Minimal split completions*

Pinar Heggernes† Federico Mancini†

Abstract

We study the problem of adding an inclusion minimal set of edges to a given arbitrary graph so that the resulting graph is a split graph, called a minimal split completion of the input graph. Minimal completions of arbitrary graphs into chordal and interval graphs have been studied previously, and new results have been added recently. We extend these previous results to split graphs by giving a linear-time algorithm for computing minimal split completions. We also give two characterizations of minimal split completions, which lead to a linear time algorithm for extracting a minimal split completion from any given split completion.

We prove new properties of split graph that are both useful for our algorithms and interesting on their own. First, we present a new way of partitioning the vertices of a split graph uniquely into three subsets. Second, we prove that split graphs have the following property: given two split graphs on the same vertex set where one is a subgraph of the other, there is a sequence of edges that can be removed from the larger to obtain the smaller such that after each edge removal the modified graph is split.

1 Introduction

Split graphs are the class of graphs whose vertices can be partitioned into an independent set and a clique; a graph class which is well studied and of wide theoretical interest and use [8]. Any graph can be embedded into a split graph by adding edges, and the resulting split graph is called a *split completion* of the input graph. A *minimum* split completion is a split completion with the minimum number of edges, and computing such split completions is an NP-hard problem [18]. A split completion H of a given graph G is *minimal* if no proper subgraph of H is a split completion of G. In this paper we show that a minimal split completion of a given graph can be computed in linear time.

Minimum and minimal chordal completions, also called *triangulations*, and minimum and minimal interval completions are defined analogously, by replacing split with chordal and with interval. Computing a minimum triangulation

*A preliminary version of this work will appear at LATIN 2006. This work is supported by the Research Council of Norway through grant 166429/V30.

†Department of Informatics, University of Bergen, N-5020 Bergen, Norway. Emails: pinar@ii.uib.no and federico@ii.uib.no

and computing a minimum interval completion of a graph are NP-hard problems [7, 21], whereas it was shown already in 1976 that minimal triangulations can be computed in polynomial time [20]. In mid-1990s minimal triangulations began to be studied again, and new characterizations have been given [5, 15, 19], which made new algorithms possible. Since then, the interest in minimal completion problems has increased, which has led to faster algorithms for minimal triangulations [16, 17, 13] and the knowledge that minimal interval completions can be computed and characterized in polynomial time [11, 12]. Minimal split completions have not been studied earlier, and with this paper we expand the knowledge about classes of graphs into which minimal completions of arbitrary graphs can be computed in polynomial time.

Minimal triangulations are well studied, and several characterizations of them have been given [10]. An algorithmically useful characterization is that a triangulation is minimal if and only if no single fill edge can be removed without destroying chordality of the triangulation [20] (fill edges are the edges added to the original graph to obtain a completion). This property does not hold for minimal interval completions. In this paper, we show that it holds for minimal split completions. In fact, we show the following more general result on split graphs: Between a split graph $G_1 = (V, E_1)$ and a split graph $G_2 = (V, E_2)$ with $E_1 \subset E_2$, there is a sequence of split graphs that can be obtained by repeatedly removing one single edge from the previous split graph, starting from G_2. This result is known to be true for chordal graphs [20, 1], and it has been posed as an open problem for chordal bipartite graphs [2]. We characterize the fill edges that are candidates for removal when a non-minimal split completion H of an arbitrary graph G is given. Based on this, we give linear-time algorithms both for computing minimal split completions, and for removing edges from a given split completion to obtain a minimal split completion. In order to obtain our results, we define a new way of partitioning the vertices of a given split graph, called a *3-partition*. This partition is always unique, which is not the case for the traditional way of partitioning the vertices of a split graph into an independent set and a clique.

This paper is organized as follows. In the next section we give the necessary graph theoretical background, assuming that the reader is familiar with the basic notions of graphs theory. In Section 3 we present our results on split graphs sandwiched between two given split graphs, and use these results to characterize minimal split completions. A new way of partitioning the vertices of a split graph uniquely is presented in Section 4, and this is used to give another characterization of minimal split completions in Section 5. These results are then combined to give an algorithm for removing redundant fill edges from a split completion to obtain a minimal split completion in Section 6, and an algorithm for directly computing a minimal split completion of an arbitrary input graph in Section 7. We conclude with some discussions and examples in Section 8.

2 Definitions and background

All graphs in this paper are simple and undirected. For a graph $G = (V, E)$, we let $n = |V|$ and $m = |E|$. The set of neighbors of a vertex $v \in V$ is denoted by $N(v)$, and the degree of a vertex v is denoted by $d(v) = |N(v)|$. We distinguish between subgraphs and induced subgraphs. In this paper, a *subgraph* of $G = (V, E)$ is a graph $G_1 = (V, E_1)$ with $E_1 \subseteq E$, and a *supergraph* of G is a graph $G_2 = (V, E_2)$ with $E \subseteq E_2$. We will denote these relations informally by the notation $G_1 \subseteq G \subseteq G_2$ (proper subgraph relation is denoted by $G_1 \subset G$). The complement of G is denoted by $\bar{G}$.

A simple cycle on k vertices is denoted by C_k and a complete graph on k vertices is denoted by K_k. Thus $2K_2$ is the graph that consists of 2 isolated edges. A graph is *chordal* if it contains no induced simple cycle of length at least 4. A subset K of V is a *clique* if K induces a complete subgraph of G. A subset I of V is an *independent set* if no two vertices of I are adjacent in G. We use $\omega(G)$ to denote the size of a largest clique in G, and $\alpha(G)$ to denote the size of a largest independent set in G.

G is a *split graph* if there is a partition $V = I + K$ of its vertex set into an independent set I and a clique K. Such a partition is called a *split partition* of G. There is no restriction on the edges between vertices of I and vertices of K. The partition of a split graph into a clique and an independent set is not necessarily unique. The following theorem from [9] states the possible partition configurations.

Theorem 1 (Hammer and Simeone [9]) *Let G be a split graph whose vertices have been partitioned into an independent set I and a clique K. Exactly one of the following conditions holds:*

(i) $|I| = \alpha(G)$ *and* $|K| = \omega(G)$
(the partition $I + K$ is unique).
(ii) $|I| = \alpha(G)$ *and* $|K| = \omega(G)-1$
(there exists a vertex $x \in I$ such that $K \cup \{x\}$ is a clique).
(iii) $|I| = \alpha(G)-1$ *and* $|K| = \omega(G)$
(there exists a vertex $y \in K$ such that $I \cup \{y\}$ is an independent set).

The following theorem characterizes split graphs, and we will use condition *(iii)* to prove one of the characterizations of minimal split completions that we present.

Theorem 2 (Földes and Hammer [6]) *Let G be an undirected graph. The following conditions are equivalent:*

(i) G is a split graph.
(ii) G and $\bar{G}$ are chordal graphs.
(iii) G contains no induced subgraph isomorphic to $2K_2, C_4$ or C_5.

Remark 3 *Every induced subgraph of a split graph is also a split graph.*
Graph classes satisfying this property are called *hereditary.*

For a given arbitrary graph $G = (V, E)$, a split graph $H = (V, E \cup F)$, with $E \cap F = \emptyset$, is called a *split completion* of G. The edges in F are called *fill edges*. H is a *minimal* split completion of G if $(V, E \cup F')$ fails to be a split graph for every proper subset F' of F.

(Minimal) chordal and interval completions of a given graph are defined analogously to (minimal) split completions. Chordal completions are also called *triangulations*. Both interval graphs and split graphs are chordal. For a chordal graph G, $\alpha(G)$ and $\omega(G)$ can be computed in linear time [8], whereas these are NP-hard problems for general graphs.

3 Sandwiching a split graph between two given split graphs

Given two chordal graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, such that $E_1 \subset E_2$, Rose, Tarjan, and Lueker [20] showed that there is an edge in $E_2 \setminus E_1$ whose removal from G_2 results in a chordal graph. A consequence of this result is that a given triangulation H of an arbitrary graph G is minimal if and only if it is impossible to obtain a chordal graph by removing a single fill edge from H. In this section, we show that analogous results hold for split graphs and minimal split completions. First we need the following observations.

Observation 4 *Let $G = (V, E)$ and $G' = (V, E')$ be two split graphs with $E \subseteq E'$, and let $V = I + K$ and $V = I' + K'$ be two split partitions of G and G', respectively. Then $|K' \cap K| \geq |K| - 1$.*

Proof. Assume that $|K' \cap K| < |K| - 1$. Then at least two vertices from K must be in the independent set I', but since they belong to a clique in G they must be connected in G' as well, which gives a contradiction. ∎

Observation 5 *Let $G = (V, E)$ and $G' = (V, E')$ be two split graphs with $E \subseteq E'$, and let $V = I + K$ and $V = I' + K'$ be two split partitions of G and G', respectively. Then $K' \setminus I \subseteq K$.*

Proof. We know that $V \setminus I = K$. Since $K' \subseteq V$, then $K' \setminus I \subseteq K$. ∎

Lemma 6 *Given two split graphs $G = (V, E)$ and $G' = (V, E \cup F)$, such that $E \cap F = \emptyset$ and $F \neq \emptyset$, there is an edge $f \in F$ that can be removed from G' so that the result is a split graph.*

Proof. Let $V = I + K$ be a split partition of G, and let $V = I' + K'$ be a split partition of G'. If there is an edge $f \in F$ with one endpoint in I' and one endpoint in K', then f can be removed, and the resulting graph is split with split partition $V = I' + K'$. Assume for the rest of the proof that there is no fill edge between I' and K'.

We define the set $T = K' \cap I$, namely those vertices that belong to an independent set in the partition of G and to a clique in the partition of G'. According to our assumption, there is no fill edge between T and I'. Thus each edge in F has either both endpoints in T or is between a vertex of T and a vertex of $K' \setminus T$, since $K' \setminus T$ was already a clique in G, by Observation 5. It follows that if $F \neq \emptyset$ then $T \neq \emptyset$, and all vertices in T must be incident to some edge of F in G'. Since T is a part of an independent set in G and a part of a clique in G', there are fill edges between each pair of vertices in T. If $|T| = 1$ the fill edges connect T to $K' \setminus T$.

By Observation 4 we now have two possible situations: either $|K' \cap K| = |K|$ or $|K' \cap K| = |K| - 1$.

Assume first that $|K' \cap K| = |K|$. Then $K \subseteq K'$, and consequently $I' \subseteq I$. This means that no vertex of T is adjacent to a vertex of I' in G', because there can neither be original edges between these two sets since $T \cup I' = I$, nor edges from F since $T \subset K'$. In such a situation it is possible to pick a vertex $y \in T$ incident to one or more edges in F, and remove any of the fill edges incident to y. Doing this the graph will remain split because we can still partition it in an independent set $I' \cup \{y\}$ and a clique $K' \setminus \{y\}$. We proved above that such vertex y must exist.

Let $|K' \cap K| = |K| - 1$. Then there must be a vertex x that in G belongs to K and in G' belongs to I', such that $(I' \setminus \{x\}) \subseteq I$. Now, each vertex of T can be adjacent to at most one vertex of I', namely x. If there is at least one vertex $y \in T$ which is not adjacent to x, then we can proceed as the previous case. If all vertices of T are adjacent to x, then $N(x) = K'$, so we can just swap x with any vertex in $y \in T$ incident to an edge of F, and remove this edge, since it now connects the independent set to the clique. Swapping the vertices we make a new partition where x is in the clique and y in the independent set, and thus the result is a split graph. ∎

The following corollary is an immediate consequence of Lemma 6.

Corollary 7 *Given two split graphs $G = (V, E)$ and $G' = (V, E \cup F)$ with $E \cap F = \emptyset$, there is a sequence of split graphs G_0, G_1, G_2, ..., $G_{|F|}$ such that G_{i-1} is obtained by removing edge f_i from G_i, for $1 \leq i \leq |F|$, where $G_0 = G$, $G_{|F|} = G'$, and $F = \{f_1, f_2, ..., f_{|F|}\}$.*

From the proof of Lemma 6 we can deduce the following corollary and obtain an algorithm to generate a sequence of split graphs G_1, G_2, ..., $G_{|F|-1}$ between $G_{|F|} = G' = (V, E \cup F)$ and $G_0 = G = (V, E)$, as mentioned in Corollary 7.

Corollary 8 *Let $G = (V, E)$ and $G' = (V, E \cup F)$ be two split graphs with $E \cap F = \emptyset$, and let $V = I + K$ and $V = I' + K'$ be two split partitions of G and G', respectively. If no edge of F is between I' and K', then either no vertex in $K' \cap I$ has neighbors in I', or a subset of them has exactly one common neighbor in I'.*

Algorithm SplitGraphSequence
Input: A split graph $G = (V, E)$ with split partition $V = I + K$, and a split graph $G' = (V, E \cup F)$ with split partition $V = I' + K'$ and $E \cap F = \emptyset$.
Output: A sequence of split graphs $G_0 = G$, G_1, G_2, ..., $G_{|F|} = G'$ such that G_{i-1} is obtained by removing a single edge of F from G_i.

```
i = |F|; G_i = G';
T = K' ∩ I; T' = T;
while there is a fill edge f between I' and K' in G_i do
  Obtain G_{i-1} by removing f from G_i;
  i = i − 1;
end-while
if F ≠ ∅ then
  Choose x to be a vertex of G_i of maximum degree belonging to I';
  if d_{G_i}(x) ≥ |K| then
    T = T \ N_{G_i}(x);
  end-if
end-if
while F ≠ ∅ do
  if T ≠ ∅ then
    Pick any t ∈ T;
    T' = T' \ {t};
    T = T \ {t};
    K' = K' \ {t};
    I' = I' ∪ {t};
    while there is a fill edge f incident to s do
      Obtain G_{i-1} by removing f from G_i;
      i = i − 1;
    end-while
  else
    K' = K' ∪ {x};
    I' = I' \ {x};
    T = T';
  end-if
end-while
```

This algorithm can clearly be implemented to run in total time $O(|V| + |E| + |F|)$. Its correctness follows from the proof of Lemma 6. First of all we remove one by one the fill edges between I' and K'. Then we check if there is a vertex x in I' that was in K, with at least a neighbor in I, meaning that $d_G(x) = d_{G_i}(x) = |K|$. If there is, we let $T = T \setminus N_{G_i}(x)$, so that we will remove first all non-neighbors of x in K', maintaining at the same time all $K' \cap I$ in T'. When x is adjacent to all vertices of the current K', that is $T = \emptyset$ for the first time, we move x to K', so that we can reset $T = T'$, and keep removing all remaining vertices of I from K'. Notice that when $T = \emptyset$ again, $F = \emptyset$ as well, so the algorithm is correct. The output can simply be the sequence of fill edges

removed at every step, which uniquely defines the sequence of split graphs of the algorithm.

Theorem 9 *Given an arbitrary graph G and a split completion G' of G, G' is a minimal split completion if and only if no single fill edge can be removed from G' without destroying the split property.*

Proof. If G' is a minimal split completion then no subset of its fill edges can be removed, so no single fill edge can be removed either. If G' is not a minimal split completion, another split graph G'' exists between G and G'. Then by Lemma 6, there is a single fill edge that can be removed from G' while preserving the split property. ■

Thus we have a characterization of minimal split completions. We will use this to give another and algorithmically more useful characterization of minimal split completions, and to describe the fill edges that can be removed from non-minimal split completions in Section 5. First, in the next section, we define a new way of partitioning the vertices of a split graph uniquely.

4 Unique 3-partitions of split graphs

In this section, as an alternative to split partitions, we define another way of partitioning the vertices of a split graph that will be useful to decide whether a given split completion is minimal or not. We will call the new partition a *split 3-partition*.

When we are given a non-minimal split completion of an arbitrary graph, according to Lemma 6, the redundant fill edges can be removed one by one until we reach a minimal split completion. The edges that can be removed without problems are the ones connecting the independent set with the clique in the split partition of the completion. However, since this partition is not necessarily unique, and since we do not know the underlying minimal split completion, we cannot simply use Algorithm SplitGraphSequence, and problems occur according to cases *(ii)* and *(iii)* of Theorem 1. To avoid this ambiguity we define a third set of vertices in the graph, that we will call Q. In case we do not have a unique split partition, this set will contain those vertices that can be chosen to be either in the independent set or in the clique, determining different partitions.

Definition 10 *Given a split graph $G = (V, E)$ that has no unique split partition, we define a* split 3-partition $V = S + C + Q$ *of G as follows:*

$$S = \{v \in V \mid d(v) < \omega(G) - 1\}$$
$$C = \{v \in V \mid d(v) > \omega(G) - 1\}$$
$$Q = \{v \in V \mid d(v) = \omega(G) - 1\}$$

If G has a unique split partition $V = I + K$, we do not need such a 3-partition, but for completeness, we define $S = I$, $C = K$, and $Q = \emptyset$ in this case, so that a split 3-partition is always defined. (Note that there can be vertices of degree

$\omega(G) - 1$ in G also when its split partition is unique.) For a split graph G, $\omega(G)$, $\alpha(G)$, and the corresponding maximum clique and independent set can be computed in linear time [8]. Thus it can be decided by Theorem 1 whether G has a unique split partition or not. Hence the 3-partition of a split graph is uniquely defined. Figure 1 shows examples of 3-partitions of some split graphs.

Lemma 11 *Let $G = (V, E)$ be a split graph with no unique split partition, and let $V = S + C + Q$ be the 3-partition of G. Then all of the following conditions hold.*

(i) $S \subseteq I$ *and* $C \subseteq K$*, for every split partition $V = I + K$ of G.*
(ii) $Q \neq \emptyset$*.*
(ii) Q *is exactly the set of vertices each of which belongs to a clique and to an independent set in two different split partitions of G, respectively.*

Proof.

(i) Let $V = I + K$ be any split partition of G. By Theorem 1, each vertex of K belongs also to a clique of maximum size, and thus has degree at least $\omega(G) - 1$. Therefore, a vertex that has degree less than $\omega(G) - 1$ cannot belong to K, and it must belong to I. A vertex of I can be adjacent to at most $\omega(G) - 1$ vertices, because otherwise we have a clique of size $\omega(G) + 1$. Thus, a vertex that has degree more than $\omega(G) - 1$ must belong to K.

(ii) Let $V = I + K$ be any split partition of G. By Theorem 1, either there is a vertex x in I such that $K \cup \{x\}$ is a clique, or there is a vertex y in K such that $I \cup \{y\}$ is an independent set. In either case, each such vertex x or y is adjacent to all vertices of K and to no other vertex, and by Theorem 1, it has degree exactly $\omega(G) - 1$. Thus $Q \neq \emptyset$.

(iii) By the argument in *(ii)* every vertex that can be moved between an independent set and a clique in some split partition of G must have degree $\omega(G) - 1$. Let us show that each vertex of degree $\omega(G) - 1$ can indeed be moved between partitions. Let $V = I + K$ be any split partition of G, and let v be a vertex of degree $\omega(G) - 1$. Assume first that $v \in I$. If $|K| = \omega(G) - 1$, then by moving v from I to K, we get another split partition of G. If $|K| = \omega(G)$, then we know by Theorem 1 that a vertex x of K can be moved to I to give a different split partition. Thus x cannot be adjacent to v. We can swap v and x between I and K, and get a new split partition. Assume now that $v \in K$. If $|K| = \omega(G) - 1$, then there must be a vertex z in I that is adjacent to all vertices of K. Thus z is the only neighbor of v outside of K. We can swap z and v and get another partition. If $|K| = \omega(G)$ then v has no neighbors in I and we can move v from K to I and get a new partition. ∎

The next two corollaries follow directly from the proof of Lemma 11.

Corollary 12 *A split graph $G = (V, E)$ has a unique partition $V = I + K$ if and only if there are exactly $\omega(G)$ vertices of degree $> \omega(G) - 1$.*

Corollary 13 *Let $G = (V, E)$ be a split graph with 3-partition $V = S + C + Q$. Then every vertex of Q is adjacent to all vertices of C and to no vertex of S.*

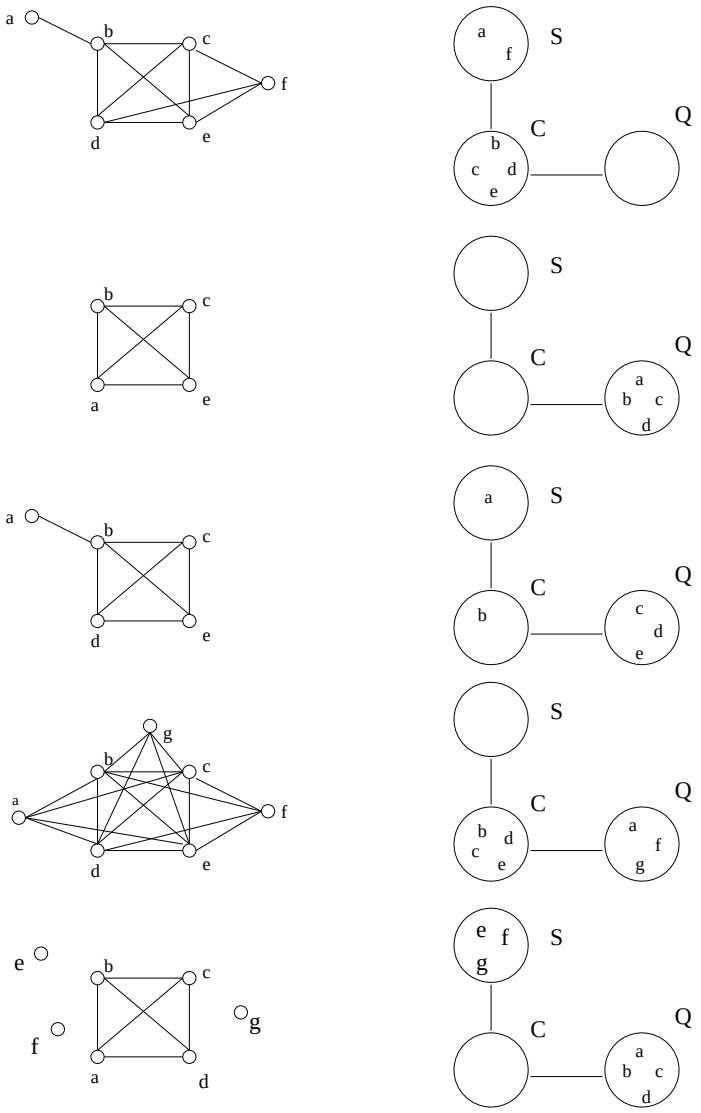


Figure 1: *Examples of 3-partitions for different kind of split graphs.*

Lemma 14 *Let $G = (V, E)$ be a split graph with 3-partition $V = S + C + Q$. Then one of the following is true:*

(i) Q is a clique and $|C| + |Q| = \omega(G)$.
(ii) Q is an independent set, $|C| = \omega(G) - 1$, and $|Q| \geq 2$.

Proof. If Q is empty (i) is true by Theorem 1. Assume that $|Q| \geq 1$, so that the split partition of G is not unique. If $\omega(G) = 1$ then G consists of vertices that all have degree 0, thus $Q = V$ is an independent set and C is empty, which means that (ii) is true. If $\omega(G) > 1$ then $|C|+|Q| \geq \omega(G)$, so we can distinguish between two situations: $|C| + |Q| = \omega(G)$ or $|C| + |Q| > \omega(G)$.

If $|C| + |Q| = \omega(G)$ then $|Q|$ is a clique, because the largest clique in G must have size $\omega(G)$ and it can only be obtained by adding to C all vertices of Q.

If $|C| + |Q| > \omega(G)$ then we will show that $|C| = \omega(G) - 1$ and Q is an independent set. If $|C| < \omega(G) - 1$ then $|Q| > 2$, and there must be at least a subset $Q' \subset Q$ that is a clique of size $\omega(G) - |C| \geq 2$. All vertices of Q' have degree $\omega(G) - 1$, and since they make a clique of size $\omega(G)$ with C, they cannot be adjacent to any vertex in $Q \setminus Q'$. The vertices in $Q \setminus Q'$ must be an independent set, or it would not be possible to make a split partition $V = (C \cup Q') + (S \cup (Q \setminus Q'))$, so they are adjacent only to C (by Corollary 13) and consequently each of them has degree at most $\omega(G) - 2$, contradicting the fact that they belong to Q. So we must have $|C| = \omega(G) - 1$ and $|Q| \geq 2$, which implies that Q is an independent set, or the size of the maximum clique would be $\omega(G) + 1$ by Corollary 13, giving a contradiction. ∎

Corollary 15 *Let $G = (V, E)$ be a split graph with 3-partition $V = S + C + Q$ and $Q \neq \emptyset$. Then in any split partition $V = I + K$ of G, at least $\omega(G) - 1 - |C|$ vertices of Q belong to K.*

Lemma 16 *Let $G = (V, E)$ be a split graph with 3-partition $V = S + C + Q$ and $Q \neq \emptyset$. If $|Q| = 1$, then $|C| = \omega(G) - 1$ and $|S| \geq 2$. If $|C| + |Q| = \omega(G)$ and $|Q| > 1$, then $|S| \geq 1$.*

Proof. Every vertex of C has degree greater than $\omega(G) - 1$, so every vertex of C has at least one neighbor in S (since $|C|+|Q| = \omega(G)$), but every vertex of S has degree at most $\omega(G)-2$, so at least two of them are needed to be connected to C in the first case, and at least one in the last. ∎

In the next section, we will use these results to give a new characterization of minimal split completions of arbitrary graphs.

5 Characterizing minimal split completions

Assume that we are given an arbitrary graph $G = (V, E)$ and a split completion $H = (V, E \cup F)$ of G. We want to find a sufficient and necessary condition for H to be a minimal split completion of G. First we identify the fill edges that can be removed from any non-minimal split completion. Note that, when

$V = S + C + Q$ is the 3-partition of H, any fill edge is either incident to a vertex of $S \cup Q$ or both of its endpoints belongs to C.

Lemma 17 *Let $H = (V, E \cup F)$ be a split completion of an arbitrary graph $G = (V, E)$, and let $V = S + C + Q$ be the 3-partition of H. Then any fill edge incident to a vertex in $S \cup Q$ can be removed so that the resulting graph is split.*

Proof. We will prove that there is a split partition $V = I + K$ of H such that any fill edge incident to a vertex of $S \cup Q$ has one endpoint in I and one endpoint in K, and we know that such edges can be removed. We also know that all vertices of S belong to the independent set of any split partition of H, and all vertices of C belong to the clique of any split partition of H. This means that any edge between S and C can be removed. Let us then assume that there are no fill edges connecting S and C. Remember also that there are no edges between S and Q. Let us also assume that the partition is not unique, so that $Q \neq \emptyset$. Under these assumptions, each fill edge incident to a vertex of $S \cup Q$ can only be between two vertices of Q or between a vertex of Q and a vertex of C, and we have the following cases.

Case 1: Q is a clique and there is a fill edge between two vertices $x, y \in Q$. If Q is a clique, then we can make a partition where at most one of the vertices of Q is chosen to be in the independent set of a partition and all the others must be in the clique. If we put x (or y) in the independent set and y (or x) in the clique, there will be a fill edge (xy) between the independent set and the clique, that we can remove.

Case 2: There is a fill edge between Q and C. If a fill edge is between a vertex $x \in Q$ and a vertex $y \in C$, since we can always choose at least one vertex of Q to be in the independent set of a partition regardless of whether Q is a clique or an independent set, let us choose exactly x. Since y is in C it will always be in the clique of any partition of H, so we now have a fill edge connecting a vertex of the independent set (x) to a vertex of the clique (y), and we can remove it. ∎

Note that Lemma 17 does not mean that all fill edges incident to $S \cup Q$ can be removed. We are guaranteed to be able to remove one such edge. After that the 3-partition of the resulting graph might change, and thus the set of fill edges that can be removed might also change.

Lemma 18 *Let $H = (V, E \cup F)$ be a split completion of an arbitrary graph $G = (V, E)$, and let $V = S + C + Q$ be the 3-partition of H. If each fill edge has both its endpoints in C, then H is a minimal split completion of G.*

Proof. Assume that G, H, S, C, and Q are as in the premise of the lemma such that all fill edges of H have both their endpoints in C. Thus if $F \neq \emptyset$ then $|C| \geq 2$ and $\omega(H) \geq 2$. We show that removing any single fill edge from H results in a non-split graph.

If Q is empty and thus H has a unique split partition, then by Theorem 1 and Corollary 12, $|C| = \omega(H)$, no vertex of S is adjacent to whole C, and

every vertex of C has a neighbor in S. Hence $|S| \geq 2$. If $|C| = 2$ then the single edge in C is a fill edge. After removing it we would get a $2K_2$, because we can pick two vertices in S adjacent each to only one vertex of C. If $|C| > 2$ then removing a fill edge we get two nonadjacent vertices $x, y \in C$. Now, x and y must each have a neighbor in S. If they have a common neighbor w, then we can find a vertex $v \in C$ which is not adjacent to w, since no vertex of S is adjacent to every vertex of C. But x and y are both adjacent to v, so this results in an induced cycle w, x, v, y, w of length 4. If they do not have a common neighbor, then there exist $w, z \in S$, where w is adjacent to x and not to y, and z is adjacent to y and not to x, So removing the edge between x and y we get a $2K_2$.

If $Q \neq \emptyset$ then S can be even empty or disconnected from C. Let us work on Q and C using Lemma 14 and 16.

In the case when $|C| + |Q| = \omega(H)$, we have that Q is a clique and $S \neq \emptyset$. If $|Q| = 1$, then $|C| = \omega(H) - 1$, so there must be at least 2 vertices in S adjacent to vertices of C, and no vertex of S is adjacent to every vertex of C, so we can use the same argument as above. If $|Q| > 1$, then $|C| < \omega(H) - 1$. Thus every vertex of C has a neighbor in S, and either we have the previous case, or there is a vertex $z \in S$ adjacent to all vertices in C. Recall that every vertex of Q is connected to all vertices of C and to no vertex of S. Let us now take any $x, y \in C$ and remove the edge xy. We can find a vertex $w \in Q$, adjacent to both x and y so that the subgraph induced by $\{z, x, w, y\}$ is a cycle of length 4.

In the case when $|C| + |Q| > \omega(H)$, we have that Q is an independent set, and $|C| = \omega(H) - 1$. In this case S can be empty. However, since $|C| \geq 2$, then $\omega(H) \geq 3$ and $|Q| \geq 2$. Since every vertex of Q is adjacent to all vertices of C, we can find two vertices w and z in Q, such that if we remove any fill edge xy from C, we get an induced cycle w, x, z, y, w of length 4. ∎

Now we are ready to state another characterization of minimal split completions, which describes the redundant fill edges more closely than the previous one.

Theorem 19 *Let $H = (V, E \cup F)$ be a split completion of an arbitrary graph $G = (V, E)$, and let $V = S + C + Q$ be the 3-partition of H. H is a minimal split completion of G if and only if all fill edges have both endpoints in C.*

Proof. One direction follows from Lemma 18. For the other direction assume that H is minimal. Then no single fill edge can be removed without destroying split property. Given the 3-partition $V = S + C + Q$, then each fill edge can have one endpoint in $S \cup Q$ and one endpoint in C, or both endpoints in C or in Q. This is because there cannot be fill edges between S and Q (by Corollary 13), and within S (it is an independent set). By Lemma 17 a fill edge incident to vertices in $S \cup Q$ can always be removed, so since the completion is minimal, the only possible fill edges are the ones in C. ∎

6 Obtaining a minimal split completion from a given split completion

In the next section we will give an algorithm that computes a minimal split completion of any given graph. However, for some applications it might be desirable to compute a minimal split completion that fits within an already given split completion. This problem has been studied and solved for triangulations [4, 3] and interval completions [12], and we solve it for split completions in this section.

Assume that we are given an arbitrary graph G, a split completion H of G, and the 3-partition $S + C + Q$ of H. Then the following is a direct consequence of Theorem 19.

Corollary 20 *Let $H = (V, E + F)$ be a split completion of an arbitrary graph $G = (V, E)$, and let $V = S + C + Q$ be the 3-partition of H. H is a minimal split completion of G if and only if, for all fill edges $uv \in F$, $d_H(u) \geq \omega(H)$ and $d_H(v) \geq \omega(H)$.*

Proof. By the definition of 3-partition, a vertex belongs to C if and only if its degree is at least $\omega(H)$, and the corollary follows from Theorem 19. ∎

The following algorithm formalizes the above corollary, and we will show that it produces a minimal split completion from any given split completion, in linear time.

Algorithm ExtractMinimal
Input: A graph $G = (V, E)$, and a set of fill edges F such that $H = (V, E \cup F)$ is split;
Output: A minimal split completion $M = (V, E \cup F')$ of G, with $F' \subseteq F$;

```
M = H; F' = F; ω(M) = ω(H);
minimal = false;
for each f = uv ∈ F' do
  d(f) = min{d_M(u), d_M(v)};
while (not minimal) and (F' ≠ ∅) do
  Let f = uv be an element of F' with smallest d(f);
  if d(f) < ω(M) then
    Remove f from M and F';
    for all fill edges f' incident to u or v do
        Update d(f');
    Update ω(M);
  else
    minimal = true;
  end-if;
end-while;
return M and F';
```

Lemma 21 *Given a graph $G = (V, E)$ and a split completion $H = (V, E \cup F)$ of G, Algorithm ExtractMinimal computes a minimal split completion $M = (V, E \cup F')$ of G, with $F' \subseteq F$.*

Proof. Notice that at each step, if the **if** condition is satisfied, the algorithm removes from the current graph M a fill edge with an endpoint of degree smaller than $\omega(M)$. By the definition of 3-partition, a vertex with degree smaller than $\omega(M)$ belongs to $S \cup Q$ in the 3-partition $V(M) = S + C + Q$, and by Lemma 17, removing a fill edge incident to such vertices, always keeps the graph split. Hence, at every step, the current graph M is split completion of G.

When the **if** condition is not satisfied, all fill edges have both endpoints of degree at least $\omega(M)$. Hence by Corollary 20, the current split graph M is a minimal split completion of G. Notice that the algorithm always terminates, and if $F' = \emptyset$, then G was a split graph. ∎

Theorem 22 *Let $H = (V, E \cup F)$ be a split completion of an arbitrary graph $G = (V, E)$, with $F \cap E = \emptyset$. A minimal split completion M of G, such that $G \subseteq M \subseteq H$, can be computed in time $O(|V| + |E| + |F|)$.*

Proof. We will show that Algorithm ExtractMinimal can be used to compute such a minimal split completion in linear time. We have already proved the correctness of this algorithm in Lemma 21, now we will prove that it runs in linear time.

Notice that most operations in Algorithm ExtractMinimal can be performed in time linear in the size of H, that is $O(|V| + |E| + |F|)$, thanks to previous results on split graphs: The size of the maximum clique of H can be found in linear time [9], and we can keep it updated in constant time after each edge deletion using the dynamic algorithm for split graphs given by Ibarra [14]. This means that, since we have at most $O(|F|)$ iterations in our algorithm, the overall running time is $O(|V| + |E| + |F|)$ if we can prove that updating M and F' and finding an edge with an endpoint of minimum degree in F' does not cost more than $O(1)$ time for each step. To remove an edge from M takes clearly constant time, and we give now a data structure that allows us also to update and find the minimum of F' in constant time at each step, after a linear time preprocessing.

First of all we create a *degree* array of size $|V|$, and for each element d of the array, we create a doubly linked list of the vertices of H with degree d, incident to at least a fill edge in F. For each vertex in a list, we create a list of the fill edges incident to it. Since every fill edge will appear in the list of exactly two vertices, we add a pointer between these two copies of the same fill edge. Eventually we add a pointer to the minimum non-empty element of the degree array. To set up this structure clearly takes time $O(|V| + |E| + |F|)$.

Given the pointer to the minimum element of the degree array, it takes constant time to find the fill edge with endpoint of minimum degree in F'. In fact we only need to follow the pointer, find the first vertex in the corresponding list, and pick the first fill edge f adjacent to it. If the minimum degree of a vertex in the array is greater or equal than $\omega(M)$, there are no fill edges in the

graph with an endpoint of degree smaller than that, and the algorithm can stop because all fill edges are in the set C of the 3-partition of M by Observation 20. In the opposite case, we delete the first fill edge f we found as described above, and we update the data structure and the pointer to the minimum non-empty element of the degree array. We will show that the update after each deletion can be done in constant time, so that the theorem will follow.

First of all we delete f and its copy, in constant time. This can be done since we set up a pointer between the two copies of every fill edge. Then we check whether the two endpoints of f are still adjacent to some fill edges or not. Let us call v the endpoint of minimum degree of f, and u the other endpoint. As far as u is concerned, if it is still incident to some fill edges, we simply move it in constant time to the list corresponding to its new degree, that is exactly one less than its old degree, so we move by it one position in the array. Otherwise we remove it from the data structure in constant time. When v has still some fill edges incident to it, v becomes the new minimum of the array, since it was in the list of the minimum non-empty element of the array before, and now its degree decreased by one. Notice that this means that, if we remove another fill edge after f, it will be again incident v since it is the new minimum, and we will keep doing this until there will be no fill edges incident to v. Besides, updating the degree of u and v, we automatically also updated the degree of all fill edges incident to any of these two vertices, and when v is incident to more fill edges than f, we need to move the pointer to the minimum only by one position at the time, therefore using constant time. Now, once we removed all the fill edges incident to v, we need to remove v from the data structure and find new minimum non-empty element of the array. This can take more than constant time, but we can show that it never takes more than $O(|V|+|F|)$ time for a sequence of $O(|F|)$ deletions, proving that we get an amortized constant time per deletion. If we have removed k fill edges incident to v, in order to find the new minimum we might have to scan all k positions through which v has moved since we started removing edges from it, and possibly further in the array. Therefore, to update the pointer to the minimum after a sequence of k constant time deletions, takes at most time $O(k)$ plus the time to scan some other elements in the array, that, however, we had never scanned before. Hence, to delete k fill edges incident to the same vertex v, and update the pointer to the minimum of the array, takes at most time $O(2k+|V_i|)$ where V_i is a subset of consecutive elements of the array that we visit for the first time. If we sum up all the deletion sequences performed by the algorithm, since we know that at most we can delete $|F|$ edges and there are $|V|$ elements in the array, we obtain an overall running time of $O(|V|+|F|)$ for $O(|F|)$ deletions.

Concluding, the whole algorithm runs in time $O(|V|+|E|+|F|)$ as claimed.

■

7 Computing a minimal split completion directly

In this section, we show that minimal split completion of a given graph can be computed in time linear in the size of the input graph. A simple and intuitive method to embed an arbitrary graph $G = (V, E)$ into a split graph, is to select a maximal independent set I of G, and add edges to make $V \setminus I$ into a clique. This procedure does not guarantee that the resulting split completion is minimal, since it can add edges even to a graph that is already split, in particular if the graph does not have a unique partition, as illustrated in Figure 2. However, it can be modified to compute a minimal split completion, by choosing vertices of minimum degree first when computing the maximal independent set. We call this modified algorithm *MinimalSplit* and present it below.

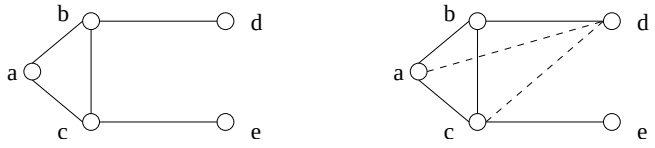


Figure 2: *The graph on the left is already split, but choosing the set $M = \{b, e\}$ as the maximal independent set and making the rest a clique, we get a larger split graph.*

Algorithm MinimalSplit
Input: An arbitrary graph $G = (V, E)$;
Output: A minimal split completion $H = (V, E \cup F)$ of G.
$I = \emptyset$; $K = \emptyset$;
Unmark all vertices;
while there are unmarked vertices in V **do**
Choose an unmarked vertex v with minimum degree in G;
Mark and add v to I;
Mark and add all neigbors of v to K;
end-while
Make K into a clique adding a set F of fill edges;
$H = (V, E \cup F)$;

Lemma 23 *Given an arbitrary graph $G = (V, E)$, the graph $H = (V, E \cup F)$ computed by Algorithm MinimalSplit is a minimal split completion of G.*

Proof. Let $V = I + K$ be the split partition of H computed by the algorithm. By construction, I is an independent set, K is a clique, and no edges are added between I and K, so H is a split graph. It follows from Lemma 18 that if $V = I + K$ is a unique partition of H, then H is a minimal split completion.

Let us consider the case when $V = I + K$ is not a unique partition of H, and let $V = S + C + Q$ be the 3-partition of H. If $|K| < 2$ then no edges are added by the algorithm, so the completion is trivially minimal. Assume therefore that $|K| \geq 2$. By construction, $K = \bigcup_{v \in I} N(v)$, so every vertex of K has a neighbor in I. It follows that $|K| = \omega(H) - 1$. Otherwise (if $|K|$ were $\omega(H)$), there

would be $\omega(H)$ vertices in K with degree greater then $\omega(H)-1$, contradicting Corollary 12. Since the split partition is not unique, there is at least one vertex z of degree $\omega(H)-1$ in I, that can be moved to K by Theorem 1. Such vertices z belong to Q, but they are not adjacent to any fill edge. Consequently, the only possibility for the completion to be non-minimal is that a vertex $x \in K$ incident to a fill edge, has degree $\omega(H)-1$ so that x belongs to Q. Thus x has exactly one neighbor in I. This means that there is exactly one vertex $y \in I$ of degree $\omega(H)-1$, since vertices of degree $\omega(H)-1$ in I must be adjacent to all vertices of K. So we have exactly one vertex $y \in I$ of degree $\omega(H)-1$ and a vertex $x \in K$ of degree $\omega(H)-1$, such that $N(x) \cap I = \{y\}$. The degree of x in the input graph G is actually less than $\omega(H)-1$, because it is incident to at least one fill edge. But the degree of y is $\omega(G)-1$ also in G since no edges are added to vertices in I. This means that $d(x) < d(y)$ in G, but a vertex can be in K only if one of its neighbors in G has been selected before it to be in I. In this case, since the only neighbor of x in G selected to be in I is y, it means that y has been processed by the algorithm before x, but that is a contradiction because $d(x) < d(y)$, and the algorithm always chooses the vertex with minimum degree among the unprocessed ones.

This means that any graph obtained by the algorithm is a minimal split completion of the input graph by Lemma 18. ∎

Let us consider the time complexity of this algorithm. Since we add edges only between the vertices of K, we can actually skip the step of adding edges, because the resulting split partition will uniquely define the edges of H. Hence the algorithm can be modified to return just I and K (the edges of H that are between I and K are also edges of G). The degrees are computed only in the beginning of the algorithm, and need not be recomputed. This and the rest of the algorithm clearly require at most $\sum_{v \in V} d(v)$ steps, which sums up to time $O(|V|+|E|)$. Thus we have the following result.

Theorem 24 *A minimal split completion of an arbitrary graph $G=(V,E)$ can be computed in time $O(|V|+|E|)$.*

8 Concluding remarks

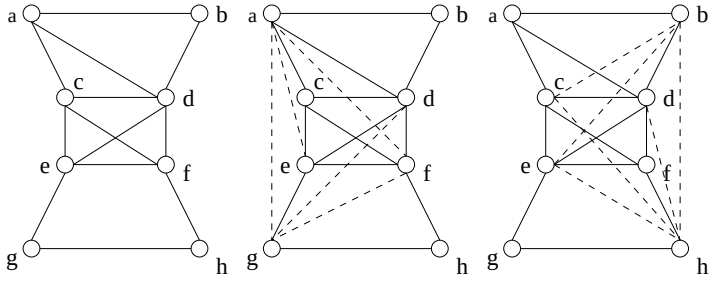
We have given two characterizations of minimal split completions and we have shown how to compute minimal split completions in linear time. We have also given an algorithm for computing a minimal split completion between the input graph G and an already given non-minimal split completion H of G. To achieve these goals, we introduced a new way of uniquely partitioning the vertices of a split graph into three subsets instead of two, and we proved the following general result on split graphs: between two split graphs on the same vertex set where one is a subgraph of the other, there is a sequence of split graphs obtained by removing one single edge from the previous.

Let us consider the last mentioned property. As already explained, chordal graphs have this property, there are easy examples to show that interval graphs

do not, and it is an open question whether bipartite chordal graphs do. We would like to know which other graph classes have this property. If a graph class $\mathcal{C}$ has this property and the edges that are candidates for removal can be identified in polynomial time, then minimal $\mathcal{C}$-completions can be computed in polynomial time, since we can start from a trivial completion (e.g. the complete graph) and remove fill edges until we reach a minimal completion.

Another interesting question is whether minimal $\mathcal{C}$-completions can be computed in polynomial time for every hereditary graph class $\mathcal{C}$ that contains complete graphs and that can be recognized in polynomial time. In fact, given such a graph class $\mathcal{C}$, let us define the following problems: I. Computing a minimal $\mathcal{C}$ completion of an arbitrary input graph, and II. Deciding whether a given $\mathcal{C}$ completion H of an arbitrary graph G is minimal. Note that a polynomial time algorithm for problem I does not necessarily imply a polynomial time solution for problem II. For chordal graphs and interval graphs, both problems are polynomially solvable, and we have shown in this paper that this is also the case for split graphs. We would like to know whether there are graph classes $\mathcal{C}$ for which problem I is solvable in polynomial time, whereas problem II is NP-hard.

Finally, we would like to mention that even if we were able to pick a maximum independent set of the arbitrary input graph G, we are not guaranteed to get a minimum split completion by completing the rest of the vertices into a clique, as illustrated in Figure 3. We suspect that one always gets a minimal split completion with this approach, but this will in any case not give a practical algorithm for general graphs, since finding a maximum independent set is an NP-hard problem.



(a) Inputgraph (b) Completion1 (c) Completion2

Figure 3: *Given the input graph in (a), we can find at least two different maximum independent sets:* $M_1 = \{b, c, h\}$ *and* $M_2 = \{a, f, g\}$. *In (b) we make* $V \setminus M_1$ *into a clique by adding 5 edges; In (c) we make* $V \setminus M_2$ *into a clique by adding 6 edges. This means that even when we take a maximum independent set, we cannot be sure to get a minimum completion.*

Acknowledgment

The authors would like to thank Jan Arne Telle for an initial discussion on the example given in Figure 2.

References

[1] M. Bakonyi and T. Constantinescu. Inheritance Principles for Chordal Graphs. *Linear Algebra Appl.*, 148:125–143, 1991.

[2] M. Bakonyi and A. Bono. Several results on chordal bipartite graphs. *Czechoslovak Mathematical Journal*, 47(7):577 – 583, 1997.

[3] A. Berry, J-P. Bordat, P. Heggernes, G. Simonet, and Y. Villanger. A wide-range algorithm for minimal triangulation from an arbitrary ordering. *J. Algorithms*, 58(1):33 – 66, 2006.

[4] J. R. S. Blair, P. Heggernes, and J. A. Telle. A practical algorithm for making filled graphs minimal. *Theor. Comput. Sci.*, 250:125–141, 2001.

[5] V. Bouchitté and I. Todinca. Treewidth and minimum fill-in: Grouping the minimal separators. *SIAM J. Comput.*, 31:212–232, 2001.

[6] S. Földes and P. L. Hammer. Split graphs. *Congressus Numerantium*, 19:311–315, 1977.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman and Co., 1978.

[8] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Second edition. Annals of Discrete Mathematics 57. Elsevier, 2004.

[9] P. L. Hammer and B. Simeone. The splittance of a graph. *Combinatorica*, 1(3):275–284, 1981.

[10] P. Heggernes. Minimal triangulations of graphs - A survey. *Dicrete Math.*, 306(3):297–317, 2006.

[11] P. Heggernes, K. Suchan, I. Todinca, and Y. Villanger. Minimal interval completions. In *Algorithms - ESA 2005*, pages 403 – 414. Springer Verlag, 2005. LNCS 3669.

[12] P. Heggernes, K. Suchan, I. Todinca, and Y. Villanger. Characterizing minimal interval completions: Towards better understanding of profile and pathwidth. In *Proceedings of Annual Symposium on Theoretical Aspects of Computer Science - STACS 2007*, Springer Verlag, LNCS, to appear.

[13] P. Heggernes, J. A. Telle, and Y. Villanger. Computing minimal triangulations in time $O(n^\alpha \log n) = o(n^{2.376})$. In *SIAM Journal on Discrete Math.*, 19(4):900–913, 2005.

[14] L. Ibarra. Fully dynamic algorithms for chordal graphs and split graphs. Technical Report DCS-262-IR, University of Victoria, Canada, 2000.

[15] T. Kloks, D. Kratsch, and J. Spinrad. On treewidth and minimum fill-in of asteroidal triple-free graphs. *Theor. Comput. Sci.*, 175:309–335, 1997.

[16] D. Kratsch and J. Spinrad. Between $O(nm)$ and $O(n^\alpha)$. In *Proceedings of SODA 2003 - 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 709–716, 2003.

[17] D. Kratsch and J. Spinrad. Minimal fill in $O(n^{2.69})$ time. *Dicrete Math.*, 306(3):366–371, 2006.

[18] A. Natanzon, R. Shamir, and R. Sharan. Complexity classification of some edge modification problems. *Disc. Appl. Math.*, 113:109–128, 2001.

[19] A. Parra and P. Scheffler. Characterizations and algorithmic applications of chordal graph embeddings. *Disc. Appl. Math.*, 79:171–188, 1997.

[20] D. Rose, R.E. Tarjan, and G. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 5:266 – 283, 1976.

[21] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM J. Alg. Disc. Meth.*, 2:77–79, 1981.