consists of all subsets of $U \cup \{p\}$ in Case 2. For $i = 1, \dots, k-1$, the set $M'(x_i)$ contains at most one vertex from every set of \mathcal{Z} . For every vertex $x_i \in C_H \setminus \{x_k\}$, let A_i be the collection of sets in \mathcal{Z} such that, for every $Z \in \mathcal{Z}$, $Z \in A_i$ if and only if $M'(x_i) \cap Z \neq \emptyset$. Let $B = (A_1, \dots, A_{k-1})$. If our algorithm processes B , then it will find a collection of essential sets $M(x_1), \dots, M(x_{k-1})$ such that, for $i = 1, \dots, k-1$, the set $M(x_i)$ contains a vertex from exactly those sets in \mathcal{Z} that $M'(x_i)$ contains a vertex from. Since $N_G(a) \cap (U \cup \{p\}) =$

$N_G(b) \cap (U \cup \{p\})$ for every two vertices a and b in the same set $Z_X \in \mathcal{Z}$, $\mathcal{M} = \{M(x_1), \dots, M(x_{k-1}), \{p\}\}$ satisfies properties (i)–(v) of Lemma 3. Hence \mathcal{M} is an essential collection for U , and U is H -compatible.

It remains to determine the running time of our algorithm. In Case 1, the set \mathcal{Z} contains all possible subsets of U , so $|\mathcal{Z}| = 2^{|U|} = 2^{|I_H|}$. Since \mathcal{A} consists of all possible subsets of \mathcal{Z} , we have $|\mathcal{A}| = 2^{\mathcal{Z}} = 2^{2^{|I_H|}}$. Since \mathcal{B} consists of all possible ordered $|C_H|$ -tuples of elements of \mathcal{A} , we have $|\mathcal{B}| = |\mathcal{A}|^{|C_H|} = 2^{|C_H| \cdot 2^{|I_H|+1}}$. In the worst case, our algorithm tries all possible elements of \mathcal{B} . Each of those elements yields a collection \mathcal{M} of candidate essential sets. Testing whether \mathcal{M} satisfies property (iv) can be done in $O(|V(G)|^2)$ time. To test whether \mathcal{M} satisfies property (v), we first delete some vertices and contract each of the candidate essential sets into a single vertex, and then test whether the obtained graph is isomorphic to H . This can be done $O(|V(G)|^2)$ time and $2^{O(\sqrt{|V(H)| \log |V(H)|})}$ time [1], respectively. Hence the algorithm for Case 1 runs in $f(|V(H)|) \cdot |V(G)|^2$ time. In Case 2, the set \mathcal{Z} contains all possible subsets of $U \cup \{p\}$, so $|\mathcal{Z}| = 2^{|I_H|+1}$. Hence $|\mathcal{A}| = 2^{\mathcal{Z}} = 2^{2^{|I_H|+1}}$. Since \mathcal{B} consists of all possible ordered $(|C_H| - 1)$ -tuples of elements of \mathcal{A} , we have $|\mathcal{B}| = |\mathcal{A}|^{|C_H|-1} = 2^{(|C_H|-1) \cdot 2^{|I_H|+1}}$. The only other difference with Case 1 is that we might have to run the algorithm for Case 2 for every vertex $p \in I_G \setminus U$, which adds an $O(|I_G|)$ factor to the running time. Hence the total running time needed to test if the set U is H -compatible is $f(|V(H)|) \cdot |V(G)|^3$. \square

Theorem 7. *For every fixed graph H , the problem of deciding whether a given split graph G can be contracted to H can be solved in polynomial time.*

Proof. Let G be a split graph with split partition (C_G, I_G) , and suppose G is an input graph of H -CONTRACTIBILITY. Observe that contracting any edge of a split graph yields another split graph. Hence G can be contracted to a graph H only if H is a split graph with $|V(G)| \geq |V(H)|$. We can check this in time linear in the size of H , which is constant. Suppose H is a split graph, and let (C_H, I_H) be a split partition of H . By Lemma 2, G can be contracted to H if and only if G contains an H -compatible set. The number of different subsets of I_G of cardinality $|I_H|$ is $\binom{|I_G|}{|I_H|} \leq |V(G)|^{|I_H|}$. For each of those sets, we can test in $f(|V(H)|) \cdot |V(G)|^3$ time whether it is H -compatible by Lemma 4. Since the graph H is fixed, it can be decided in time polynomial in $|V(G)|$ whether G can be contracted to H . \square

5 Concluding Remarks

It is known that INDUCED SUBGRAPH ISOMORPHISM is NP-complete on cographs and on interval graphs [4, 5]. Hence Corollary 1 strengthens these existing NP-completeness results. The INDUCED SUBGRAPH ISOMORPHISM problem is also known to be NP-complete on another subclass of interval graphs, called proper interval graphs [4, 5], but only if the input graph H is disconnected [9]. Thus

we find it interesting that this problem is NP-complete on *connected* trivially perfect graphs.

The positive results on H -CONTRACTIBILITY given in this paper and in [11] give rise to other interesting questions as well. Is H -CONTRACTIBILITY solvable in polynomial time when G is a chordal graph? Is CONTRACTIBILITY fixed parameter tractable, parameterized by the size of H , when G is a split graph or a planar graph, i.e., is there an algorithm with running time $f(|V(H)|) \cdot |V(G)|^{O(1)}$, where the function f only depends on H and not on G ?

References

1. Babai, L., Luks, E.: Canonical labelling of graphs. In: Proceedings of STOC 1983, pp. 171–183, ACM, New York (1983)
2. Brandstädt, A., Le, V. B., Spinrad, J.: Graph Classes: A Survey. SIAM Monographs on Discrete Mathematics and Applications (1999)
3. Brouwer, A. E., Veldman, H. J.: Contractibility and NP-completeness. Journal of Graph Theory 11:71–79 (1987)
4. Damaschke, P.: Induced subgraph isomorphism for cographs is NP-complete. In: Proceedings of WG 1990, LNCS, vol 484, pp. 72–78, Springer, Heidelberg (1991)
5. Garey, M. R., Johnson, D. S.: Computers and Intractability. W.H. Freeman and Co., New York (1979)
6. George, A., Liu, J. W.: Computer Solution of Large Sparse Positive Definite. Prentice Hall Professional Technical Reference (1981)
7. Golumbic M.C.: Algorithmic Graph Theory and Perfect Graphs. Annals of Discrete Mathematics 57, Elsevier (2004)
8. Heggernes, P., Kratsch, D.: Linear-time certifying recognition algorithms and forbidden induced subgraphs. Nordic Journal of Computing 14:87–108 (2007)
9. Heggernes, P., Meister, D., Villanger, Y.: Induced Subgraph Isomorphism on interval and proper interval graphs. In: Proceedings of ISAAC 2010, LNCS, vol. 6507, Springer, Heidelberg (2010)
10. van 't Hof, P., Kamiński, M., Paulusma, P., Szeider, S., Thilikos, D. M.: On contracting graphs to fixed pattern graphs. In: Proceedings of SOFSEM 2010, LNCS, vol. 5901, pp. 503–514, Springer, Heidelberg (2010)
11. Kamiński, M., Paulusma, D., Thilikos, D. M.: Contractions of planar graphs in polynomial time. In: Proceedings of ESA 2010, LNCS, vol. 6346, pp. 122–133, Springer, Heidelberg (2010)
12. Levin, A., Paulusma, D., Woeginger, G. J.: The computational complexity of graph contractions I: polynomially solvable and NP-complete cases. Networks 51:178–189 (2008)
13. Levin, A., Paulusma, D., Woeginger, G. J.: The computational complexity of graph contractions II: two tough polynomially solvable cases. Networks 52:32–56 (2008)
14. Lovász, L.: Coverings and colorings of hypergraphs. In: Proceedings of the 4th Southeastern Conference on Combinatorics, Graphs Theory, and Computing, pp. 3–12, Utilitas Mathematica Publishing, Winnipeg (1973)
15. Mahadev, N., Peled, U.: Threshold graphs and related topics. Annals of Discrete Mathematics 56, North Holland (1995)
16. Matoušek, J., Thomas, R.: On the complexity of finding iso- and other morphisms for partial k -trees. Discrete Mathematics 108(1–3):343–364 (1992)

17. Robertson, N., Seymour P.D.: Graph Minors .XIII. The Disjoint Paths Problem. *Journal of Combinatorial Theory, Series B* 63(1) (1995)
18. Semple, C., Steel M.: *Phylogenetics*. Oxford University Press graduate series Mathematics and its Applications (2003)
19. Wolk, E. S.: The comparability graph of a tree. In: *Proceedings of the American Mathematical Society*, vol. 13, pp. 789–795 (1962)
20. Wolk, E. S. A note on “The comparability graph of a tree”. *Proceedings of the American Mathematical Society*, vol. 16, pp. 17–20 (1965)
21. Yan, J.-H., Chen, J.-J., Chang, G. J.: Quasi-threshold graphs. *Discrete Applied Mathematics* 69:247–255 (1996)