

**REPORTS
IN
INFORMATICS**

ISSN 0333-3590

**Faster parameterized algorithms for
MINIMUM FILL-IN**

**Hans L. Bodlaender Pinar Heggernes
Yngve Villanger**

REPORT NO 375

August 2008



Department of Informatics
UNIVERSITY OF BERGEN
Bergen, Norway

This report has URL

<http://www.ii.uib.no/publikasjoner/textrap/pdf/2008-375.pdf>

Reports in Informatics from Department of Informatics, University of Bergen,
Norway, is available at <http://www.ii.uib.no/publikasjoner/textrap/>.

Requests for paper copies of this report can be sent to:

Department of Informatics, University of Bergen, Høyteknologisenteret,
P.O. Box 7800, N-5020 Bergen, Norway

Faster parameterized algorithms for MINIMUM FILL-IN

Hans L. Bodlaender* Pinar Heggernes† Yngve Villanger†

Abstract

We present two parameterized algorithms for the MINIMUM FILL-IN problem, also known as CHORDAL COMPLETION: given an arbitrary graph G and integer k , can we add at most k edges to G to obtain a chordal graph? Our first algorithm has running time $\mathcal{O}(k^2nm + 3.0793^k)$, and requires polynomial space. This improves the base of the exponential part of the best known parameterized algorithm time for this problem so far. We are able to improve this running time even further, at the cost of more space. Our second algorithm has running time $\mathcal{O}(k^2nm + 2.35965^k)$ and requires $\mathcal{O}^*(1.7549^k)$ space.

1 Introduction

The MINIMUM FILL-IN problem asks, given as input an arbitrary graph G and an integer k , whether a chordal graph can be obtained by adding at most k new edges to G . A chordal graph is a graph without induced cycles of length at least four. This is one of the most extensively studied problems in graph algorithms, as it has many practical applications in various areas of computer science. The problem initiated from the field of sparse matrix computations, where the result of Gaussian Elimination corresponds to a chordal graph, and minimizing the number of edges in a chordal completion is equivalent to minimizing the number of non-zero elements in Gaussian Elimination [19]. Among other application areas are data-base management systems [20], knowledge-based systems [16], and computer vision [6]. Since the problem was proved NP-complete [23], it has been attacked using various algorithmic techniques, and there exist polynomial-time approximation algorithms [17], exponential-time exact algorithms [9, 10], and parameterized algorithms [14, 5]. The current best bounds are $\mathcal{O}^*(1.7549)$ time and space for an exact algorithm [10], and $\mathcal{O}((m+n)4^k/(k+1))$ time for a parameterized algorithm [5], where the \mathcal{O}^* -notation suppresses factors polynomial in n .

In this paper we contribute with new parameterized algorithms for the solution of the MINIMUM FILL-IN problem. The field of parameterized algorithms,

*Department of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, the Netherlands. Email:hansb@cs.uu.nl

†Department of Informatics, University of Bergen, PB 7803, 5020 Bergen, Norway. Email:{pinar.heggernes,yngve.villanger}@ii.uib.no. Supported by the Research Council of Norway.

first formalized by Downey and Fellows [7], has been growing steadily and attracting more and more attention recently [8, 18]. Informally, a *parameterized algorithm* computes an exact solution of the problem at hand, but the exponential part of the running time is limited to a (hopefully small) parameter, typically an integer. For the MINIMUM FILL-IN problem, the natural parameter is the number of added edges. The number of vertices and edges of G are denoted by n and m , respectively. The first parameterized algorithms for this problem were given by Kaplan et al. and appeared more than a decade ago [14, 15], with running times $\mathcal{O}(16^k m)$ and $\mathcal{O}(k^2 nm + k^6 16^k)$. A refined analysis of these algorithms by Cai gave the current best parameterized running time of $\mathcal{O}((m+n)4^k/(k+1))$ [5].

We present two algorithms that improve on the basis of the exponential part of the running time of these parameterized algorithms. Central in our algorithms is a new result, describing edges that can always be added when computing a minimum solution. Based on this result, our first algorithm is intuitive and easy to understand, and requires $\mathcal{O}(k^2 nm + 3.0793^k)$ time and polynomial space. We are able to improve the base of the exponential part even further in a second algorithm, at the cost of more space. Our second algorithm, which is more involved, requires $\mathcal{O}(k^2 nm + 2.35965^k)$ time and $\mathcal{O}^*(1.7549^k)$ space.

In this extended abstract, we use well-known insights on chordal graphs and triangulations without further explanations. For overviews, the reader can consult e.g., [12, 13].

2 Preliminaries

All graphs in this work are undirected and simple. A graph is denoted by $G = (V, E)$, with vertex set V and edge set $E(G) = E$. For a vertex subset $S \subseteq V$, the *subgraph of G induced by S* is $G[S] = (S, \{\{v, w\} \in E \mid v, w \in S\})$. The *neighborhood* of S in G is $N_G(S) = \{v \in (V \setminus S) \mid \exists w \in S : \{v, w\} \in E\}$. We write $N_G(v) = N_G(\{v\})$ for a single vertex v , and $N_G[S] = N_G(S) \cup S$. Subscripts are omitted when not necessary. A vertex v is *universal* if $N(v) = V \setminus \{v\}$.

A vertex subset $S \subseteq V$ is a *separator* in G if $G[V \setminus S]$ has more than one connected component. A connected component C of $G[V \setminus S]$ is called *full* for S if $N(C) = S$. Vertex set S is a *minimal u, v -separator* for G if u and v are in different connected components of $G[V \setminus S]$ and S is an inclusion minimal vertex set separating u and v . The separator S is a *minimal separator* of G if there exist u and v in V such that S is a minimal u, v -separator.

A pair of vertices $\{u, v\}$ is a *non-edge* if u and v are not adjacent. For a vertex set S , we let $F(S)$ denote the set of non-edges in $G[S]$. S is a *clique* if $F(S) = \emptyset$ or $|S| = 1$. A clique is a *maximal clique* in G if it is not a proper subset of another clique in G . A set of vertices that is a clique and a separator is called a *clique separator*. A vertex v is *simplicial* if $N(v)$ is a clique.

A graph is *chordal*, or *triangulated*, if it does not contain a cycle with four or more vertices as an induced subgraph. A graph $H = (V, F)$ is a *triangulation*

or *chordal completion* of a graph $G = (V, E)$ if $E \subseteq F$ and H is chordal. The edges in $F \setminus E$ are called *fill edges*. H is a *minimal triangulation* of G if there is no triangulation $H' = (V, F')$ of G where F' is a proper subset of F . A triangulation with the minimum number of edges is called a *minimum triangulation*. Every minimum triangulation is thus minimal.

A set of vertices $S \subseteq V$ is a *potential maximal clique* in G if there is a minimal triangulation H of G where S is a maximal clique in H . Potential maximal cliques play an essential role in several algorithms computing minimum triangulations and related problems, like treewidth. We use an algorithm one of [10] in our second algorithm.

A vertex set $U \subset V$ is a *moplex* if $G[U]$ is a clique, $N[v] = N[u]$ for any pair of vertices in U , and $N(U)$ is a minimal separator in G . Moplex U is *simplicial* if $G[N(U)]$ is a clique.

Proposition 1 (Folklore) *Let $G = (V, E)$ be an induced cycle, let v_1, v_2, \dots, v_n be the order of the vertices on the cycle, and let $H = (V, F)$ be a minimal triangulation of G , where $\{v_1, v_3\} \notin F$. Then there exists an edge $\{v_2, v\} \in F$ such that $v \in V \setminus \{v_1, v_2, v_3\}$.*

Proposition 2 (Folklore) *A set S of vertices in a graph G is a minimal u, v -separator if and only if u and v are in different full components associated to S . In particular, S is a minimal separator if and only if there are at least two distinct full components associated to S .*

Before we start describing our algorithms, we present an important new result that describes fill edges that can be safely added when computing a minimum triangulation, independent of k . This is the first result of its kind to our knowledge, and it is crucial for our further results.

Lemma 3 *Given a graph $G = (V, E)$, let S be a minimal separator of G such that $|F(S)| = 1$ and $S \subseteq N(u)$ for a vertex $u \in V$. Then there exists a minimum triangulation of G that has the single element of $F(S)$ as a fill edge.*

Proof. Let $H = (V, F)$, $E \subset F$ be a minimum triangulation of G , and let $F(S) = \{\{x, y\}\}$. If $\{x, y\} \in F$, then there is nothing to prove, so assume that $\{x, y\} \notin F$. Let T be a clique tree of H , and let X and Y be the closest pair of bags in T such that $x \in X$ and $y \in Y$. Notice that $S \cup \{u\} \setminus \{x, y\}$ is a subset of every minimal x, y -separator, and by Helly property every vertex in $S \cup \{u\} \setminus \{x, y\}$ appears in every bag on the unique path from X to Y in T .

Let C_0, C_1, \dots, C_r be the connected components of $G[V \setminus S]$, let $u \in C_0$, and let C_0, C_1, \dots, C_p for $p \leq r$ be the connected components whose neighborhoods contain both x and y . Clearly $H[C_i \cup S \cup \{u\}]$ is chordal for $i \in \{1, \dots, r\}$, since any induced subgraph of a chordal graph is chordal.

We will now construct a tree decomposition T' of G where the maximal bags will be the maximal cliques of a triangulation $H' = (V, F')$ of G , such that $\{x, y\} \in F'$ and $|F'| \leq |F|$. The first step is to make a bag $\mathcal{S} = S$. The expense of adding the edge $\{x, y\}$ will be compensated at a later point in the proof.

For $i \in \{p+1, \dots, r\}$ take a clique tree T_i of $H[N_G[C_i]]$ and add an edge from a maximal clique of T_i containing $N_G(C_i)$ to \mathcal{S} . Notice that $N_G(C_i) \subset S$, and $N_G(C_i)$ does not contain both x and y and thus $H[N_G(C_i)]$ is a clique.

For $i \in \{1, \dots, p\}$, let T_i be a clique tree of $H[N_G[C_i] \cup \{u\}]$, and let X_i, Y_i be the closest pairs of maximal cliques in T_i containing x and y , respectively. Notice that u is contained in every maximal clique on the unique path from X_i to Y_i in T_i , and that u has a fill edge to every vertex that appears in one of these maximal cliques, since S separates C_i from C_0 which contains u . Obtain the new tree T'_i by removing u from Y_i , and replacing u with y in any other maximal clique of T_i . The number of fill edges will not increase, since all edges to u from C_i were fill edges, and these are now incident to y instead. The edge $\{x, y\}$ is also added, but as mentioned this will be compensated for later. Now add an edge from X_i in T'_i to \mathcal{S} .

By Proposition 2 the minimal separator S has at least two full components, and thus $p \geq 1$. When producing T'_1 the vertex u was replaced by y in all maximal cliques of T_1 which contained u . In $H[N_G[C_1]]$ there exists an induced path $P_1 = x, v_1, v_2, \dots, v_q, y$. Each vertex v_i for $i \in \{1, \dots, q\}$ is contained in some minimal separator of $H[N_G[C_1] \cup \{u\}]$, and is thus also contained in a maximal clique between X_1 and Y_1 [4, 11, 22]. Thus, it follows that $\{v_1, \dots, v_q\} \subset N_H(u) \cap C_1$. We have now saved one edge since $\{u, v_q\}$ is a fill edge removed during the creation of T' and $\{v_q, y\}$ is already an edge of $H[N_G[C_1]]$. This edge is used to pay for the added edge $\{x, y\}$.

It remains to find a triangulation of $H[N_G[C_0]]$. By [2] a chordal graph H_0 is obtained by adding edge $\{x, y\}$ and an edge from y to every vertex contained in a minimal x, y -separator of $H[N_G[C_0]]$. Let T_0 be a clique tree of H_0 , and let the final tree T' be obtained by adding a edge from X_0 in T_0 to \mathcal{S} . For an added $\{y, u'\}$ edge where u' is contained in some minimal x, y separator of $H[N_G[C_0]]$ there exists an $\{u', v_i\}$ edge that is removed in H' . Let $P_0 = x, u_1, u_2, \dots, u_t$ be an induced path in $H[N_G[C_0]]$, such that $u_j = u'$. This path exists, since u' is contained in a minimal x, y -separator of $H[N_G[C_0]]$. Consider again the clique tree T of H , and let X and Y be the closest pair of bags in T that contains x and y . Every maximal clique on the unique path X, Z_1, Z_2, \dots, Z_l from X to Y in T , contains a vertex v_i where $i \in \{1, \dots, p\}$ and every vertex u_1, \dots, u_t is contained in some Z_i where $i \in \{1, \dots, l\}$. Thus, there exists an fill edge from u' to at least one vertex in $\{v_1, \dots, v_q\}$ which is removed when obtaining the new tree T' .

In total we have not added more edges than we have removed. For the final justification let us argue that T' is a tree decomposition of G . Every vertex is either contained in S or one of the connected components of $G[V \setminus S]$ and is thus contained in one bag of T' . For an edge $\{a, b\} \in E(G)$ we have the same, it is either between vertices in S or between vertices in $G[N[C]]$ for some connected component C of $G[V \setminus S]$. Finally, all bags containing a vertex z induces a connected subtree of T' , since this holds for every T'_i and for T' by the fact that a vertex is either completely in T'_i or the fact that every T'_i is directly attached to \mathcal{S} . ■

We start now the description of our algorithms, each of which will be pre-

sented in its own section. For both of our algorithms, the input is an undirected graph $G = (V, E)$ and an integer k ; and each algorithm outputs either a minimum size set of fill edges of size at most k , or NO if each triangulation of G requires at least $k + 1$ edges.

3 An $\mathcal{O}^*(3.0793^k)$ -time algorithm for MINIMUM FILL-IN

The first algorithm that we present uses polynomial time reductions and some branching rules. In the subproblems generated by this branching algorithm, some vertices have a *marking*. As will be clear when the subproblems are analyzed, sometimes we will have the choice of adding a set of fill edges or concluding with a set of vertices that each must be incident to a fill edge. These vertices will be marked, to give the desired restriction in the solution of resulting subproblems. More precisely, subproblems can be associated with problem instances of the form (G, k, r, M) with $G = (V, E)$ a graph, k and r integers, and $M \subseteq V$ a set of *marked* vertices. For such an instance, we ask whether there exists a triangulation $H = (V, F)$ of G with $|F \setminus E| \leq k$ and $2|F \setminus E| - |M| \leq r$, such that each vertex in M is incident to a fill edge. We say that a vertex v is *marked* if $v \in M$, and r denotes the number of marks we still can place at later steps during the algorithm. From the original problem where G and k are given, the new initial problem instance is $(G, k, 2k, \emptyset)$. Any triangulation of G which requires k fill edges is also a solution to the new instance since $r = 2k$ and $M = \emptyset$.

Lemma 4 *If $(G = (V, E), k, r, M)$ has a solution with $\{v, w\}$ as a fill edge, then $((V, E \cup \{\{v, w\}\}), k - 1, r - \gamma, M \setminus \{v, w\})$ has a solution, where $\gamma \in \{0, 1, 2\}$ is the number of unmarked endpoints of $\{v, w\}$.*

At several points during our algorithm, we write: *add an edge e to F and update accordingly*. Following Lemma 4, the update consists of decreasing k by one, decreasing r by the number of unmarked endpoints of e , and removing the marks of marked endpoints of e . If we add more edges, we do this iteratively. Whenever we mark a vertex, we decrease r by one. Note that if two edges are added with a common endpoint, this endpoint is unmarked after the first addition, and thus causes a decrease of r at the second addition.

The algorithm is based on checking the existence of the structures described in the following paragraphs, and performing the corresponding actions. When a change is made to the input, we start again by checking trivial cases.

Trivial cases. First, the algorithm tests whether G is chordal and $k \geq 0$ and $r \geq 0$. If so, it returns \emptyset . Next, it tests if $k \leq 0$ or $r \leq -1$. If so, it returns NO.

4-Cycles. Then, the algorithm branches on induced cycles of length four (*4-cycles*). Suppose that v, w, x, y induce a 4-cycle. Then, in any triangulation, $\{v, x\}$ is an edge or $\{w, y\}$ is an edge. The algorithm recursively solves the two subcases: one where we add $\{v, x\}$ as a fill edge and update accordingly, and one where we add $\{w, y\}$ as a fill edge and update accordingly.

An invariant of the algorithm is that each 4-cycle has at least two adjacent vertices that are not marked. Initially, this holds as all vertices are unmarked. Whenever we create a 4-cycle by adding an edge, we unmark the endpoints of the added edge. Marks are only added in graphs that do not have a 4-cycle.

Note that we create in this case two subproblems. In each, k is decreased by one, and r is decreased by at least one. We will show by induction that the search tree formed by the algorithm has at most $a^k \cdot b^r$ leaves for inputs where we can use k fill edges, and place at most r marks. Thus, this case gives as condition $a^k \cdot b^r \geq 2 \cdot a^{k-1} \cdot b^{r-1}$, i.e., $ab \geq 2$.

Mplexes with marked and unmarked vertices. We use several times the following insight, following from well-known results on mplexes and triangulations (see e.g., [1]).

Lemma 5 *Let U be a mplex. There is a minimum triangulation that has a fill edge incident to each vertex in U , or there is a minimum triangulation where $N(U)$ is a clique and no fill edge is incident to any vertex in U .*

Thus, when we have a mplex that contains marked and unmarked vertices, we mark all vertices of the mplex.

Finding mplexes with unmarked vertices. Then, the algorithm tests whether there is a mplex U that contains no marked vertices. If there is no such mplex, the algorithm returns NO. Safeness of this step comes from the following lemma, which follows from the work on mplexes by Berry et al. [1], by simply considering the first mplex in a mplex elimination ordering of H .

Lemma 6 *Let $G = (V, E)$ and let $H = (V, F)$ be a minimum triangulation of G . There is a mplex U such that $F \setminus E$ has no incident edge to U , and $N(U)$ is a clique in H .*

We take such a mplex U , and let $S = N(U)$. We compute $F(S)$, i.e., the set of non-edges in the neighborhood of U .

Simplicial vertices. If $F(S) = \emptyset$, then all vertices in U are simplicial. We recurse on the instance $(G \setminus U, k, r, M)$.

By well-known theory on chordal graphs this instance is equivalent to, and a minimum set of fill edges for the new instance is also a minimum set of fill edges for, the original instance.

Mplexes missing one edge. Next, we test if $|F(S)| = 1$. By Lemma 3 it is always *safe* to add the edge in $F(S)$, but in some cases we need to compensate for this by removing one mark from a vertex. The proof of the following Lemma relies heavily on Lemma 3.

Lemma 7 *Let $G = (V, E)$ be a graph and let $M \subseteq V$ be the set of marked vertices in G . Suppose there exists a minimum triangulation of G such that each vertex in M is incident to a fill edge. Let $u \in V$ be an unmarked vertex, and let $S \subseteq N(u)$ be a minimal separator $F(S) = \{\{x, y\}\}$.*

1. *If there is a unique vertex v^* , such that v^* is the last vertex on each induced path from x to y through a full component of S not containing u , then there is a minimum triangulation of G that contains the edge $\{x, y\}$ and such that each vertex in $M \setminus \{v^*\}$ is incident to a fill edge.*
2. *If there is no unique vertex v^* , such that v^* is the last vertex on each induced path from x to y through a full component of S not containing u , then there is a minimum triangulation of G that contains the edge $\{x, y\}$ and such that each vertex in M is incident to a fill edge.*

Proof. Let $H = (V, F)$, $E \subset F$ be a minimum triangulation of G , such that every vertex in M has at least one incident edge in $F \setminus E$. If $\{x, y\} \in F$, then there is nothing to prove, so let us assume that $\{x, y\} \notin F \setminus E$. We will now analyze the triangulation $H' = (V, F')$ constructed in the proof of Lemma 3. If all vertices in M has an incident edge in $F' \setminus E$, then the proof of Lemma 3 can be applied directly.

Let us start by listing the set of vertices that might loose their incident fill edges. For the connected component C_i for $i \in \{1, \dots, p\}$ fill edges incident to u in $H[N_G[C_i] \cup \{u\}]$ are moved to be incident to y in stead. This is unproblematic for any vertex not adjacent to y in G , but for neighbours of y , this means that they are loosing an incident fill edge. We used this saved edge in the proof of Lemma 3 to pay for the new and added edge $\{x, y\}$ and thus there exists exactly one such edge, vertex, and component since H is an minimum solution. Let $\{v_1, \dots, v_q\}$ be an induced path from x to y in $H[N_G[C_1] \cup \{x, y\}]$. It follows that v_q is the single vertex that have lost one incident fill edge. In the case of two different first vertex on an induced path from x to y through C_1 would be a contradiction to H , since two edges can be saved which is a contradiction to the minimality of H . Taking $v^* = v_q$ shows Lemma 7. ■

Suppose we have a moplex U , with $F(N(U))$ consisting of the single edge $\{x, y\}$. The condition of Lemma 7 now becomes: there is a vertex v^* that is the last vertex on each induced path in $G[V \setminus U]$ from x to y . A simple modification of standard breadth first search allows us to find v^* if existing in linear time. If v^* exists, we remove its marking. Then, in both cases, we add the edge $\{x, y\}$ and update accordingly.

In this case, we possibly decrease k by one, and increase r by one. This gives us as condition on the running time constants: $a \geq b$.

The condition is not symmetric. We can apply the condition with roles of x and y switched and save a marking in some cases.

Branching on moplexes. If none of the earlier tests succeeds, we arrive at the last branching, performed on a moplex U with all vertices in U unmarked.

Recall Lemma 5. It dictates which two subproblems we consider. In the first subproblem, we mark all vertices in U . In the second subproblem, we add all edges in $F(N(U))$ and update accordingly. In the first, r is decreased by $|U|$. In the second, $|F(N(U))| \geq 2$, as otherwise there is no pair of edges with a common endpoint in $F(N(U))$, so k is decreased by two. Note that there must be a vertex that is common to two elements of $F(N(U))$: if not, then suppose

$\{x, y\}$ and $\{v, w\}$ are elements in $F(N(U))$, but no other combination of x , y , v , and w is an element of $F(N(U))$. Then, these four vertices form a 4-cycle, which is a contradiction. Thus, in the second subproblem, r is also decreased by at least one.

This gives us as condition for the running time analysis: $a^k \cdot b^r \geq a^{k-2} \cdot b^{r-1} + a^k \cdot b^{r-1}$, or $a^2 b \geq 1 + a^2$.

Each of these subproblems is solved recursively, and from these solutions, we then return the best one, adding $F(N(U))$ to the set returned by the second subproblem except when it returned NO.

Analyzing the running time. By standard graph algorithmic tools, each recursive call can be performed in $\mathcal{O}(nm)$ time, except that the checking for all 4-cycles before any other operation is done costs once $\mathcal{O}(m^2)$ time. We now analyze the number of recursive calls in the search tree. We start with an instance with $r = 2k$, so the running time of the algorithm is bounded by $a^k \cdot b^{2k}$. Each of the steps gave a condition on a and b , and we get as minimum $ab^2 = 3.0793$ when we set $a = 1.73205$ and $b = 1.33334$. Thus, the total running time becomes $\mathcal{O}(m^2 + nm \cdot 3.0793^k)$.

By results of [15] and [17] it is possible to reduce a given instance $(G = (V, E), k)$ of MINIMUM FILL-IN to an equivalent instance $(G' = (V', E'), k')$ where $k' \leq k$ and $|V'| = \mathcal{O}(k^2)$, in $\mathcal{O}(k^2 nm)$ time. By preprocessing the input by such an algorithm we get an additive time cost of $\mathcal{O}(k^2 nm)$ but the size of n and m have been reduced to respectively $\mathcal{O}(k^2)$ and $\mathcal{O}(k^4)$. Thus, the time complexity for our algorithm becomes $\mathcal{O}(k^2 nm + 3.0793^k)$.

Theorem 8 *The MINIMUM FILL-IN problem can be solved in $\mathcal{O}(k^2 nm + 3.0793^k)$ time, using polynomial space.*

4 An $\mathcal{O}^*(2.35965^k)$ -time algorithm for MINIMUM FILL-IN

In this section, we give a second algorithm for the MINIMUM FILL-IN problem. This algorithm uses less time as a function of k , at the cost of exponential space as a function of k . Like the previous algorithm, we create subinstances with some vertices marked and with an additional parameter r , which is the number of marks that still can be handed out.

An important difference from the previous algorithm is that the mark is a vertex set containing the vertices which are candidates to add a fill edge incident to. The algorithm involves a more extensive analysis of subproblems, a mixing of eliminating moplexes with partitioning the graph on clique separators, resolution of cycles with four vertices, and a resolution of certain cases with the exact algorithm, recently given by Fomin and Villanger [10]. In order to properly execute the steps where we partition on clique separators, marks are *annotated*. We also allow that the algorithm returns solutions that do not respect marks. When there is a solution respecting marks with α fill edges, the algorithm may return any solution with at most α fill edges. If the algorithm

returns NO, we know there is no solution that respects marks. (This is needed for the last step, where we forget the marks.)

With our algorithm description, we will also make the first steps towards the time analysis. We derive a number of conditions on a function $T(k, r)$, such that the running time of all recursive calls that originate at a node with parameters k (number of fill edges) and r (number of marks that still can be placed) is bounded by $T(k, r)$ times a function, polynomial in n , not depending on k . As the time for non-leaf nodes of the search tree is bounded by a polynomial in n times the time for leaf nodes, we only count the time at leaf nodes. We want to show that $T(k, r) \leq a^k \cdot b^r \cdot o(k)$ and derive some conditions on a and b .

The algorithm consists of carefully handling subproblems of various types. We describe in the next paragraphs which conditions are tested, in what order, and what steps are executed if a certain condition holds. First we will present some polynomial-time reduction rules. Several cases are similar to or the same as in our previous algorithm.

Trivial cases. If G is chordal and $k \geq 0$ and $r \geq 0$, then we return the empty set. If G is not chordal, and $k \leq 0$ or $r \leq -1$, we return NO.

Universal vertex. If G contains a universal vertex then we simply remove this vertex. This is safe, since no induced cycle of length at least 4 contains such a vertex.

Simplicial vertices. If an unmarked vertex is simplicial then we remove the vertex, and obtain an equivalent instance. If a marked vertex is simplicial then we return NO, since no minimal triangulation will add fill edges incident to a simplicial vertex.

Clique separators. Then, the algorithm tests if there is a clique separator. If there is a clique separator S , then let V_1, \dots, V_r be the vertex sets of the connected components of $G[V \setminus S]$. We create now r subinstances, with graphs $G[S \cup V_1], \dots, G[S \cup V_r]$.

Vertices in subinstances in $V \setminus S$ keep their marks. A marked vertex in S is marked in only one subinstance, containing the annotated vertex set related to the mark. We will describe this more in detail when the annotated vertices are defined.

These subproblems are now independent. First, we test for each subproblem if there is a solution with at most two fill edges. If so, we solve this in polynomial time and use this to reduce the parameter of the remaining problems.

When we have α subproblems each of whose solutions requires at least three fill edges, each can add at most $k - 3\alpha + 3$ fill edges. Thus, we need to choose a and b such that $T(k, r) \leq \alpha \cdot T(k - 3\alpha + 3, r)$ for all α , for all $\alpha > 1$. This gives $a^{3\alpha-3} \geq \alpha$, which holds for every integer $\alpha > 1$ and $a \geq 1.3$.

A minimal separator missing one edge. The next step is similar to the steps in our previous algorithm that uses Lemma 7, but now we apply it also

to vertices that do not belong to a moplex.

We test if there is an unmarked vertex v and a minimal separator S contained in $N(v)$, such that $F(S)$ contains only one edge. If we have such a minimal separator S , we add the edges in $F(S)$, test if vertex v^* described in Lemma 7 exists, remove the mark of v^* , update accordingly, and solve recursively the remaining instance.

If this instance returns NO, we return NO, otherwise we return the union of $F(S)$ and the solution found by the instance. This again gives as condition for the running time analysis: $a \geq b$.

If none of the above reduction steps applies, we consider the following branching steps.

4-Cycles. Like in the previous algorithm, we now test if there is an induced 4-cycle, and branch on the two ways of adding an edge between non-adjacent vertices in the cycle. Again, we get as condition $ab \geq 2$.

Minimal separator S with $|F(S)| \geq 3$. Test if there is an unmarked vertex v , and a minimal separator $S \subseteq N(v)$ with $|F(S)| \geq 3$. If so, we branch on this vertex, similarly as in the previous algorithm: we create two subinstances and recurse on these, and then output the smallest fill set of these instances, treating NO as a solution of size ∞ .

In one subinstance, we add all fill edges in $F(S)$, and k is decreased by $|F(S)|$. For each unmarked vertex incident to an edge of $F(S)$, r is decreased by one. For each vertex incident to $j > 1$ edges in $F(S)$, r is decreased by $j - 1$. We also remove all marks from vertices incident to edges in $F(S)$.

In the other subinstance, we mark vertex v , but we also have to define the annotation for the mark of v . Let W be a connected component of $G[V \setminus N[v]]$ not containing v such that $N(W) = S$. The connected component W exists by definition of S . The annotated vertices for the mark of v will be W . Let us justify this.

Since S is not completed into a clique there is an edge $\{x, y\} \in F(S)$ which is not used as a fill edge in the optimal solution we are searching for. Vertex set W is a full component of S , and thus there exists an induced path u_1, u_2, \dots, u_r from x to y only containing vertices in W . The vertex set $\{y, v, x, u_1, u_2, \dots, u_r\}$ induces a cycle in G . By Proposition 1, v has a fill edge to one of the vertices in $\{u_1, u_2, \dots, u_r\}$ if $\{x, y\}$ is not a fill edge. Since we do not know which one of the edges in $F(S)$ is not added when v is marked, we use W as the annotation and in this way ensure that the correct vertex is in the set.

Lemma 9 *Given a graph G , let $S \subset V$ be a clique separator, let $v \in S$ be a marked vertex, and let X be the annotation of v . Then there exists a connected component W of $G[V \setminus S]$ such that $X \subseteq W$.*

Proof. Note first that $S \cap X = \emptyset$, since no vertex of X is adjacent to v when v is marked. Secondly, the mark of v is removed when fill edge $\{x, v\}$, where $x \in X$, is added. Finally, S does not separate any pair of vertices in X , since S contains no vertices in X , and $G[X]$ is connected. ■

Again, we return the smallest solution found by the two subinstances, treating NO as a solution of size ∞ .

Similar arguments as before show correctness of this step. In a minimum triangulation, we must either add all edges in $F(S)$ or vertex v will be incident to a fill edge. This gives: $T(k, r) \leq T(k-3, r-2) + T(k, r-1)$, leading to the condition $a^3b^2 \geq 1 + a^3b$. In the first subinstance r is reduced by two since there are no induced 4-cycles, and thus the three edges of $F(S)$ induces a connected component. One consequence of this is that at most 4 vertices will be incident to the three edges in $F(S)$.

Notice that, for any remaining minimal separator S contained in the neighborhood of an unmarked vertex, $|F(S)| = 2$.

Split the problem into two non-chordal subproblems. Let v be an unmarked vertex, let S be a minimal separator in $N(v)$, and let G' be the resulting graph where the edges $F(S)$ are added to G . We test if there are two connected components W_1 and W_2 of $G[V \setminus S]$ where $G'[N[W_1]]$ and $G'[N[W_2]]$ are non-chordal. We will then know that at least one fill edge will be required for each of the connected components W_1 and W_2 in the case where S is completed into a clique. The algorithm proceeds as follows: Check if one of the subproblems $G'[N[W_1]]$ or $G'[N[W_2]]$ can be triangulated by adding at most three fill edges. If this is the case, we get the recursive condition: $T(k, r) \leq T(k-3, r-2) + T(k, r-1)$, giving $a^3b^2 \geq 1 + a^3b$. If not, we have the subproblems $G'[N[W_1]]$ and $G'[V \setminus N[W_1]]$, and we get the recursive condition: $T(k, r) \leq 2T(k-5, r-1) + T(k, r-1)$, giving $a^5b \geq 2 + a^5$.

Using a list of potential maximal cliques. In this case, we use another algorithm to solve the MINIMUM FILL-IN problem. This algorithm is a variant of the exact algorithm for MINIMUM FILL-IN by Fomin and Villanger [10]. Suppose none of the above holds, then we can make several observations. Let S_v be a minimal separator contained in $N(v)$ for an unmarked vertex v , and let W_v be a connected component of $G[V \setminus N[v]]$ which is full for S_v . The graph obtained by adding the two edges in $F(S_v)$ to $G[N[W_v]]$ will not be chordal, since that would imply a vertex $w \in W_v$, where $S_v \subseteq N(w)$, and thus the endpoints of an edge of $F(S_v)$ and vertices v, w would induce a 4-cycle. Since all connected components of $G[V \setminus N[v]]$ generate non-chordal subproblems by the argument above, we can notice that $G[V \setminus N[v]]$ contains exactly one connected component, since zero components would imply that v is universal, and more than one would imply that the previous described rule could be applied. A consequence of this again is that for any unmarked vertex $N(v)$ only contains one minimal separator, which we can call S_v , and there is only one connected component W_v in $G[V \setminus N[v]]$ which is full for S_v . Notice also that $G[N[v]] = G[V \setminus W_v]$ is chordal when the two edges in $F(S_v)$ are added. Before starting to describe the rule, we need more knowledge about the problem instance.

Lemma 10 *Given a graph $G = (V, E)$ where none of the rules above can be applied, let u and v be unmarked vertices, where $S_u \subset N[v]$. Then u and v are*

contained in the same connected component W of $G[V \setminus S_v]$.

Proof. Both u and v have the endpoints of the edges in $F(S_u)$ in their neighborhood, and thus they are adjacent, since there are no induced 4-cycle. Vertex v is contained in W by definition, and by the edge $\{u, v\}$ vertex u is contained in $N[W]$. Let us on the contrary assume that $u \in S_v$. Then there exists a neighbor w of u in W_v . Vertex w is not contained in S_u , since S_u is completely contained in $N(v)$. Let S_w be the minimal separator in $N(w) \subset N[u]$. Separator S_w is not a clique in G by the previous rules, and $F(S_w) \subseteq F(S_u)$ since $G[N[u]]$ is chordal when adding the two edges $F(S_u)$. Vertices v, w and the endpoints of an edge in $F(S_w)$ induces now a 4-cycle which is a contradiction to the statement of the lemma. ■

Lemma 11 *Given a graph $G = (V, E)$ where none of the rules above can be applied, let u and v be unmarked vertices. Then $N[u] = N[v]$, or $F(S_u) \cap F(S_v) = \emptyset$.*

Proof. Let us first on the contrary assume that unmarked vertices u and v have minimal separators S_u and S_v such that $|F(S_u) \cap F(S_v)| = 1$, and let $\{x, y\}$ be the edge in $F(S_u) \cap F(S_v)$. Then edge $\{u, v\}$ is contained in E , since the graph would otherwise contain a 4-cycle.

There are now two subcases: either $v \in S_u$ (equivalently $u \in S_v$) or not. If v is not in S_u , then v is contained in the connected component C_u of $G[V \setminus S_u]$ that contains u , because of the edge $\{u, v\}$. By the previous rules we know that $G[N[C_u]]$ is chordal when the two edges in $F(S_u)$ are added. Since v does not have any neighbors outside of $N[C_u]$, and $|F(S_v)| = 2$, then $F(S_u) = F(S_v)$, which is a contradiction to the assumption. If $v \in S_u$, then $S_u \subset N[v]$, since an edge $\{v, z\} \in F(S_u)$ for $z \in \{x, y\}$ would make either x or y non adjacent to v . By Lemma 10 u and v are contained in the same connected component of $G[V \setminus S_v]$, and thus the arguments of the first case, where $u \notin S_v$, can be applied again.

At this point we know that $|F(S_u) \cap F(S_v)| \in \{0, 2\}$, and it remains to prove that $N[u] = N[v]$ if $|F(S_u) \cap F(S_v)| = 2$. Like in the first half of the proof we can notice that $\{u, v\} \in E$, since otherwise a 4-cycle exists, consisting of u, v and the endpoints of an edge in $F(S_v)$. Also there is no vertex w in $S_u \setminus N(v)$ (equivalently $S_v \setminus N(u)$) since this would create a 4-cycle with v, w (equivalently, u, w) and the endpoints of an edge of $F(S_u)$.

Finally we consider the case where there is a vertex $w \in (N(u) \setminus S_u) \setminus N(v)$. There are two cases again, either $v \in S_u$ or $v \notin S_u$. Let us first consider the case where v is not in S_u . Then u, v, w are in $G[N[u]]$, which becomes chordal after adding the edges $F(S_u)$. There exists a minimal separator S_w in $N[w] \subseteq N[u]$ that separates w and v . Notice that S_w becomes a clique after adding the edges $F(S_u)$ to $G[N[u]]$, since none of the edges in $F(S_u)$ are incident to w . This construction requires that v, w and the endpoints of an edge in $F(S_w)$ induce a 4-cycle, which is a contradiction.

Second case is when $v \in S_u$. Since $F(S_u) = F(S_v)$, then none of the edges in $F(S_u)$ or $F(S_v)$ are incident to u or v . As a result $S_u \subseteq N[v]$ if $v \in S_u$.

By Lemma 10 u is not contained in S_v , and the arguments for the case where $u \notin S_v$ can be applied. ■

Lemma 12 *Given a graph $G = (V, E)$ where none of the rules above can be applied, let v be an unmarked vertex. Then the number of unmarked vertices in S_v is at most 3.*

Proof. Let us on the contrary assume that 4 unmarked vertices w_1, w_2, w_3, w_4 are contained in S_v . By Lemma 11 $F(S_u) \cap F(S_v) = \emptyset$ for $u \in \{w_1, w_2, w_3, w_4\}$, since otherwise $N(u) = N(v)$ which would be a contradiction to the existence of $u \in S_v$. If $S_v \subset N[u]$ for $u \in \{w_1, w_2, w_3, w_4\}$, then by Lemma 10 $N[v] \subseteq N[u]$, and since $|F(S_v)| = 2$ and $G[N[u]]$ is triangulated by adding the two edges in $F(S_u)$, then $F(S_u) = F(S_v)$. This is not possible since by Lemma 11 $N(u) = N(v)$ in this case. We can now conclude that $S_v \not\subseteq N[u]$ for $u \in \{w_1, w_2, w_3, w_4\}$. A consequence of this is that $F(S_v)$ have edges incident to all four vertices $\{w_1, w_2, w_3, w_4\}$, which is a contradiction to the fact that $|F(S_v)| = 2$ and these two edges have a common endpoint. ■

Lemma 13 *Given a graph $G = (V, E)$ with no induced 4-cycle or clique separator, let $H = (V, F)$ be a triangulation of G with $E \subset F$. Then only one connected component of $(V, F \setminus E)$ contains edges.*

Proof. Notice that every minimal separator of H contains a fill edge, and thus every maximal clique of H also contains a fill edge. Since every minimal separator of H is contained in at least two maximal cliques of H [4, 11, 22], it is enough to show that the fill edges inside any maximal clique induce a connected graph. For fill edges that belong to different maximal cliques, there is a series of minimal separators between the two cliques (in any clique tree) containing fill edges, and hence if the fill edges of each maximal clique are connected, so are the fill edges of the whole graph. Let X be a maximal clique of H , and assume that $\{u, v\}$ and $\{x, y\}$ are fill edges in X . There are two cases; either one of the edges $\{x, u\}, \{u, y\}, \{y, v\}, \{v, x\}$ is a fill edge and the lemma holds, or $\{x, u\}, \{u, y\}, \{y, v\}, \{v, x\}$ are edges of G that induce a 4-cycle which is a contradiction. ■

A consequence of this lemma is that there are at most $k+1$ vertices incident to a fill edge, and hence we can return NO at this point if there are more than $k+1$ marked vertices. So assume there are at most $k+1$ marked vertices.

Next step is to control the unmarked vertices. For each unmarked vertex v we know that $|F(S_v)| = 2$, and by Lemma 11 that $F(S_v) = F(S_u)$, or $F(S_v) \cap F(S_u) = \emptyset$. Partition the unmarked vertices into groups, where $F(S_v) = F(S_u)$ for all pairs u, v of vertices in the same group. Consider the minimum triangulation H , which respects all given marks. By Lemma 13 there is at most $k+1$ vertices incident to fill edges of H . Since $N[u] = N[v]$ for a pair of unmarked vertices in the same group, we can notice that either all or none of them will have an incident fill edge in H . Remove the at most $k+1$ groups that have incident fill edges. The number of remaining groups is at most $k/2$,

since each group have two private fill edges in its neighborhood. In total there is at most $3k/2$ groups of unmarked vertices.

By Fomin et. al. [9] a minimum triangulation can be computed in $\mathcal{O}(|\Delta_G| + |\Pi_G|) \cdot n^3$ time, where Δ_G is the set of minimal separators of G , and Π_G is the set of potential maximal cliques of G . The algorithm of [9] is more powerful than that stated; among all the tree decompositions that can be constructed, using provided potential maximal cliques as bags and provided minimal separators as edges between the bags, the algorithm returns the tree decomposition that introduces the minimum number of fill edges when completing every bag in the decomposition into a clique. Thus, we only have to provide a minimal separator S if $|F(S)| \leq k$ and a potential maximal clique Ω if $|F(\Omega)| \leq k$ to the algorithm if we want to solve the MINIMUM FILL-IN problem with parameter k . Previously we partitioned the unmarked vertices into $3k/2$ groups, where all vertices in the same group have the same neighborhood. When listing minimal separators and potential maximal cliques, we can treat vertices with the same closed neighborhood as a single vertex. A vertex is for instance contained in a minimal u, v -separator S if and only if it has a neighbor in the connected components of $G[V \setminus S]$ which contains respectively u and v . For potential maximal cliques there exists a similar definition. A vertex is contained in the potential maximal clique, if and only if it can reach all other vertices in the potential maximal clique by a direct edge or a path consisting only of vertices that are not contained in the potential maximal clique. Given these arguments, no vertex set that contains only a part of a group of unmarked vertices are candidates to be minimal separators or potential maximal cliques. By Lemma 12 there exists at most three unmarked vertices that have adjacency to vertices in a group, without being contained in it. This means that a minimal separator or potential maximal clique can at most contain $4\sqrt{k}$ of the $3k/2$ groups of unmarked vertices.

Lemma 14 *Given a graph G , let M be the set of marked vertices, let q be the number of groups of unmarked vertices, and let $k = |M| - 1$. Then all minimal separators and potential maximal cliques containing at most ℓ groups of unmarked vertices can be listed in $\mathcal{O}\left(1.7549^{k+1} \cdot \sum_{i=0}^{\ell} \binom{q}{i}\right)$ time and space.*

Proof. To prove Lemma 14, we show that all minimal separators and potential maximal cliques containing $W \subset (V \setminus M)$ where $|W| \leq \ell$ can be listed in $\mathcal{O}(1.7549^{k+1})$ time and space. For the rest of this proof we assume W is the intersection between $V \setminus M$ and the minimal separator or potential maximal clique we are searching for.

Due to Proposition 2 every minimal separator S has two full components C_1 and C_2 . Without loss of generality let C_1 be the component such that $|C_1 \cap M| \leq |C_2 \cap M|$. Fomin and Villanger [10] give an algorithm that lists all minimal separators by searching for C_1 , starting from every vertex of G . Since the inclusion/exclusion of unmarked vertices is already decided, we only have to branch on marked vertices, and thus the algorithm of [10] can be adapted to list

all the minimal separators containing W and not $V \setminus (M \cup W)$ in $\mathcal{O}(1.6181^{k+1})$ time and space.

Listing potential maximal cliques is done in the same way, but is slightly more involved. The set of potential maximal cliques can be partitioned into two sets, *nice* potential maximal cliques and non nice potential maximal cliques. A potential maximal clique is defined as *nice* if it not an induced clique when all but one minimal separator completely contained in the potential maximal clique is completed into a clique. By [3, 9, 10] the set of non nice potential maximal cliques containing at most ℓ unmarked vertices can be generated from the set of minimal separators and nice potential maximal cliques contains at most ℓ unmarked vertices with a polynomial delay for each potential maximal clique. Our problem is then reduced to listing nice potential maximal cliques where the graph induced over these contains at most ℓ unmarked vertices.

One property for a nice potential maximal clique Ω is the existence of a triple $x, y, z \in \Omega$ and a partitioning C, D_x, D_y of $V \setminus \Omega$ such that $C \cup D_x \cup \{x\}$, $C \cup D_y \cup \{y\}$, and $D_x \cup D_y \cup \{z\}$ induces connected subgraphs, and $\Omega = N(C \cup D_x \cup \{x\}) \cup \{x\} = N(C \cup D_y \cup \{y\}) \cup \{y\} = N(D_x \cup D_y \cup \{z\}) \cup \{z\}$ [21]. Let us assume that $(C \cup D_x \cup \{x\}) \cap M \leq (C \cup D_y \cup \{y\}) \cap M \leq (D_x \cup D_y \cup \{z\}) \cap M$.

Again we use the algorithm in [10] and search for the connected vertex set $C \cup D_x \cup \{x\}$. In the same way as for minimal separators we do not have to branch on unmarked vertices ($V \setminus M$). As a result, the algorithm for listing potential maximal cliques in [10] can be adapted to list all nice potential maximal cliques containing the unmarked vertices W in $\mathcal{O}(1.7549^{k+1})$ time and space. Given the set of minimal separators and nice potential maximal cliques containing at most ℓ unmarked vertices the set of non nice potential maximal cliques containing at most ℓ unmarked vertices can be generated with a polynomial delay [10]. ■

We also observe that if $H = (V, F)$ is a triangulation of $G = (V, E)$ with at most k fill edges, and $V \setminus M$ contains the set of at most $3k/2$ group, then no clique in H contains more than $4\sqrt{k}$ of the $3k/2$ groups of vertices of $V \setminus M$. So, we can list all potential maximal cliques that can contribute to a triangulation of G with at most k fill edges using Lemma 14 with $\ell = 4\sqrt{k}$ in time $\mathcal{O}^*(1.7549^k)$. Given this list, we can employ the algorithm of Fomin et al. [9] and compute a triangulation with minimum fill of G ; this algorithm uses time, polynomial in n times the size of the list of potential maximal cliques.

This gives the condition: $T(k, r) \leq 1.7549^k$, and thus $a \geq 1.7549$.

Analyzing the running time. We derived a number of conditions on a and b , such that, if these hold, then by induction it follows that $T(k, r) \leq a^k b^k \cdot o(k)$. As we start with an instance with k and $r = 2k$, the running time is a polynomial in n times $T(k, 2k)$. We get as minimum $ab^2 = 2.35965$ when we set $a = 1.7549$ and $b = 1.15956$. Rounding this up allows to ignore the $o(k)$ term, and thus the algorithm requires $\mathcal{O}^*(2.35965^k)$ time. By the same arguments as the ones used for the polynomial part of the running time of our previous algorithm, we can conclude that this algorithm has running time $\mathcal{O}(k^2 nm + 2.35965^k)$, and requires $\mathcal{O}^*(1.7549^k)$ space.

Theorem 15 *The MINIMUM FILL-IN problem can be solved in $\mathcal{O}(k^2nm + 2.35965^k)$ time, using $\mathcal{O}^*(1.7549^k)$ space.*

5 Conclusions

In this extended abstract, we presented parameterized algorithms for the MINIMUM FILL-IN problem. The first algorithm is relatively simple and uses polynomial space; the second algorithm uses for one step exponential space, but less time as a function of k . We expect that the first algorithm is practical for small variants of k . Using some of the steps of the second algorithm in combination with the first algorithm probably gives speedup for many inputs.

Acknowledgments

We thank Guido Diepen, Igor Razgon, Thomas van Dijk, and Johan van Rooij for useful discussions.

References

- [1] A. BERRY AND J. BORDAT, *Moplex elimination orderings*, Electronic notes in Discrete Mathematics, 8 (2001), pp. 6–9.
- [2] A. BERRY, P. HEGGERNES, AND Y. VILLANGER, *A vertex incremental approach for maintaining chordality*, Discrete Mathematics, 306 (2006), pp. 318–336.
- [3] V. BOUCHITTÉ AND I. TODINCA, *Listing all potential maximal cliques of a graph.*, Theoret. Comput. Sci., 276 (2002), pp. 17–32.
- [4] P. BUNEMAN, *A characterization of rigid circuit graphs*, Discrete Math., 9 (1974), pp. 205–212.
- [5] L. CAI, *Fixed-parameter tractability of graph modification problems for hereditary properties*, Inf. Process. Lett., 58 (1996), pp. 171–176.
- [6] F. R. K. CHUNG AND D. MUMFORD, *Chordal completions of planar graphs*, J. Comb. Theory, 31 (1994), pp. 96–106.
- [7] R. G. DOWNEY AND M. R. FELLOWS, *Parameterized complexity*, Springer-Verlag, New York, 1999.
- [8] J. FLUM AND M. GROHE, *Parameterized Complexity Theory*, Springer-Verlag, New York, 2006.
- [9] F. V. FOMIN, D. KRATSCHE, AND I. TODINCA, *Exact (exponential) algorithms for treewidth and minimum fill-in*, in ICALP 2004, vol. 3142 of LNCS, Springer, 2004, pp. 568–580.

- [10] F. V. FOMIN AND Y. VILLANGER, *Treewidth computation and extremal combinatorics*, in ICALP, vol. 5125 of Lecture Notes in Computer Science, Springer, 2008, pp. 210–221.
- [11] F. GAVRIL, *The intersection graphs of subtrees in trees are exactly the chordal graphs*, J. Combin. Theory Ser. B, 16 (1974), pp. 47–56.
- [12] M. C. GOLUMBIC, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.
- [13] P. HEGGERNES, *Minimal triangulations of graphs: A survey*, Discrete Mathematics, 306 (2006), pp. 297–317.
- [14] H. KAPLAN, R. SHAMIR, AND R. E. TARJAN, *Tractability of parameterized completion problems on chordal and interval graphs: Minimum fill-in and physical mapping*, in IEEE Symposium on Foundations of Computer Science, 1994, pp. 780–791.
- [15] ———, *Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs*, SIAM J. Computing, 28 (1999), pp. 1906–1922.
- [16] S. L. LAURITZEN AND D. J. SPIEGELHALTER, *Local computations with probabilities on graphical structures and their applications to expert systems*, J. Royal Statist. Soc., ser B, 50 (1988), pp. 157–224.
- [17] A. NATANZON, R. SHAMIR, AND R. SHARAN, *A polynomial approximation algorithm for the minimum fill-in problem*, SIAM J. Computing, 30 (2000), pp. 1067–1079.
- [18] R. NIEDERMEIER, *Invitation to Fixed-Parameter Algorithms*, Oxford University Press, 2006.
- [19] D. J. ROSE, *Triangulated graphs and the elimination process*, J. Math. Anal. Appl., 32 (1970), pp. 597–609.
- [20] R. E. TARJAN AND M. YANNAKAKIS, *Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs*, SIAM J. Comput., 13 (1984), pp. 566–579.
- [21] Y. VILLANGER, *Improved exponential-time algorithms for treewidth and minimum fill-in.*, in LATIN 2006, vol. 3887 of Lecture Notes in Computer Science, Springer, 2006, pp. 800–811.
- [22] J. WALTER, *Representations of rigid cycle graphs*, PhD thesis, Wayne State University, USA, 1972.
- [23] M. YANNAKAKIS, *Computing the minimum fill-in is NP-complete*, SIAM J. Alg. Disc. Meth., 2 (1981), pp. 77–79.