Dynamically maintaining split graphs*

Pinar Heggernes† Federico Mancini†

August 2, 2007

Abstract

We present an algorithm that supports operations for modifying a split graph by adding edges or vertices and deleting edges, such that after each modification the graph is repaired to become a split graph in a minimal way. In particular, if the graph is not split after the modification, the algorithm computes a minimal, or if desired even a minimum, split completion or deletion of the modified graph. The motivation for such operations is similar to the motivation for fully dynamic algorithms for particular graph classes. In our case we allow all modifications to the graph and repair, rather than allowing only modifications that keep the graph split. Fully dynamic algorithms of the latter kind are known for split graphs [24].

Our results can be used to design linear time algorithms for some recognition and completion problems, where the input is supplied in an on-line fashion.

1 Introduction

Split graphs are a well-studied graph class discovered independently at the end of the '60s by Gyárfäs and Lehel [6] and Hammer and Földes [14]. They can be recognized in time $O(n+m)$ for input graphs with n vertices and m edges [15, 16] and they have very important algorithmic and theoretical relevance [3, 15, 34, 35] as well as various generalizations [4, 2, 30, 31]. A fully dynamic algorithm for split graphs, namely an algorithm that supports queries for edge addition and deletion, and update after deleting or adding an edge if this keeps the graph split, was given by Ibarra [23, 24]. His algorithm is based on the characterization of split graphs by their degree sequences [16], and each query or update for edge addition or edge deletion takes constant time, after an initial linear-time preprocessing. The algorithm that we present in this paper is based on a more powerful structure: the 3-partition defined recently by the authors [17]. This structure allows us to support the following additional operations: adding a new vertex or edge to the graph and deleting an edge from the graph also when this operation does not keep the graph split, in which case we can repair it into a split graph in a minimal way. In particular, in the addition case, the user has the choice to add a minimal set of edges to repair the graph, while in the edge deletion case, a minimal set of edges can be removed from the current graph to make it split again.

The motivation for studying particular graph classes is that inputs arising from some applications satisfy certain properties. Using these properties, problems that are intractable in general can be solved in polynomial time for these inputs. Many times the data corresponding to the input can be erroneous, such that the assumed properties are not satisfied. In such cases, it is desirable to be able to repair the input in a minimal or minimum way to fit the properties of the application. Furthermore, data and desired modifications might be supplied in an on-line fashion, which is particularly interesting for maintaining databases. This is the main motivation behind studying minimal and minimum modifications explained above, and similar operations have been given for cographs, chordal, interval and comparability graphs [7, 20, 18, 29]. Vertex incremental algorithms are well studied algorithms where the input graph

*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

is supplied one vertex at a time. There exists a variety of them for various problems and, because of their nature, they often follow from dynamic algorithms [1, 5, 7, 9, 10, 11, 12, 13, 27, 39]. The operation for adding a new vertex that we present in this paper, for example, very easily leads to the following two new vertex incremental algorithms: A certifying algorithm that recognizes split graphs in time linear in the size of the input graph, and an algorithm for computing a minimal split completion of an arbitrary graph in time linear in the size of the output graph. Note that for both problems, algorithms exist for solving them within these time bounds [17, 19], however none of the existing algorithms is vertex incremental.

In our algorithm, adding a new vertex x, with a given neighborhood N_x in the current graph, takes $O(|N_x| + |A_x|)$ time, where A_x is the minimal set of additional neighbors that are given to x to keep the graph split. In the case of adding an edge uv, the minimal update operation adds either an inclusion minimal set of edges A_u incident to u or an inclusion minimal set of edges A_v incident to v, in addition to uv, and the time required for this update is either $O(d(u) + |A_u|)$ or $O(d(v) + |A_v|)$, according to which endpoint we choose. In case of deleting an edge uv, the minimal update operation removes an inclusion minimal set of edges from one of the endpoints of uv, and the time required for this update is $O(d(v))$ or $O(d(u))$, according to the chosen endpoint. If a modification step does not require repairing, then the minimal additional set will be empty and the update can be performed in constant time. We would like to mention that, if desired, our algorithm can be used as a fully dynamic algorithm for split graphs by turning off the operations that allow changes that destroy the split property. In that case, all operations match the running time of Ibarra's algorithm [24].

Finally, our dynamic algorithm allows us also to add or delete in linear time a *minimum* set of additional edges when a modification does not keep the graph split and the update is still desired. From this follows that the following problems can be solved in linear time: Computing minimum split completions of split$+1v$ and split$+1e$ graphs, and computing minimum split deletions of split$-1e$ graphs (these are the graph classes that contain all graph that can be obtain, respectively, adding a vertex, adding an edge, and deleting an edge from a split graph). Similar results have been shown for cographs [37]. In addition we can also give a certifying algorithm for recognizing such classes in linear time, using a slight modification of the incremental algorithm for the recognition of split graphs. Although it follows from a result of Cai [8] that these classes can be recognized in polynomial time, it was not known whether they could be recognized in linear time and, in particular, no certification algorithm was given.

We would like to mention that when we repair the graph in a minimal way, the additional set of edges are always added incident to one single vertex. We will give a more technical explanation for this in the last section, but the main motivation is that we want ensure that such operations are useful for vertex incremental algorithms as mentioned above and because it is reasonable that a user do not want to introduce changes in the input graph that is already split. When the user chooses to repair in a minimum way, instead, edges are added or deleted incident to several vertices.

In the next section, we introduce some notation and list some results from [17] that will be used in this paper. In Section 3 we characterize the situations where a vertex addition, an edge addition, and an edge deletion of a split graph keeps the graph split. The operations for updating and repairing the graph in a minimal way after vertex addition, edge addition, and edge deletion, are given in Sections 4.2.1, 4.2.4, and 4.2.2, respectively. In Section 5 we show how to repair the graph in a minimum way, and finally in Section 6 we discuss the algorithms that can easily be obtained from our results.

2 Definitions and background

All graphs in this paper are simple and undirected. For a graph $G = (V, E)$, we let $n = |V|$ and $m = |E|$. We will also use $V(G)$ and $E(G)$ to denote the vertex and edge sets of a graph, respectively. The set of *neighbors*, or the *neighborhood* of a vertex $v \in V$ is denoted by $N_G(v) = \{u \mid uv \in E\}$, or simply $N(v)$ if there is no ambiguity, and the *degree* of v is $d(v) = |N(v)|$. The neighborhood of a set of vertices $S \subseteq V$ is defined as $N(S) = \bigcup_{x \in S} N(x) \setminus S$. A vertex is called *universal* if it is adjacent to all other vertices in the graph, and it is called *isolated* if it has no neighbors. An *induced subgraph* of $G = (V, E)$ over a set of vertices $U \subseteq V$, is the graph $G[U] = (U, E_U)$, where $E_U = \{uv \in E \mid u, v \in U\}$. The *complement* $\overline{G}$ of

G consists of all vertices of G and $uv \in E(\overline{G})$ if and only if $uv \notin E(G)$.

A set of vertices $K \subseteq V$ is a *clique* if $G[K]$ is complete. A set of vertices $I \subseteq V$ 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. Although computing $\alpha(G)$ and $\omega(G)$ are NP-hard problems for an arbitrary graph G, when G is a split graph $\alpha(G)$ and $\omega(G)$ can be computed in linear time [15, 16].

A graph $G = (V, E)$ 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. Split graphs can be recognized and a split partition can be found in linear time [16]. A split partition is not necessarily unique. The following theorem from [16] states the possible partition configurations.

Theorem 1 (Hammer and Simeone [16]) *Let $G = (V, E)$ be a split graph with a split partition $V = I + K$. Exactly one of the following conditions holds:*

1. *$|I| = \alpha(G)$ and $|K| = \omega(G)$*
   (in this case the partition $I + K$ is unique),
2. *$|I| = \alpha(G)$ and $|K| = \omega(G) - 1$*
   (in this case there exists a vertex $x \in I$ such that $K \cup \{x\}$ is a clique),
3. *$|I| = \alpha(G) - 1$ and $|K| = \omega(G)$*
   (in this case there exists a vertex $y \in K$ such that $I \cup \{y\}$ is independent).

Split graphs are hereditary, which means that every induced subgraph of a split graph is also a split graph [15]. For the following result, note that a simple cycle on k vertices is denoted by C_k and that a complete graph on k vertices is denoted by K_k. Thus $2K_2$ is the graph that consists of 2 isolated edges. Chordal graphs are the graphs that do not contain an induced C_k for $k \geq 4$.

Theorem 2 (Földes and Hammer [14]) *Let G be an undirected graph. The following conditions are equivalent:*

1. *G is a split graph.*
2. *G and $\overline{G}$ are chordal graphs.*
3. *G contains no induced subgraph isomorphic to $2K_2, C_4$ or C_5.*

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 *minimum* split completion of G if there is no set F' with $|F'| < |F|$ such that $(V, E \cup F')$ is a split graph. Minimum split completions are NP-hard to compute [36]. 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 split completions can be computed in linear time [17]. A *minimal split deletion* is an edge-maximal split subgraph on the same vertex set.

A graph is *split*$+1v$ (*split*$+1e$) if it contains a vertex (edge) whose deletion from the graph results in a split graph. A graph is *split*$-1e$ if a split graph can be obtained by adding an edge to it.

Throughout the paper we will be using results from [17], most of which are based on a new kind of partition for split graphs. A 3-*partition* of the vertices V of a split graph $G = (V, E)$ consists of three sets $V = S + C + Q$ (Stable set, Clique and eQuivalent vertices) defined as follows:

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

This partition is obviously unique, however it has been defined only to overcome the ambiguity caused by the many possible split partitions a split graph can have; therefore, when a graph has a unique split partition $V = I + K$, we set $S = I$, $C = K$ and $Q = \emptyset$, so that S might contain also the vertices of degree $\omega(G) - 1$.

The uniqueness of a split partition can be checked by Theorem 1 and by the following lemma, hence it is easy to maintain the 3-partition correctly. In our dynamic algorithm, we will have a global variable

that indicates at all times whether or not the current graph has a unique split partition, thus Q will be empty exactly when the graph has a unique split partition.

The following properties follows quite easily from the definition of 3-partition, but they will be very useful to simplify many of the proofs in the paper.

Lemma 3 ([17]) *A split graph $G = (V, E)$ with 3-partition $V = S + C + Q$ has a unique split partition $V = I + K$ if and only if $|C| = \omega(G)$.*

Property 4 ([17]) *Given a split graph $G = (V, E)$ and its 3-partition $V = S + C + Q$, any split partition $V = I + K$ of G satisfies $S \subseteq I$ and $C \subseteq K$.*

Property 5 ([17]) *Let $G = (V, E)$ be a split graph with 3-partition $V = S + C + Q$. Every vertex of Q is adjacent to all vertices of C and to no vertex of S.*

Lemma 6 ([17]) *Let $G = (V, E)$ be a split graph with 3-partition $V = S + C + Q$. One of the following is true:*

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

Property 7 *If a split graph $G = (V, E)$ with 3-partition $V = S + C + Q$ has a unique split partition or Q is a clique, then each vertex in C has at least one neighbor in S.*

Proof. It has been proved in [17] that this is true when G has a unique partition. When Q is a clique, $Q \cup C$ is the maximum clique of the graph by Lemma 6, so let us assume for the sake of contradiction that a vertex $c \in C$ does not have any neighbor in S. This means that all neighbors of c are in $C \cup Q$, hence $d(c) = \omega - 1$, contradicting the fact the it belongs to C by the definition of 3-partition. ∎

Notice that when there is only one vertex in Q, Q is both an independent set and a clique, but we assume it to be a clique unless we specify differently. This means that if Q is an independent set, so it contains at least 2 vertices, then S might be even empty, so that the vertices of C do not necessary have a neighbor in S.

The following theorem is one of the main results in [17] and characterizes minimal split completions, so it will be useful for the update operations when we need to repair the graph in a minimal way.

Theorem 8 ([17]) *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 each fill edge has both its endpoints in C.*

Finally we give two results that will allow us to express deletion operations in terms of addition and vice versa, thanks to the fact that split graphs are self complementary.

Lemma 9 *Let $G = (V, E)$ be a split graph with 3-partition $V = S + C + Q$. Then $V = \overline{S} + \overline{C} + Q$ is the 3-partition of $\overline{G}$ with $\overline{S} = C$ and $\overline{C} = S$.*

Proof. Let us denote $\omega = \omega(G)$, $\omega(\overline{G}) = \overline{\omega}$ and α and $\overline{\alpha}$ similarly. By Theorem 1 we know that either $|V| = \omega + \alpha$ if the split partition is unique, or $|V| = \omega + \alpha - 1$ otherwise. Also, since split graphs are perfect, we know that $\alpha = \overline{\omega}$, hence either $|V| = \omega + \overline{\omega}$ or $|V| = \omega + \overline{\omega} - 1$. If the split partition of G is unique, it follows immediately that also the split partition of $\overline{G}$ is unique, so $Q = \overline{Q} = \emptyset$ in both graphs. Also, in this case, we know that $d_G(s) \leq \omega - 1$ for each $s \in S$ and $d_G(c) > \omega - 1$ for each $c \in C$. Hence $d_{\overline{G}}(s) \geq (|V| - 1) - (\omega - 1)$ and $d_{\overline{G}}(c) < (|V| - 1) - (\omega - 1)$. Substituting $|V|$ we get $d_{\overline{G}}(s) \geq \overline{\omega} > \overline{\omega} - 1$ and $d_{\overline{G}}(c) < \overline{\omega} \leq \overline{\omega} - 1$, proving that $S = \overline{C}$ and $C = \overline{S}$. If the split partition is not unique, we can use exactly the same argument with $|V| = \omega + \overline{\omega} - 1$, $d_G(s) < \omega - 1$, $d_G(c) > \omega - 1$ and $d(q) = \omega - 1$ for each vertex $q \in Q$. In this case we get $d_{\overline{G}}(s) > \overline{\omega} - 1$, $d_{\overline{G}}(c) < \overline{\omega} - 1$ and $d_{\overline{G}}(q) = \overline{\omega} - 1$, proving the statement. ∎

From Lemma 9 and Theorem 8, it is straightforward to give the following.

Theorem 10 *Let $H = (V, E \setminus F)$ be a split deletion 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 the endpoints of the deleted edges are in S.*

3 Characterizing vertex/edge addition and edge removal

In this section we give three theorems to characterize, in terms of 3-partition, when the addition of a vertex, the addition of an edge and the deletion of an edge from a split graph, keeps the graph split. We start with the vertex addition. Also, given the theorem for the edge addition, the edge deletion will easily follow by the self-complementarity of split graphs.

Theorem 11 *Given a split graph $G = (V, E)$, its 3-partition $V = S + C + Q$, and a vertex $x \notin V$ with a set of neighbors $N_x \subseteq V$, $G_x = (V \cup \{x\}, E \cup \{xy \mid y \in N_x\})$ is a split graph if and only if one of the following holds:*

1. *$|N_x| \leq \omega(G) - 1$, $N_x \subset C \cup Q$, and Q is a clique or empty*
2. *$|N_x| \leq \omega(G) - 1$, $N_x \subset C \cup Q$, $|N_x \cap Q| \leq 1$, and Q is an independent set*
3. *$|N_x| > \omega(G) - 1$, $C \subseteq N_x$, $|N_x \cap Q| \geq |Q| - 1$, and Q is a clique or empty*
4. *$|N_x| > \omega(G) - 1$, $C \subseteq N_x$, and Q is an independent set*

Proof.

(Only if) For this direction of the proof, we assume that G_x is split, and for each possible case we show that one of the four listed conditions must hold or we can find a forbidden subgraph in G_x.

Case 1:$|N_x| \leq \omega(G) - 1$ and Q is a clique or empty. In this case we prove that G_x is split only if $N_x \subset C \cup Q$, so that Condition 1 holds. Let us keep in mind that in this case $|C| + |Q| = \omega(G)$ by Lemma 6. Assume for the sake of contradiction that x is adjacent to a vertex $s \in S$. Then x has at most $\omega(G) - 2$ neighbors belonging to $C \cup Q$. Besides, if Q is a clique with at least two vertices, then x must have at least $|Q| - 1$ neighbors in Q or we get a $2K_2$ induced by $\{q_1, q_2, x, s\}$, for two vertices $q_1, q_2 \in Q \setminus N_x$. We can conclude that if $S \cap N_x \neq \emptyset$, there exists at least one vertex $c \in C$ that is not adjacent to x in G_x. Now we consider three situations.

If $N_x \cap N(c) \cap S = \emptyset$ then by Property 7 there exists $s_1 \in N(c) \cap (S \setminus N_x)$ such that $\{c, s_1, x, s\}$ induces a $2K_2$ in G_x.

If $N_x \cap N(c) \cap S \geq 2$, let $s_1, s_2 \in N_x \cap N(c) \cap S$. Then $\{c, s_1, x, s_2\}$ induces a C_4 in G_x.

If $N_x \cap N(c) \cap S = \{s_1\}$, then we have some cases to consider. If x has at least one neighbor $q \in Q$, we get a $C_4 = \{x, s_1, c, q\}$. If x has no neighbors in Q, then Q has either one single element q or it is empty. In either case s_1 cannot be adjacent to all C, hence there exists at least another vertex $c_1 \in C$ that is not adjacent to s_1. If x is adjacent to c_1 we get a $C_4 = \{x, s_1, c, c_1\}$. Otherwise we can find a vertex $s_2 \in S$ adjacent to c_1 by Property 7, such that, either x is incident to s_2 and we get a $C_5 = \{x, s_1, c, c_1, s_2\}$, or, if s_2 is not adjacent to x, we get a $2K_2 = \{s, x, s_2, c_1\}$.

Case 2: $|N_x| \leq \omega(G) - 1$, and Q is an independent set. In this case we prove that Condition 2 must be satisfied, showing that $N_x \subset C \cup Q$ and $|N_x \cap Q| \leq 1$. We can assume that Q is an independent set of size at least two, or we are in the previous case with $|Q| = 1$. We first show that x cannot be adjacent to more than one vertex of Q in G_x. Assume on the contrary that x is adjacent to at least two vertices q_1 and q_2 of Q. Thus there is at least one vertex c in C which x is not adjacent to, because $|C| = \omega(G) - 1$. Since every vertex of Q is adjacent to every vertex of C by Property 5, the set $\{x, q_1, c, q_2\}$ induces a C_4 in G_x. Hence $|N_x \cap Q| \leq 1$. Now, we must also show that $N_x \subset C \cup Q$, so we assume for the sake of contradiction that x is adjacent to at least one vertex $s \in S$. This implies that, since $|N_x \cap (C \cup Q)| \leq \omega(G) - 2$ and $|C| = \omega(G) - 1$, there exists at least one vertex $c \in C$ to which x is not adjacent. If x is adjacent to a vertex $q \in Q$, either c is also incident to s and we get a $C_4 = \{x, s, c, q\}$, or for any other vertex $q_1 \in (Q \setminus \{q\})$ there is $2K_2 = \{x, s, c, q_1\}$. Let us now assume x has no neighbors in Q. If c is not incident to s we get a $2K_2 = \{x, s, c, q\}$ for any $q \in Q$, otherwise there exists a vertex $c_1 \in C$ that is not incident to s (or s would belong to Q), and either we get a $2K_2 = \{x, s, c_1, q\}$ if x is not incident to c_1, or a $C_4 = \{x, s, c, c_1\}$ otherwise. All possibilities lead to a forbidden induced subgraph, contradicting that G_x is split, hence $S \cap N_x = \emptyset$.

Case 3: $|N_x| > \omega(G) - 1$ *and* Q *is a clique or empty.* In this case we show that Condition 3 must hold, namely $C \subseteq N_x$ and $|N_x \cap Q| \geq |Q| - 1$, or G_x is not a split graph. When x has no edges to S, then $|N_x| = \omega(G)$, meaning that it is adjacent to all vertices of C and Q, so Condition 3 is satisfied. What is left to show is that when x is adjacent to at least one vertex $s \in S$, then it must also be adjacent to at least $|Q| - 1$ vertices of Q and all C. Assume on the contrary that there are two vertices q_1 and q_2 in Q which are not adjacent to x in G_x. Since s is not adjacent to any vertex of Q, we get a $2K_2 = \{x, s, q_1, q_2\}$, contradicting that G_x is split. Thus x is adjacent to at least $|Q| - 1$ vertices in Q. It remains to show that all vertices of C are adjacent to x in G_x. Assume on the contrary that there is a vertex $c \in C$ which is not in N_x, then the proof goes exactly as in case 1.

Case 4: $|N_x| > \omega(G) - 1$*, and* Q *is an independent set.* In this case we show that G_x cannot be split unless $C \subseteq N_x$, so that Condition 4 is satisfied. We can assume that $|Q| \geq 2$ or we can consider it a clique and use case 3. Assume for the sake of contradiction that there is a vertex c in C which is not adjacent to x in G_x. Then, by the proof of case 2 we know that x cannot be adjacent to more than one vertex of Q, hence, since its degree is at least $\omega(G)$, it must be adjacent to at least one vertex of S. In this situation we can use the rest of proof of case 2 to show that if $C \cap N_x \neq C$ we always get a forbidden subgraph, contradicting that G_x is split.

(If) Now we assume that one of the four conditions listed in the theorem holds, and we show that G_x is a split graph giving a split partition $V \cup \{x\} = I + K$ of it.

Assume that $|N_x| \leq \omega(G) - 1$, $N_x \subset C \cup Q$, and Q is a clique or empty. Then $I = S \cup \{x\}$ and $K = C \cup Q$.

Assume that $|N_x| \leq \omega(G) - 1$, $N_x \subset C \cup Q$, $|N_x \cap Q| \leq 1$, and Q is an independent set. Then $I = S \cup \{x\} \cup (Q \setminus \{q\})$ and $K = C \cup \{q\}$, where $q \in Q \cap N_x$.

Assume that $|N_x| > \omega(G) - 1$, $C \subseteq N_x$, $|N_x \cap Q| \geq |Q| - 1$, and Q is a clique or empty. Then $I = S \cup \{q\}$ and $K = C \cup \{x\} \cup (Q \setminus \{q\})$, if $Q \setminus N_x = \{q\}$, or $I = S$ and $K = C \cup Q \cup \{x\}$ otherwise.

Assume that $|N_x| > \omega(G) - 1$, $C \subseteq N_x$, and Q is an independent set or empty. Then $I = S \cup Q$ and $K = C \cup \{x\}$. ∎

Lemma 12 *Given a split graph $G = (V, E)$, its 3-partition $V = S + C + Q$, and two vertices $a, b \in V$ such that $ab \notin E$, $G_{ab} = (V, E \cup \{ab\})$ fails to be a split graph if and only if $a, b \in S$.*

Proof. There are four possibilities regarding which set each of a and b belongs to: they can both belong to S, one can belong to S and the other to either C or Q, or they can both belong to Q (only if Q is an independent set). We will show that G_{ab} is not split exactly in the first case.

Assume that $a, b \in S$. If $|Q| \geq 2$, then if Q is a clique, $\{a, b, q_1, q_2\}$ induces a $2K_2$ in G_{ab} for every two vertices $q_1, q_2 \in Q$. Otherwise, since Q is an independent set, there exists a vertex $c \in (C \setminus N(a))$. Now, if $c \notin N(b)$, then $\{a, b, c, q\}$ induces a $2K_2$ in G_{ab} for each $q \in Q$, otherwise there exists $c_1 \in N(a) \setminus N(b)$ and $\{a, b, c, c_1\}$ induces a C_4 in G_{ab}. Let us assume $|Q| \leq 1$. Then by Property 7 we know that each vertex $c \in C$ must have at least a neighbor in S, and each vertex in S must have at least a non-neighbor in C. Hence, if $N(a) \subseteq N(b)$ (or $N(b) \subseteq N(a)$), there must exists $c \in C \setminus N(b)$ and c must have a neighbor $s \in S$ with $s \neq a, b$. In this case $\{c, s, a, b\}$ induces a $2K_2$ in G_{ab}. Otherwise there exist a vertex $x \in N(a) \setminus N(b)$ and a vertex $y \in N(b) \setminus N(a)$ such that $\{x, a, b, y\}$ induces a C_4 in G_{ab}.

For the remaining cases, let us recall that we can always find a split partition of G where at least one vertex of Q is in K and another one in I, and $S \subseteq I$ for every I. Choosing such vertices to be the endpoints of ab, we always add the edge between K and I, so that $V = K + I$ is also a valid split partition of G_{ab}. Hence G_{ab} is split. ∎

From Lemma 9 and 12, the edge deletion rule follows.

Lemma 13 *Given a split graph $G = (V, E)$, its 3-partition is $V = S + C + Q$, and an edge $ab \in E$, $G^-_{ab} = (V, E \setminus \{ab\})$ fails to be a split graph if and only if $a, b \in C$.*

4 Implementation

4.1 The data structure

In this section we describe how to implement the 3-partition as a data structure and we give the details needed to justify the running time of the operations that will be analyzed in the next sections.

Given an input graph G represented with an adjacency list, as an initial preprocessing step, we can find the degrees of all vertices in linear time and store the values in a variable for each vertex. Using the degrees, $\omega(G)$ can be found in linear time, as well [16]. After this, the vertices can be partitioned into S, C, and Q according to their degrees, in linear time. The sets S and C, are each represented by an object containing a degree array of n elements where each element d of the array has two pointers: one to an object containing the label of the set, and one to a list of elements representing the vertices in that set with degree d. We call such a list, a degree list L_d^i, with $i \in \{S, C\}$. Each element in a degree list points both to the object with the label of the set it belongs to, and to the corresponding vertex of the graph. There can therefore be up to n labels for a set, but only one for each degree list. Besides, we keep a pointer to the maximum and minimum non empty element of the degree array, and, from the last element of a degree list, we have a double pointer to the first element of its and the next degree list in the array.

Notice also that, since in Q all vertices have always the same degree, we do not need a degree array, but just a simple list of elements representing the vertices in Q and a label to which they all point. However we maintain one more variable for Q that holds a value from $\{0, 1, 2\}$, indicating, respectively, whether Q is empty, a clique, or an independent set (of size at least 2, or we consider it a clique). This information can be updated and accessed in constant time.

Eventually, we also maintain a variable with the size of the maximum clique of the graph and each vertex of the graph has a double pointer to its corresponding element in a degree list.

This way, finding out which set a given vertex belongs to or its degree, deleting a given vertex from a set, and adding a given vertex to a set take each constant time. Consequently, moving a given set A of vertices, takes at most $O(|A|)$ time.

Given this structure, to scan or return a set of the 3-partition takes time linear in the size of the set, because the degree lists are all connected by pointers, so we can just scan them one after the other, starting with the one corresponding to the minimum degree in the array, to which we have a pointer as well.

As we said, the size of a degree array is n, but since we allow addition of vertices to the graph, for every such operation we should increase the size of the array by 1. This can be done in amortized constant time using dynamic arrays, or allocating from the beginning an array of size $n + n'$, where n' is an upper bound to the number of possible vertex additions given by the user of the algorithm. Notice that this is something every dynamic algorithm has to deal with, independently from the 3-partition.

Now we give a lemma about the running time of some specific operations on this data structure, that will be useful when analyzing the running time of the algorithms in the next sections.

Lemma 14 *Let $X, Y \in \{S, C, Q\}$ for the 3-partition $V = C + S + Q$ of a split graph $G = (V, E)$, and let A be a subset of the degree list L_d^X. All vertices of $X_d = L_d^X \setminus A$ can be moved to Y in time $O(\max\{|A|, 1\})$, if L_d^Y contains a constant number of elements.*

Proof. In $O(|A|)$ time it is possible to remove all elements of A from L_d^X, so that we get a new degree list $L_1 = A$ where all elements point to a new label X, while the current L_d^X is now equal to X_d. Now in constant time we can replace L_d^X with L_1 and attach X_d to L_d^Y. Since by assumption L_d^Y contains a constant number of elements, we can move their pointers from their current label to the label of X_d, and change the value of such label from X to Y.

Therefore the overall running time to move X_d to a set Y and update the current set X, is $O(\max\{|A|, 1\})$, since if $A = \emptyset$, $L_d^X = X_d$ and we can move it to Y in constant time. ∎

4.2 Operations

In this section we show how to update efficiently the data structure of the graph after the addition or deletion of a new vertex or edge. Except for the vertex deletion, in which case the resulting graph is always split, we also give algorithms to repair the graph in a minimal way after every modification that does not keep it split.

Let $G = (V, E)$ be the split graph given as input with its 3-partition. Given a vertex x incident to a set $N_x \subseteq V$, we show that a minimal split completion G' of $G_x = (V \cup \{x\},\ E \cup \{xy \mid y \in N_x\})$ and the corresponding 3-partition can be computed in time $O(|N_x| + |F|) = d_{G'}(x)$. It follows that if G_x is split, we can update its 3-partition in time $O(|N_x|)$. When deleting an edge b, we call the resulting graph G^-_{ab} and we give an algorithm that can compute a minimal split deletion G' of G^-_{ab} and the corresponding 3-partition in time $\min\{d_G(a), d_G(b)\}$. In particular, checking whether G^-_{ab} is still split, and update its 3-partition if so, can be done in constant time. Using the edge deletion algorithm we give a vertex deletion algorithm that, given a vertex x to remove from G, updates the 3-partition of $G - x$ in time $O(d_G(x))$. Finally, the edge addition case can be easily handled using a vertex deletion and a vertex addition operation. That is, given an edge ab to add to G, if the resulting graph G_{ab} is not split, we first remove the endpoint of ab of minimum degree from G, and then we add it back with the new edge ab incident to it. This means that a minimal split completion G' of G_{ab} and its 3-partition can be computed in time $O(\max\{d_{G'}(a), d_{G'}(b)\})$. If G_{ab} is split instead, we can update the 3-partition in constant time. Let us recall that, as we mentioned in the introduction, all fill edges will be adjacent only to the new vertex or to one of the endpoints of the added/deleted edge.

4.2.1 Vertex addition

We begin with an observation that will help simplifying the analysis of the running time. Then we give the algorithm that computes a minimal split completion G' of G_x and its 3-partition in case G_x is not split. Notice that we can always find a minimal split completion of G_x adding edges only to x, because adding a universal vertex to a split graph, keeps the resulting graph split.

Observation 15 *Let $G = (V, E)$ be a split graph with 3-partition $V = S + C + Q$ and $G_x = (V \cup \{x\}, E \cup E_x)$ the split graph with 3-partition $V \cup \{x\} = S' + C' + Q'$. Then $C \subseteq C'$ and $S \subseteq S'$.*

Proof. $C \subseteq C'$ because either $\omega(G) = \omega(G_x)$, or $\omega(G_x) = \omega(G)+1$ and x is incident to all C by Theorem 11. This means that the degree of the vertices of C remains strictly higher than $\omega(G_x) - 1$ when adding x, so that they will belong to C' in G_x. Using the same argument on $\overline{G}$ and $\overline{G_x}$, by Lemma 9 we get that $S \subseteq S'$. ∎

Theorem 16 *Given a split graph $G = (V, E)$ together with 3-partition $V = S + C + Q$, a vertex $x \notin V$, and a set $N_x \subseteq V$, a minimal split completion G' of $G_x = (V \cup \{x\},\ E \cup \{xy \mid y \in N_x\})$ and the 3-partition $V \cup \{x\} = S' + C' + Q'$ of G' can be computed in time $O(d_{G'}(x))$.*

Proof. Using Theorem 11 we can check whether G_x is split in $O(|N_x|)$ time. If so, $G_x = G'$ and its 3-partition can be computed in $O(|N_x|)$ time as well by Observation 15 and Lemma 14. By Observation 15, none of the vertices in C and S needs to be moved, hence the update consists in updating the degree of the vertices in N_x by one, just placing x in one of the three sets and possibly move either the set $N_x \cap Q$ to C, or $Q \setminus N_x$ to S, or both. This because all vertices in Q have the same degree. Therefore, when adding x, the degree of the vertices in $N_x \cap Q$ increases by one and they either stay in Q' if $\omega(G') = \omega(G) + 1$, or they must move to C'. On the other hand, the degree of the vertices in $Q \setminus N_x$ does not change, so they stay in Q' if $\omega(G') = \omega(G)$ or must move to S' otherwise. The running time follows from Lemma 14: At most $|N_x|$ vertices might have to go to C', so we can move them in $O(|N_x|)$ time; Moving $Q \setminus N_x$ to S' can be done again in $O(|N_x|)$ time because the set A of Lemma 14 in this case is exactly $N_x \cap Q$ and no vertices in S' have degree $\omega(G) - 1$.

Assume now that G_x is not split. Then we can find a minimal split completion G' of G_x and update the 3-partition in $O(d_{G'}(x))$ using Algorithm Minimal-vertex-completion.

Algorithm: Minimal-vertex-completion

Input: A split graph $G = (V, E)$, its 3-partition $V = S + C + Q$, a vertex $x \notin V$, and a set $N_x \subset V$.
Output: A minimal split completion G' of $G_x = (V \cup \{x\},\ E \cup \{xy \mid y \in N_x\})$, and the 3-partition $V \cup \{x\} = S' + C' + Q'$ of G'.

If $N_x \cap S \neq \emptyset$ and Q is a clique **then**
  If $|N_x \cap Q| \leq |Q| - 1$ **then** *(1a)*
    Let q be a vertex of $Q \setminus N_x$;
    $F = \{xy \mid y \in (C \setminus N_x) \cup (Q \setminus N_x \setminus \{q\})\}$;
    $S' = S \cup \{q\}$; $C' = C \cup (Q \setminus \{q\}) \cup \{x\}$; $Q' = \emptyset$;
  ElseIf $N_x \cap Q = Q$ **then** *(1b)*
    $F = \{xy \mid y \in C \setminus N_x\}$;
    $S' = S$; $C' = C \cup \{x\}$; $Q' = Q$;
  EndIf
ElseIf Q is an independent set **then** *(2)*
  $F = \{xy \mid y \in C \setminus N_x\}$;
  $S' = S \cup (Q \setminus N_x)$; $C' = C \cup \{x\}$; $Q' = N_x \cap Q$;
EndIf
$G' = (V \cup \{x\},\ E \cup \{xy \mid y \in N_x\} \cup F)$;

Let us prove the correctness of the algorithm first, keeping in mind that we want to add fill edges only incident to x. If Q is a clique, then G_x is not split only if $N_x \cap S \neq \emptyset$ and $N_x \cap C \neq C$ and/or $|N_x \cap Q| < |Q| - 1$. In this case we can only add edges to x in order to satisfy condition 3 of Theorem 11, namely make x incident to C and at least $|Q| - 1$ vertices of Q. Cases 1a and 1b of Algorithm Minimal-vertex-completion, cover all such possibilities. If Q is an independent set, then G_x is not split only if $N_x \cap C \neq C$, so the only way to fix the graph is covered by case 2 of Algorithm Minimal-vertex-completion, that make x incident to C satisfying condition 4 of Theorem 11.

Checking that the sets S', C', and Q' are correct is straightforward since, given F, we know $\omega(G_x)$ and the new degrees of the vertices of G. Finally, observe that in each case both endpoints of every edge in F are in C', ensuring that the computed split completion is minimal by Theorem 8.

We have already shown at the beginning of the proof that if G' if split, the 3-partition can be updated in time $O(|N_x|)$. Hence, given the fill edges incident to x that make the graph split again, we can perform the update in time $O(|N_x| + |F|) = O(d_{G'}(x))$. What is left to prove is that the set of vertices to be made incident to x can be found within the same time bound. By Algorithm Minimal-vertex-completion, we always make all vertices of C adjacent to x, but $|C| \leq |N_x| + |F|$, so this operation takes $O(|N_x| + |F|)$ time. Sometimes we also need to scan Q, but then we will make x adjacent to all vertices of Q except one vertex, meaning that $|C| + |Q| - 1 = |N_x| + |F|$, so that $O(|C| + |Q|) = O(|N_x| + |F|)$. ∎

We end this section with the following observation which we find interesting on its own, and which will be useful in section. 5

Observation 17 *Given a split graph $G = (V, E)$ and a vertex x with a set of neighbors $N_x \subseteq V$ in G, all possible minimal split completions of $G_x = (V \cup \{x\},\ E \cup \{xy \mid y \in N_x\})$, where all fill edges are incident to x, have the same number of edges.*

Proof. As can be seen in Algorithm Minimal-vertex-completion and the proof of Lemma 16, either we have to make x adjacent to all vertices of C, in which case the minimal completion is unique, or we have to make x adjacent to all vertices of C and all but one vertices of Q, in which case all minimal completions have the same number of fill edges. ∎

4.2.2 Edge deletion

In this section we handle the case when an edge ab is removed from G. We call the resulting graph G^-_{ab} and we give an algorithm to compute a minimal split deletion G' of G^-_{ab} and its 3-partition when removing ab does not keep the graph split. Remember that the edges are deleted from only one the endpoint of ab. Using our data structure we can perform the minimal split deletion in time $O(\min\{d_G(a), d_G(b)\})$, while, if G^-_{ab}, it is easy to show that the 3-partition can be updated in constant time.

Algorithm: Minimal-edge-deletion
Input: A split graph $G = (V, E)$, its 3-partition $V = S + C + Q$, and an edge $ab \in E$
Output: A minimal split deletion (maximal split subgraph) G' of $G^-_{ab} = (V, E \setminus \{ab\})$ and the 3-partition $V = S' + C' + Q'$ of G'.

$\omega = \omega(G)$;
If Q is a clique **then** *(1)*
  $F^- = \{av \mid v \in S \wedge av \in E\}$;
  $S' = S \cup \{a\}$; $Q' = \emptyset$; and $C' = C \setminus \{a\} \cup Q$;
ElseIf Q is an independent set **then** *(2)*
  Let q be a vertex in Q;
  $F^- = \{av \mid v \in (S \cup Q \setminus \{q\}) \wedge av \in E\}$;
  $S' = S \cup \{a\} \cup (Q \setminus \{q\})$; $Q' = \emptyset$; and $C' = (C \setminus \{a\}) \cup \{q\}$;
ElseIf Q is empty **then**
  $F^- = \{av \mid v \in S \wedge av \in E\}$;
  $W = \{x \mid x \in S \setminus N_G(a) \wedge d_G(x) = \omega - 1\}$;
  If $|W| = 0$ **then** *(3a)*
    $S' = S \cup \{a\}$; $Q' = \emptyset$; $C' = C \setminus \{a\}$;
  ElseIf $d(b) > \omega$ **then** *(3b)*
    $S' = S \setminus W \cup \{a\}$; $Q' = W$;$C' = C \setminus \{a\}$;
  ElseIf $W = \{s\}$ **and** $d(b) = \omega$ **then** *(3c)*
    $S' = (S \setminus \{s\}) \cup \{a\}$; $Q' = \{b, s\}$; $C' = C \setminus \{a, b\}$;
EndIf
$G' = (V, E \setminus \{ab\} \setminus F^-)$;

Theorem 18 *Given a split graph $G = (V, E)$ together with its 3-partition and an edge $ab \in E$, a minimal split deletion G' of $G^-_{ab} = (V, E \setminus \{a, b\})$ and the 3-partition of G' can be computed in time $O(\min\{d_G(a), d_G(b)\})$ if G^-_{ab} is not split, or $O(1)$ otherwise.*

Proof. We can check whether the graph is split after the deletion in constant time by Lemma 13. If so, it means that ab cannot belong to C and $G' = G^-_{ab}$, so that its 3-partition can be computed in constant time. Notice, in fact, that when removing an edge ab from G only the endpoints must be moved unless: either the split partition of G is unique while the one of G_x is not (meaning that the degree of the endpoint of ab belonging to C is $\omega(G)$); or $\omega(G') = \omega(G) - 1$. In the first case we need to move all vertices of S with degree $\omega(G) - 1$ to Q', and since $Q = \emptyset$, by Lemma 14 we can do it in constant time. In the second case we know that Q was a clique and either $a, b \in Q$ or ab was between C and Q. When removing an edge from Q, if $|Q| = 2$, we need to move all vertices of S with degree $\omega(G) - 2$ to Q', otherwise, if $|Q| > 2$ we need to move all vertices in $Q \setminus \{a, b\}$ to C'. In both cases we can perform the update in constant time by Lemma 14. In the last case, when $a \in Q$ and $b \in C$ (or vice versa), we need to move $Q \setminus \{a\}$ to C', again in constant time by Lemma 14.

If G^-_{ab} is not split, then we apply Algorithm Minimal-edge-deletion, and delete a minimal set of edges incident to a (everything still holds by symmetry if considering b) in order to get G'. In this case both a and b belong to C, so after the deletion they are not adjacent. Besides, the clique size decreases always by one unless the partition of G is unique and there is at least a vertex of degree $\omega(G) - 1$ in $S \setminus N(a)$.

This means that b always remains in C' or it goes into Q' only if its only neighbor outside C is the unique vertex of degree $\omega(G)-1$ in $S \setminus N(a)$, so that Q' must be a clique. In every case a can belong only to S', hence all edges to its neighbors in S, and possibly to $|Q|-1$ neighbors in Q if Q is an independent set, must be removed. All these cases are covered by the algorithm. Given the cases, it is easy to check that the new split partition is correct and that the split deletion is minimal by Theorem 10 since all vertices that were incident to a deleted edge end up in S'.

For the running time analysis we need to prove that every case of Algorithm Minimal-edge-deletion can be executed in $O(d_G(a))$ time. Remember that $a, b \in C$ so they are adjacent to all vertices of $C \cup Q$. To find F^-, and update the degree of the vertices involved in the deletion, takes $O(d_G(a))$ time, because we always delete edges to vertices in the neighborhood of a in G.

In case 1a and 2a, we can move Q respectively to C' or S', in $O(1)$ time, by Lemma 14. Notice that in cases 3a, 3b and 3c, F^- is the same, so it does not depend on the **If** condition. Hence, in $O(d_G(a))$ time, we can find F^-, remove these edges from G, and update the degree lists of the sets of the 3-partition accordingly, without moving any vertex from a set to another. Now the current sets Q, S and C are not a correct 3-partition of G', but to make them right we need to move only W, a and possibly b, to other sets. However, F^- is exactly the set of edges from a to S, and since we removed them and updated all degree lists, now there cannot be vertices with degree $\omega(G)-1$ in S that were neighbors of a. Hence W is exactly the degree list $L_{\omega(G)-1}$ of the current S. This means that we can apply Lemma 14 to all cases 3a, 3b, 3c and 3d, updating the 3-partition in constant time, after we removed F^-.

Since we proved that each case can be executed in $O(d_G(a))$ time and we can choose in constant time the endpoint of ab with minimum degree, the theorem follows. ∎

4.2.3 Vertex deletion

Removing a vertex $x \in V$ from a split graph $G = (V, E)$, always keeps the graph split. Therefore in this section we are only interested in showing that the running time of the corresponding update can be performed in time $O(d_G(v))$. In order to achieve this running time we show that there exists an order of the edges incident to x such that, if these edges are deleted in that order, the resulting graph remains split after each edge removal. We call such an order a *good edge deletion order*. Such result is interesting on its own and allows us to remove a vertex just running $d_G(x)$ times the edge removal algorithm, thus getting the desired running time.

Lemma 19 *Given a split graph $G = (V, E)$, its 3-partition $V = S + C + Q$, and a vertex $x \in Q \cup S$, any order of the edges incident to x is a good edge deletion order.*

Proof. We know that there exists a split partition $V = I + K$ of G such that x is in I (there is always at least one vertex of Q that can be moved to I, so we can always choose x). Since all edges incident to x are then between the independent set and the clique, they can be removed in any order, and the graph will be split after each removal. ∎

Lemma 20 *Given a split graph $G = (V, E)$, its 3-partition $V = S + C + Q$, and a vertex $x \in C$, the following is a good edge deletion order of the edges incident to x: First remove edges between x and its neighbors in S, then remove edges between x and its neighbors in Q, and finally remove edges between x and its neighbors in C.*

Proof. Let $v_0, v_1, ..., v_k$ be the neighbors of x in G, listed according to the edge deletion order described in the lemma. Let $G_0 = G$, and G_i be the result of deleting edge xv_{i-1} from G_{i-1}, and let $V = S_i + C_i + Q_i$ be the 3-partition of G_i, for $1 \leq i \leq k+1$. We have to show that when we delete edge xv_i from split graph G_i, the resulting graph G_{i+1} is split for each $i \in \{0, 1, ..., k\}$. By Lemma 13, this is equivalent to showing that x and v_i do not both belong to C_i.

Let us assume for the sake of contradiction that at some step i, both x and v_i belong to C_i, and let i be the first such step. In this case, we show that v_i belongs to $S \cup Q$. Since all vertices of $S \cup Q$ are

ordered before all vertices of C among the endpoints of edges incident to x, if x has any neighbors in G_i belonging to $S \cup Q$, then $v_i \in S \cup Q$. Assume on the contrary that x has no neighbors belonging to $S \cup Q$ in G_i. Let $j < i$ be the first step where v_j belongs to C. Observe that the degree of each vertex of C remains unchanged until step j, and the size of the maximum clique can only have decreased. Thus every vertex of C belongs to C_j. Since in G_j, x has no neighbors outside of C_j, x does not belong to C_j. By Lemma 19, we can remove each remaining edge incident to x, contradicting the assumption that x and v_i both belong to C_i at some later step i. Thus we can conclude that $v_i \in S \cup Q$. It follows that $d_G(v_i) \leq \omega(G) - 1$, but since now v_i belongs to C_i, it means also that $d_{G_i}(v_i) \geq \omega(G_i)$ and $\omega(G_i) < \omega(G)$. On the other hand, the size of the maximum clique cannot decrease more than one since we remove edges incident only to one vertex, so $\omega(G_i) \geq \omega(G) - 1$. We can then conclude that $\omega(G_i) = \omega(G) - 1$ and therefore $d_G(v_i) = \omega(G) - 1$, so either $v_i \in S$ and $Q = \emptyset$, or $v_i \in Q$. In the first case, before step i, we have removed only edges incident to vertices of S, meaning that the size of the maximum clique cannot have decreased, so we have a contradiction. In the latter case, we also consider that since $x \in C_i$, it must have a neighbor y outside C_i. However $d_{G_i}(y) = d_G(y) < d_{G_i}(v_i) = d_G(v_i) = \omega(G) - 1$, meaning that $y \in S$ and we should have deleted the edge incident to y before the one incident to v_i, so we have a contradiction again. ∎

Theorem 21 *Given a split graph $G = (V, E)$ together with its 3-partition and a vertex $x \in V$, the 3-partition of $G[V \setminus \{x\}]$ can be computed in time $O(d_G(x))$.*

Proof. We first compute a good edge deletion order of $N_G(x)$ in time $O(d_G(x))$. It is enough to create a list (or an array of size $d(x)$) and scan the adjacency list of x at most three times, adding to the list at each iteration the neighbors of x in one of the three sets S, Q, and C in this order. In section 4.2.2 we showed that it takes constant time to remove one edge from a split graph if the removal keeps the graph split, and since we run such algorithm $d_G(x)$ times, the theorem follows. ∎

4.2.4 Edge addition

In this section we cover the last operation, namely adding an edge ab to a split graph G. We call the resulting graph G_{ab}. Using the results of the previous section we can easily show that computing a minimal split completion G' of G_{ab} and its 3-partition can be done in time $O(\max\{d_{G'}(a), d_{G'}(b)\})$ when we are given the 3-partition of G. Also, if G_{ab} is still split, then $G' = G_{ab}$ and its 3-partition can be computed in constant time.

Theorem 22 *Given a split graph $G = (V, E)$ together with its 3-partition, and a pair of vertices $a, b \in V$ such that $ab \notin E$, a minimal split completion G' of $G_{ab} = (V, E \cup \{ab\})$ and the 3-partition of G' can be computed in time $O(\max\{d_{G'}(a), d_{G'}(b)\})$ when G_{ab} is not split, or $O(1)$ time otherwise.*

Proof. We can check in constant time whether G_{ab} is split or not by Lemma 12. If so, also the update can be done in constant time. If the added edge does not increase the size of the maximum clique, then the only affected vertices are a and b, unless the split partition becomes unique, meaning that we added the edge between S and Q and Q is an independent set of size at least one. Assume $b \in Q$, then we have to move $Q \setminus \{b\}$ to S' and b to C'. We can do this in constant time by Lemma 14. If adding ab increases the size of the maximum clique, it means that we added the edge between two vertices of Q (that is therefore an independent set) or between a vertex of S of degree $\omega(G) - 1$ and C. In the first case we have to situations: $Q = 2$, so after the addition we might need to move all vertices of C that have degree $\omega(G)$ to Q'; or $|Q| > 2$, so we need to move $Q \setminus \{a, b\}$ to S'. For both we can apply Lemma 14 and perform the update in constant time. In the latter case we know that G has a unique split partition, hence $Q = \emptyset$. After adding the edge we need to move the endpoint that was in S and all vertices in C with degree $\omega(G)$ to Q'. Again, by Lemma 14, also this update can be done in constant time.

Let us consider the case when G_{ab} is not a split graph. Then we can simply remove an endpoint of ab from the graph, let us say a, in time $O(d_G(a))$ by Theorem 21, and put it back with the new edge. At

this point we can use Theorem 16 to obtain a minimal split completion G' of G_{ab} in time $d_{G'}(a)$. Notice that since in this case both a and b belong to S, their degree is at most $\omega(G)-1$. Also, the one that we decide to complete, namely a, will have to belong to C', so it will have a degree at least $\omega(G)$ in G'. Hence $d_{G'}(a) \geq d_{G'}(b)$ and the Lemma follows. ∎

5 Minimum split completions and deletions

In this section we will show that it is possible to compute, in linear time, a minimum split completion of a split graph plus one vertex, when adding such vertex does not keep the graph split. We give also a similar result for minimum split completions of split graphs plus one edge and minimal split deletion of split graph minus one edge.

Lemma 23 *Let $G=(V,E)$ be a split graph with 3-partition $V=S+C+Q$, $x \notin V$ and $N_x \subset V$ such that $G_x=(V,\ E \cup \{xy \mid y \in N_x\})$ is not split. If there exists a minimum split completion of G_x with some fill edges incident to x, then there exists a minimum completion of G_x where all the fill edges are incident to x.*

Proof. Let G' be a minimum split completion of G_x, with 3-partition $V \cup \{x\} = S'+C'+Q'$. Let us assume that there are some fill edges incident to x. Then x must belong to C' by Theorem 8. Let us assume that there is at least another fill edge yz in C' not incident to x. Then we can show that $Q'=\emptyset$. If Q' is an independent set of size at least 2, then there exists a C_4 $\{y,z,q_1,q_2\}$ in G, where $q_1,q_2 \in Q$. So Q is a clique and both y and z must have a neighbor in S' by Lemma 5. If they have two distinct neighbors, there is a $2K_2$ in G, and if they have more than one common neighbor, there is a C_4. Hence they have exactly one common neighbor w and $Q=\emptyset$ or there would be a C_4 in G. Also, for one of y or z, let us say z, w must the only neighbor in S'. Assume now that w is not universal for $C' \setminus \{x\}$. Then there is a vertex $v \in C' \setminus \{x\}$ not incident to w, with at least a neighbor $s \in S'$. We know that vz cannot be a fill edge or $\{s,v,w,z\}$ induces a $2K_2$ in G. This forces yv to be a real edge as well, or there would be a C_5 induced by $\{s,y,w,z,v\}$ if s is incident to y, or a $2K_2$ induced by $\{w,y,s,v\}$ otherwise. However if both vz and yv are not fill edges, $\{w,z,v,y\}$ induces a C_4 in G. We can conclude that w must be universal for $C' \setminus x$. At this point we can add the fill edge xw, switch z with w, so that $C'=C' \setminus \{z\} \cup \{w\}$ and $S'=S' \setminus \{w\} \cup \{z\}$, and remove yz, getting a split completion with at most as many fill edges as before. In particular all the fill edges must now be incident to x. Assume this is not true, then we should be able to argue again that there exists another vertex in S' universal for $C' \setminus \{x\}$, but this is a contradiction. No vertex in $S' \setminus \{z\}$ is incident to w and z is not incident to y. ∎

Also from Lemma 17, we know that all minimal completions where all fill edges are incident only to x, are equivalent, so we can just consider any one of them.

Theorem 24 *Given a split graph $G=(V,E)$ with 3-partition $V=S+C+Q$, a vertex $x \notin V$, a set $N_x \subseteq V$, and the knowledge that $G_x=(V,\ E \cup \{xy \mid y \in N_x\})$ is not split, a minimum set of fill edges that added to G_x makes it split again, is the smallest between the following two possible sets of fill edges F_1 and F_2:*

1. *If $N_x \cap S \neq \emptyset$, Q is a clique and $|N_x \cap Q| \leq |Q|-1$, let $q \in Q \setminus N_x$, then:*
   - $F_1 = \{xv \mid v \in (C \setminus N_x) \cup Q \setminus \{q\}\}$
   - $F_2 = \{sv \mid s \in N_x \cap S \wedge v \in (C \cup Q \setminus \{q\} \cup (N_x \cap S)) \wedge sv \notin E\}$

   For the remaining cases let $F_1 = \{xv \mid v \in C \wedge xv \notin E\}$, $X = \{v \mid v \in C \setminus N_x \wedge N(v) \cap S \subseteq N_x \cap S\}$, and c an element of X of minimum degree.

2. *$N_x \cap S \neq \emptyset$, Q is a clique and $N_x \cap Q = Q$:*
   - $F_2 = \{sv \mid s \in N_x \cap S \wedge v \in (C \setminus \{c\} \cup Q \cup (N_x \cap S)) \wedge sv \notin E\}$

3. $N_x \cap S \neq \emptyset$ and Q is an independent set:

- *If $Q \setminus N_x = \emptyset$:*
  $F_2 = \{sv \mid s \in N_x \cap (S \cup Q) \wedge v \in C \setminus \{c\} \cup (N_x \cap (S \cup Q)) \wedge sv \notin E\}$
- *Else:*
  $F_2 = \{sv \mid s \in N_x \cap (S \cup Q) \wedge v \in C \cup (N_x \cap (S \cup Q)) \wedge sv \notin E\}$

Proof. By Lemma 23, we know that we can consider only two types of completions: one where all fill edges are incident to x and one where all fill edges are in G. For the first kind, thanks to Observation 17, we can use the minimal completions defined in Section 4 obtaining the sets F_1 in the statement. For the second kind we will show a minimum way to modify the 3-partition of G into a new partition $V = C' + S' + Q'$, so that, when adding x, one of the conditions of Theorem 11 is satisfied. In particular, if $|N_x| \leq \omega(G) - 1$ we have to try and satisfy either condition 1 or 2 of Theorem 11, because adding edges to G can only increase the maximum clique size. Notice also that, since we give a minimum set of edges to add to G, when adding x to G' we will automatically get a minimal completion of G_x.

First of all we prove that if $|N_x| > \omega(G) - 1$, it is always possible to choose safely the set F_1 as the minimum. In this case, in order to satisfy condition 1 or 2 of Theorem 11 adding edges to G, a necessary condition is that all neighbors of x belongs to $C' \cup Q'$ and they are in the same clique. Let us assume that $|F_1| > 1$ or we can always choose F_1. This means that if Q is a clique, then $N_x \cap S \geq 2$, so we have to make $N_x \cap S$ into a clique and add at least two edges between $N_x \cap S$ and $C \cup Q$. If Q is an independent set, then $|N_x \cap (S \cup Q)| \geq 3$ and we have to make $N_x \cap (S \cup Q)$ into a clique. However in the first case $|F_1| \leq |(C \cup Q) \setminus N_x| = \omega(G) - (|N_x| - |N_x \cap S|) \leq |N_x \cap S| \leq (|N_x \cap S|^2 - |N_x \cap S|)/2 + 2$ and in the latter $|F_1| = |C \setminus N_x| = \omega(G) - 1 - (|N_x| - |N_x \cap (S \cup Q)|) < |N_x \cap (S \cup Q)| \leq (|N_x \cap (S \cup Q)|^2 - |N_x \cap (S \cup Q)|)/2$. Therefore we can always consider F_1 to be the smallest set. In order to satisfy condition 3 or 4 of Theorem 11 adding edges to G, instead, necessary conditions are that $N_x \cap C' = C'$ and also $|N_x \cap Q'| \geq |Q'| - 1$ if Q' is a clique. Assume x is not adjacent to at least 2 adjacent vertices in $u, v \in C \cup Q$, or we have $|F_1| = 1$. This means that in the new 3-partition: neither u nor v can belong to C', otherwise $N_x \cap C' \neq C'$; they cannot both belong to S' since they are connected, and they cannot both belong to Q' because then Q' would be a clique and $N_x \cap Q' \leq |Q'| - 2$ violating condition 3. Hence exactly one of them must be in Q' and the other one in S'. However this leads to a contradiction because there cannot be edges between S' and Q'. For the rest of the proof we can therefore consider only the case when $|N_x| \leq \omega(G) - 1$, so that we always have the necessary condition that N_x is a clique in G' and $N_x \subset C' \cup Q'$.

Case 1: $N_x \cap S \neq \emptyset$, Q is a clique and $N_x \cap Q \leq |Q| - 1$. In this case the completion F_1 corresponds to case 1a of Algorithm Minimal-vertex-completion. Let us see how to add a minimum set of fill edges F_2 to G in order to get a new graph G' with 3-partition $V = C' + S' + Q'$. If Q' is a clique, at most one vertex from $(C \cup Q) \setminus N_x$ might not belong to the clique $C' \cup Q'$, otherwise, if Q' is an independent set, at most one vertex from $(C \cup Q) \setminus N_x$ might not belong to the clique C'. Hence a necessary condition is to make $((C \cup Q) \setminus \{q\}) \cup (N_x \cap S)$ into a clique where $q \in (C \cup Q) \setminus N_x$. We can always choose q to be a vertex of $Q \setminus N_x$, that exists by assumption, minimizing the number of fill edges used. Notice that doing this, the necessary condition is also sufficient. All the neighbors of x are in $C' \cup Q'$, in particular either Q' is a clique and $Q' = N_x \cap S$ or Q' is an independent set with $Q' = (N_x \cap S) \cup \{q\}$, where $|N_x \cap S| = 1$. We can summarize this case as follows: $F_2 = \{sv \mid s \in N_x \cap S \wedge v \in (C \cup Q \setminus \{q\} \cup (N_x \cap S)) \wedge sv \notin E\}$.

Case 2: $N_x \cap S \neq \emptyset$, Q is clique (or empty) and $N_x \cap Q = Q$. In this case, F_1 corresponds to case 1b of Algorithm Minimal-vertex-completion. For F_2 we proceed as in the previous point, but now we must have $Q \subset C' \cup Q'$. Hence, if there is a vertex in $C \cup Q$ that might not belong to $C' \cup Q'$ if Q' is a clique or to C' if Q' is an independent set, it must come from $C \setminus N_x$. Also this vertex should not have neighbors in $S \setminus N_x$. Among the ones satisfying these properties, we can choose the one with the smallest degree, in order to minimize the fill edges we need to add. Let us call such vertex c, if it exists, and define the set F_2 as: $F_2 = \{sv \mid s \in N_x \cap S \wedge v \in (C \setminus \{c\} \cup Q \cup (N_x \cap S)) \wedge sv \notin E\}$.

Case 3: $N_x \cap S \neq \emptyset$ and Q is an independent set. In this case, F_1 corresponds to case 2 of Algorithm Minimal-vertex-completion. For F_2, we know that $N_x \cap (S \cup Q)$ must be made into a clique and we have

similar situations as in Case 1. If Q' is a clique, then $|(C' \cup Q') \cap C| \geq |C| - 1$, otherwise $|C' \cap C| \geq |C| - 1$. However if $Q \setminus N_x \neq \emptyset$, then we must have $C \subset (C' \cup Q')$ and $C \subset C'$ respectively. Notice, in fact, that if there exists $q \in Q \setminus N_x$, it would be adjacent to all C. Thus, if we choose a vertex $c \in C$ not to be in $C' \cup Q'$ when Q' is a clique or not to be in C when Q' is an independent set, we must have $Q \setminus N_x \subset C' \cup Q'$ or $Q \setminus N_x \subset C'$. However it is always at least as convenient to make c adjacent to exactly $S \cap N_x$, so that it belongs to $C' \cup Q'$ or C' respectively. In this case $F_2 = \{sv \mid s \in N_x \cap (S \cup Q) \wedge v \in C \cup (N_x \cap (S \cup Q)) \wedge sv \notin E\}$. When $Q \setminus N_x = \emptyset$, instead, as in the previous Case we can pick a vertex $c \in C \setminus N_x$ with neighbors only in $N_x \cap S$ and smallest degree, and make $(N_x \cap (S \cup Q)) \cup C \setminus \{c\}$ into a clique, getting $F_2 = \{sv \mid s \in N_x \cap (S \cup Q) \wedge v \in C \setminus \{c\} \cup (N_x \cap (S \cup Q)) \wedge sv \notin E\}$.

■

Theorem 25 *Given a split graph $G = (V, E)$, a vertex $x \notin V$, and a set $N_x \subseteq V$, a minimum split completion H of $G_x = (V,\ E \cup \{xy \mid y \in N_x\})$ can be computed in $O(|V| + |E|)$ time.*

Proof. What we need to compute is: The 3-partition of G; The minimum set of fill edges; The graph H. We will show that each of them can be computed in time at most $O(|V| + |E|)$.

The 3-partition of G, that can be computed in $O(|V| + |E|)$ time as described in Section 4.1. Then, finding out which is the smallest set between F_1 and F_2, can be also done in linear time since it requires only to compute the intersections between N_x and the sets of the 3-partition of G. For each vertex v incident to a fill edge it does not take more than $d_H(v)$ to find out to which vertices it must be made adjacent. Notice now that $|F_{min}| \leq |F_1| \leq |V|$, hence H has at most $|E| + |V|$ edges, meaning that $\sum_{v \in V} d_H(v) = O(|V(H)| + |E(H)|) = O(|V| + 1 + |E| + |V|) = O(|V| + |E|)$. ■

Finding the minimum split completion of a split graph plus one edge is a special case of the vertex addition.

Theorem 26 *Given a split graph $G = (V, E)$ together with its 3-partition $V = S + C + Q$, and a pair of vertices $a, b \in V$ such that $ab \notin E$, a minimum split completion G' of $G_{ab} = (V, E \cup \{ab\})$ can be computed choosing to minimally complete the endpoint of ab with highest degree.*

Proof. Let us define $G_a^- = G[V \setminus \{a\}]$, with 3-partition $V \setminus \{a\} = S_a + C_a + Q_a$. Then the proof follows by applying Theorem 24 to G_a^- when adding back a with adjacencies $N_a = N_G(a) \cup \{b\}$. Notice, in fact, that in G_a^-, either $N_a \cap S_a = \{b\}$ or $N_a \cap S_a = \emptyset$ because $b \in Q_a$. In the first case we apply either case 1,2 or 3 of Theorem 24, and since $N_a \cap S_a = \{b\}$, choosing between F_1 and F_2 reduces to choosing the vertex of maximum degree between a and b. In the latter case, a vertex of $s \in S$ can move to Q_a after the removal of a, only if: G had a unique partition, a had exactly one neighbor $c \in C$ of degree $\omega(G)$ and $d_G(s) = \omega(G) - 1$. In this situation $Q_a = \{c\} \cup W$, where $W = \{s \in S \mid d_G(s) = \omega(G) - 1\}$. This means that if $b \in Q_a$, all previous conditions applied and $b \in W$. Also $S_a \cap N_a = \emptyset$, Q_a is an independent set and $Q_a \cap N_a = \{b, c\}$, so condition 4 of Theorem 24 applies, and again the choice is reduced to choosing the vertex of maximum degree between a and b. ■

From Theorem 26 and Theorem 22 we get the following.

Theorem 27 *Given a split graph $G = (V, E)$ together with its 3-partition $V = S + C + Q$, and a pair of vertices $a, b \in V$ such that $ab \notin E$, a minimum split completion G' of $G_{ab} = (V, E \cup \{ab\})$ can be computed in $O(\max\{d_{G'}(a), d_{G'}(b)\})$ time.*

A minimal split deletion of a split graph minus one edge can be seen as a minimal split completion of the complement graph plus the edge, so the following result is straightforward.

Theorem 28 *Given a split graph $G = (V, E)$ together with its 3-partition, and an edge $ab \in E$, a minimum split deletion (split subgraph with the maximum number of edges) G' of $G_{ab}^- = (V, E \setminus \{ab\})$ can be computed in time $O(\min\{d_G(a), d_G(b)\})$.*

Proof. By symmetry with Theorem 26, it can be proven that the minimum deletion can be obtained choosing the endpoint with lowest degree of the edge that has been deleted. Hence the theorem follows from Theorem 18. ■

6 Concluding remarks

In this paper we showed how to implement efficiently some dynamic operations to maintain a graph split. These operations can be used to design many useful incremental algorithms on split graphs, that can run in the same time as the corresponding static algorithms. In particular, if the user does not want to repair the graph when a modification does not keep it split, we automatically get a fully dynamic algorithm for split graphs with optimal running time for each operation (therefore matching the algorithm given in [24]). For example, using just the vertex addition operation, we can check whether a graph is split in time $O(n+m)$, and furthermore, if it is not, we can easily give a forbidden subgraph as a certificate within the same time, just implementing the proof of Theorem 11. This give a simple linear time certification algorithm for split graphs. At the same time, if the graph is not split, we can minimally repair it at each vertex addition, producing a minimal split completion of the input graph in time $O(n+m+|F|)$. The minimality of each step is guaranteed by Algorithm Minimal-Vertex-Addition, while the minimality for the output graph follows from a property of hereditary graph families that is well discussed in [18]. This property guarantees that a minimal completion into an hereditary graph family with the universal vertex property (adding a universal vertex to a graph in the family, produce a graph still in the family) can be obtained in an incremental way. The only requirement is to make sure that when adding a new vertex to the graph, the current completion is minimal and all fill edges are incident to the new vertex. Being split graph self-complementary, it is not very difficult to design a similar algorithm for minimal split deletion using the again the vertex incremental approach.

In addition, let us we define the graph classes split$+kv$, split$+ke$ and split$-ke$ to be the classes containing the graphs that can be obtained from a split graph respectively adding k vertices, adding k edges and removing k edges. For $k=1$ we can give a certification algorithm and an algorithm to compute the minimum split completion or each of these classes. Both algorithms run in linear time in the size of the input. This kind of graph classes are often referred to as parametrized graph classes and algorithms for recognizing them or solving problem on them are well studied topics in the literature (see for example [8, 26, 32, 33]). In particular there is a recognition algorithm for parametrized split graphs that runs in polynomial time [8], but neither a linear nor a certifying one. To obtain the certifying algorithm, we use the incremental certifying algorithm for split graphs. We explain in detail the algorithm for split$+1v$. We start checking whether the graph is split adding a vertex at the time, and when we find a forbidden subgraph, we branch on each of its vertices, running again the split certifying algorithm on the graph minus one of them. If for at least one of such vertices the graph minus this vertex is split, we can answer yes and return such vertex. Otherwise we answer no and we give as certificate the first forbidden subgraph we found and all the others we found after the branching. Since the largest forbidden subgraph is a C_5, we need to run the linear time certifying algorithm for split graph at most 6 times, hence giving a linear running time. For split$+1e$ and split$-1e$, we use exactly the same technique, but we branch on the edges of the forbidden subgraph, rather than the vertices. Once we can recognize whether a graph is either split$+1v$, split$+1e$ or split$-1e$ and we know which vertex or edge we need to add/remove to make the graph split again, the result for minimum split completions follows by Theorems 25, 27 and 28.

References

[1] G. Ausiello, G.F. Italiano, A. Marchetti-Spaccamela, and U. Nanni. Incremental Algorithms for Minimal Length Paths. *Proc. ACM-SIAM Symp. on Discrete Algorithms*, 12–21, 1990.

[2] A. Brandstädt and V.B. Le. Split-Perfect Graphs: Characterizations and Algorithmic Use. *SIAM J. Discrete Math.*, 17-3:341–360, 2004.

[3] A. Brandstädt and V.B. Le and J.P. Spinrad. Graph Classes: a Survey. *SIAM Monographs on Discrete Mathematics and Applications*, 1999.

[4] Z. Blázsik, M. Hujter, A. Pluhár and Z. Tuza. Graphs with no induced C_4 and $2K_2$. *Discrete Mathematics*, 115:51–55, 1993.

[5] G.A. Cheston. Incremental algorithms in graph theory. *PhD Thesis*, 1976.

[6] A. Gyárfás and J. Lehel On a Helly type problem in trees. *Combinatorial Theory and Its Application, Colloquia Math*, 4:571–584, 1969.

[7] A. Berry, P. Heggernes, and Y. Villanger. A vertex incremental approach for maintaining chordality. *Discrete Mathematics.*, 306-3:318–336, 2006.

[8] L. Cai. Fixed-Parameter Tractability of Graph Modification Problems for Hereditary Properties. *Inf. Process. Lett.*, 58(4):171–176, 1996.

[9] C. Capelle, A. Cournier and M. Habib. Cograph recognition algorithm revisited and online induced P_4 search. Rapport de recherche 94073, LIRMM Universite Montpellier , 1994.

[10] R.M. McConnell. An $O(n^2)$ incremental algorithm for modular decomposition of graphs and two-structures. *Algorithmica*, 14:209–227, 1995.

[11] D. G. Corneil and Y. Perl and L. K. Stewart. A Linear Recognition Algorithm for Cographs. *SIAM Journal on Computing*, 14(4):926–934,1985.

[12] C. Crespelle and C. Paul. Fully dynamic algorithm for recognition and modular decomposition of permutation graphs. *Proc. of WG 2005, Springer Lecture Notes in Computer Science*, 3787:38–48, 2005.

[13] C. Demetrescu and G. Italiano. A new approach to dynamic all pairs shortest paths. *Proc. of STOC 2003*, 159–166, 2003.

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

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

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

[17] P. Heggernes and F. Mancini, Minimal Split Completion of Graphs, *Proc. of LATIN 2006, Springer Lecture Notes in Computer Science*, 3887:592–604, 2006.

[18] P. Heggernes, F. Mancini, and C. Papadopoulos. Making arbitrary graphs transitively orientable: Minimal comparability completions. *Proc. of ISAAC 2006, Springer Lecture Notes in Computer Science*, 4317:419–428, 2006.

[19] P. Heggernes and D. Kratsch. Linear-time certifying algorithms for recognizing split graphs and related graph classes. Reports in Informatics 328, University of Bergen, Norway, June 2006.

[20] P. Heggernes, K. Suchan, I. Todinca, and Y. Villanger. Minimal interval completions. *Proc. of ESA 2005, Springer Lecture Notes in Computer Science*, 3669:403–414, 2005.

[21] P. Heggernes, C. Paul, J. A. Telle and Y. Villanger. Interval Completion is Fixed Parameter Tractable. *Proc. of STOC 2007*, 374–381, 2007.

[22] P. Hell, R. Shamir, and R. Sharan, A fully dynamic algorithm for recognizing and representing proper interval graphs. *SIAM J. Comput.*, 31:289-305, 2002.

[23] L. Ibarra, Fully dynamic algorithms for chordal graphs. *Proc. of SODA 1999*, 923-924, 1999.

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

[25] L. Ibarra. A fully dynamic algorithm for recognizing interval graphs using the clique-separator graph. Technical Report DCS-263-IR, University of Victoria, Canada, 2001.

[26] H. Kaplan, R. Shamir and R. E. Tarjan. Tractability of Parametrized Completion Problems on Chordal, Strongly Chordal, and Proper Interval Graphs. *SIAM J. Comput.*, 28(5):1906–1922, 1999.

[27] Norbert Korte and Rolf H. Möhring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM J. Comput.*, 18(1):68–81,1989.

[28] D. Kratsch, R. M. McConnell, K. Mehlhorn and J. P. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. *Proc. of SODA 2003*, 158–167, 2003.

[29] D. Lokshtanov, F. Mancini, and C. Papadopoulos. Computing and extracting minimal cograph completions in linear time. Reports in Informatics 352, University of Bergen, Norway, April 2007.

[30] F. Maffray and M. Preissmann. Linear Recognition of Pseudo-split Graphs. *Discrete Applied Mathematics*, 52-3:307–312, 1994.

[31] F. Maffray and M. Preissmann. Split-Neighborhood Graphs and the Strong Perfect Graph Conjecture. *J. Comb. Theory, Ser. B*, 63-2:294–309,1995.

[32] D. Marx. Chordal Deletion Is Fixed-Parameter Tractable. *Proc. of WG 2006, Springer Lecture Notes in Computer Science*, 4271:37–48.

[33] D. Marx and I. Schlotter. Obtaining a Planar Graph by Vertex Deletion. *Proc. of WG 2007, Springer Lecture Notes in Computer Science*, to appear.

[34] R. Merris. Graph Theory *Wiley Interscience*, New York, 2001.

[35] R. Merris. Split graphs. *Europ. J. Comb.*, 24:413–430, 2003.

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

[37] S.D. Nikolopoulos and L. Palios, Adding an Edge in a Cograph. *Proc. of WG 2005, Springer Lecture Notes in Computer Science*, 3787:214-226, 2005.

[38] R. Shamir and R. Sharan, A fully dynamic algorithm for modular decomposition and recognition of cographs. *Discrete Appl. Math.*, 136:329-340, 2004.

[39] Pavol Hell, Ron Shamir, and Roded Sharan. A fully dynamic algorithm for recognizing and representing proper interval graphs. *SIAM J. Computing* 31(1):289–305, 2001