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. To achieve these results, we present a new lemma describing the edges that can safely be added to achieve a chordal completion with the minimum number of edges, regardless of k.

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, called fill 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 [23]. Among other application areas are data-base management systems [24], knowledge-based systems [18], and computer vision [6]. Since the problem was proved NP-complete [27], it has been attacked using various algorithmic techniques, and there exist polynomial-time approximation algorithms [20], exponential-time exact algorithms [10, 11], and parameterized algorithms [16, 5]. The current best bounds are $\mathcal{O}^*(1.7549^n)$ time and space for an exact algorithm [11], and $\mathcal{O}((n+m)\,\frac{4^k}{k+1})$ time for a parameterized algorithm [5], where n and m denote the number of vertices and edges of G, respectively, and the $\mathcal{O}^*$-notation suppresses factors polynomial in n.

In this paper we contribute with new parameterized algorithms to the solution of the Minimum Fill-In problem. The field of parameterized algorithms, first formalized by Downey and Fellows [8], has been growing steadily and attracting more and more attention recently [9, 21]. Informally, a *parameterized algorithm* computes an exact solution of the problem at

*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.

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 k, the number of fill edges. The first parameterized algorithms for this problem were given by Kaplan et al. and appeared more than a decade ago [16, 17], with running times $\mathcal{O}(m\,16^k)$ and $\mathcal{O}(k^2nm + k^6 16^k)$. An algorithm similar to the $\mathcal{O}(m\,16^k)$ algorithm was obtained independently by Cai who also gave a refined analysis that yielded the current running time of $\mathcal{O}((n+m)\,\frac{4^k}{k+1})$ [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 $\mathcal{O}(k^2nm + 3.0793^k)$ 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 $\mathcal{O}(k^2nm + 2.35965^k)$ time and $\mathcal{O}^*(1.7549^k)$ space.

This paper is organized as follows. In Section 2, we give some preliminary definitions and review well known results on chordal graphs and related notions. In Section 3, we give an important and useful result, where we characterize edges that can be safely added without disturbing optimality of a solution to the MINIMUM FILL IN problem. Sections 4 and 5 give the descriptions and analyses of our algorithms, each of which will be presented in its own section: the algorithm that uses polynomial space is presented in Section 4, and Section 5 gives the algorithm that uses exponential space. For both of our algorithms, the input is an undirected graph $G = (V, E)$ and an integer k; each algorithm outputs either a minimum size set of at most k fill edges, or NO if each chordal completion of G requires at least $k+1$ fill edges. Finally, Section 6 gives some concluding remarks.

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. 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 at least two connected components. A connected component C of $G[V \setminus S]$ is said to be *associated to* S, and it is called a *full component* if $N(C) = S$. Vertex set S is a *u,v-separator* for G if u and v are in different connected components of $G[V \setminus S]$, and a *minimal u,v-separator* for G if no proper subset of S is a u,v-separator. Two separators S and T are said to be *crossing* if S is a u,v-separator for some two vertices $u, v \in T$, in which case T is an x,y-separator for two vertices x, y in S [22].

In general, S is a *minimal separator* of G if there exist u and v in V such that S is a minimal u,v-separator. If S is a u,v separator and u is not in a full component associated to S, then S is not a minimal u,v-separator, since $N(C) \subset S$ is also a u,v-separator, where C is the component of $G[V \setminus S]$ that contains u. Hence the proposition below follows immediately.

Proposition 1 (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.*

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 both a clique and a separator is called a *clique separator*. A vertex v is *simplicial* if $N(v)$ is a clique. 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, $N(U)$ is a minimal separator in G, and U is an inclusion maximal set where these conditions holds. Moplex U is *simplicial* if $G[N(U)]$ is a clique.

A *perfect elimination ordering (peo)* of a graph $G = (V,E)$ is an ordering of the vertices of G into $v_1, \ldots, v_n$, such that for each i, $1 \leq i \leq n$, the higher indexed neighbors of v_i form a clique, i.e., v_i is a simplicial vertex in $G[\{v_i, v_{i+1}, \ldots, v_n\}]$.

A *tree decomposition* of a graph $G = (V,E)$ is a tree whose nodes correspond to subsets of V, called *bags*, that satisfies the following: every vertex of V appears in a bag; for all $\{u,v\} \in E$, there is a bag where u and v appear together; for any vertex $u \in V$, the nodes of T corresponding to the bags that contain u induce a connected subtree of T. (We will simply use bags to denote both bags and nodes.) It follows that for two bags X and Y of a tree decomposition T, $X \cap Y$ is contained in every bag on the unique path between X and Y in T. A *clique tree* is a special kind of tree decomposition with a bijection between the nodes of the tree and the maximal cliques of G.

A *chord* in a cycle (path) is an edge that is between two non-consecutive vertices of the cycle (path). A graph is *chordal*, or *triangulated*, if every cycle on four or more vertices has a chord. 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 with $F' \subset F$. 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 (pmc)* in G if there is a minimal triangulation H of G where S is a maximal clique in H.

For details on chordal graphs and triangulations, the reader can consult e.g., [14, 15]. Here we give the minimum necessary background that is needed for our results and proofs.

Theorem 2 ([7, 12, 19, 4, 13, 26]) *Let $G = (V,E)$ be a graph. The following are equivalent.*

- *G is chordal.*
- *Every minimal separator of G is a clique.*
- *Every minimal separator contained in the neighborhood of a vertex of G is a clique.*
- *G has a peo.*
- *G has a clique tree.*

Hence only chordal graphs have clique trees, whereas all graphs have tree decompositions (a trivial tree decomposition is one with a single bag containing all vertices of the graph). A clique tree is a very useful structure since it contains all the information of the minimal separators of a chordal graph. If we view each edge of a clique tree as the intersection of its endpoint maximal cliques, every edge of a clique tree of G corresponds to a minimal separator of G, and every minimal separator of G appears as an edge in every clique tree of G [4]. It is also important to note that chordal graphs have at most n maximal cliques [7], and hence at most $n-1$ minimal separators. It is an easy observation that every tree decomposition of an arbitrary graph G corresponds to a triangulation H of G (not necessarily minimal) obtained by adding edges to G so that every bag of the tree decomposition becomes a clique. The given tree decomposition of G corresponds then to a clique tree of H.

The following characterization of minimal triangulations related to minimal separators is important for understanding our results.

Theorem 3 ([22]) *Given an arbitrary graph G, a chordal graph H is a minimal triangulation of G if and only if H is the result of making a maximal set of pairwise non-crossing minimal separators into cliques (by adding necessary edges to G to achieve this).*

From the above theorem it follows that if there is a clique separator S in $G = (V, E)$, no minimal triangulation contains a fill edge between two vertices separated by S. Hence the MINIMUM FILL-IN problem decomposes into subproblems $G[S \cup C]$ for each connected component C of $G[V \setminus S]$. The same is true for minimal separators that are completed into cliques while computing a minimal triangulation [22].

In addition, we will use several times the following characterization of minimal triangulations related to moplexes.

Theorem 4 ([1]) *Given an arbitrary graph G, a chordal graph H is a minimal triangulation of G if and only if H is the result of repeatedly choosing a moplex X and making the neighborhood of X into a clique by adding the missing edges, before deleting X.*

Finally, we end this section with the following easy but useful observation.

Proposition 5 (Folklore) *Let $v_1, v_2, v_3, v_4, \ldots, v_t$ be a chordless cycle in a graph $G = (V, E)$, and let $H = (V, F)$ be a minimal triangulation of G, where $\{v_1, v_3\} \notin F$. Then there exists a fill edge $\{v_2, v\} \in F$ for some $v \in \{v_4, \ldots v_t\}$.*

Proof. Since $v_3, v_4, \ldots, v_t, v_1$ is a path between v_1 and v_3 in G, there is also a path in H between v_1 and v_3 involving only (a subset of) vertices $v_3, v_4, \ldots, v_t, v_1$. Let $v_1, u_1, u_2, ..., u_r, v_3$ be a shortest path of this kind between v_1 and v_3 in $H[V \setminus \{v_2\}]$. Since v_1 and v_3 are not adjacent, we can conclude that $r \geq 1$. Since H is chordal and every induced subgraph of a chordal graph is also chordal, $H[\{v_1, u_1, \ldots, u_r, v_3, v_2\}]$ is chordal, has at least four vertices, and contains a cycle involving all its vertices. Since $H[\{v_1, u_1, \ldots, u_r, v_3\}]$ is a chordless path, $H[\{v_1, u_1, \ldots, u_r, v_3, v_2\}]$ contains fill edges and all fill edges are incident to v_2. ■

3 Edges that can be safely added

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 6 *Let S be a minimal separator of $G = (V, E)$ 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 the above mentioned properties of tree decompositions, every vertex in $S \cup \{u\} \setminus \{x, y\}$ appears in every bag on the unique path from X to Y in T.

Let $C_0, C_1, ..., C_r$ be the connected components of $G[V \setminus S]$, let $u \in C_0$ and let $C_0, C_1, ..., C_p$ 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 $i \in \{1, ..., r\}$ since any induced subgraph of a chordal graph is chordal.

We will now construct a tree decomposition T' of G where the the bags that are not subsets of other bags will be exactly 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 $i \in \{p + 1, ..., r\}$ 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. We do not make any changes in subgraphs of this type.

For $i \in \{1, ..., p\}$, let T_i be a clique tree of $H[N_G[C_i] \cup \{u\}]$, and let X_i, Y_i be the closest pair 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 1 the minimal separator S has at least two full components, and thus $p \geq 1$, and C_0 and C_1 are two full components. When producing T_1' the vertex u was replaced by y in all maximal cliques of T_1 which contained u. Let $P_1 = x, v_1, ..., v_q, y$ be a shortest path between x and y in $H[C_1 \cup \{x, y\}]$. Such a path must exist with $q \geq 1$ since x and y are not adjacent in H and this subgraph is connected. Each vertex v_i for $i \in \{1, ..., q\}$ is contained in some minimal x, y-separator of $H[N_G[C_1] \cup \{u\}]$, and is thus also contained in a maximal clique between X_1 and Y_1 by the previously mentioned properties of clique trees. Thus, it follows that $\{v_1, ..., v_q\} \subset N_H(u) \cap C_1$. 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 $H[N_G[C_0]]$ containing $\{x, y\}$ as a fill edge. In [2] it is shown that if we have a chordal graph that does not contain an edge between two vertices x and y, then the graph obtained by adding an edge between x and y, and adding edges from y to every vertex of every minimal x, y-separator is also chordal. Consequently, a chordal graph H_0 is obtained by adding edge $\{x, y\}$ and an edge from y to every vertex of every 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 an edge from X_0 in T_0 to $\mathcal{S}$. For an added fill edge $\{y, u'\}$ where u' is contained in some minimal x, y-separator of $H[N_G[C_0]]$, there exists a fill edge $\{u', v_i\}$ that is removed from H', where v_i is a vertex of the path P_1 in the other full component C_1. To see this, let $P_0 = x, u_1, u_2, ..., u_t, y$ be a chordless path in $H[N_G[C_0]]$, such that $u_j = u'$ for a $j \in \{1, \ldots, t\}$. 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 contain x and y, respectively. Every maximal clique on the unique path $X, Z_1, Z_2, ..., Z_\ell$ from X to Y in T contains a vertex v_i of path P_1, since Z_τ for $\tau \in \{1, ..., \ell\}$ separates x from y. On the other hand, every vertex $u_1, ..., u_t$ of path P_0 is contained in some Z_τ where $\tau \in \{1, ..., \ell\}$. The reason for this is that T is a tree and $Z_1, Z_2, ..., Z_\ell$ is the unique path from X to Y. If path P_0 started in clique X and left the path $Z_1, Z_2, ..., Z_\ell$ at clique Z_τ in the clique tree T, then P_0 has to enter Z_τ again, since T is a tree. Second entrance to Z_τ would be a contradiction to the claim that P_0 is chordless. Thus, there exists a fill edge from u' to at least one vertex in $\{v_1, ..., v_q\}$ 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 $\mathcal{S}$. ■

4 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. The input to the algorithm is an undirected graph $G = (V, E)$, and the algorithm either outputs 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.

In the subproblems generated by this branching algorithm, some vertices have a *marking*. As will be clear when the subproblems are analyzed, we will have the choice between adding a set of fill edges or concluding with a set of vertices that each must be incident to a fill edge. Vertices that we have decided to be incident to a fill edge 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 $M = \emptyset$. We can now make the following claim.

Lemma 7 *If* $(G{=}(V,E),k,r,M)$ *has a solution with* $\{x,y\}$ *as a fill edge, then* $((V,E \cup \{\{x,y\}\}),k-1,r-\gamma,M \setminus \{x,y\})$ *has a solution, where* $\gamma \in \{0,1,2\}$ *is the number of unmarked endpoints of* $\{x,y\}$.

Proof. Since $(G{=}(V,E),k,r,M)$ has a solution with $\{x,y\}$ as a fill edge, we only have to argue that this solution will also be a solution for $((V,E \cup \{\{x,y\}\}),k-1,r-\gamma,M \setminus \{x,y\})$. Adding a correct fill edge $\{x,y\}$ to E and reducing k by one enable us to still use the rest of the solution. Vertices x,y are removed from M, since the condition for putting them in M has been fulfilled. Finally r is updated to match the new number of available marks. ■

At several points during our algorithm, we write: *add an edge e to F and update accordingly.* Following Lemma 7, 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. As an example consider the triangulation of a chordless cycle with five vertices. There exist two fill edges and thus four endpoints, but these two fill edges are incident to only three vertices altogether.

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 chordless 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, $ab \geq 2$.

Moplexes with marked and unmarked vertices. As a consequence of Theorem 4 and Proposition 5, we have the following result.

Lemma 8 *Let* G *be a graph and* U *be a moplex in* G. *Then for every minimal triangulation(including the minimum triangulations) of* G, *there is a fill edge* $\{u,v\}$ *for every vertex* $u \in U$, *or* $N(U)$ *is a clique and there is no fill edge* $\{u,v\}$ *for* $u \in U$ *and* $v \notin U$.

Proof. Note first that $N(U)$ is a minimal separator of G. By Theorem 3 and the subsequent discussion, $N(U)$ is turned into a clique in a minimal triangulation of G if and only if no fill edge is added incident to a vertex of U. If $N(U)$ is not a clique in a minimal triangulation H of G, then every vertex u of U is involved in a chordless cycle containing u, the endpoints of a missing edge $\{x, y\}$ in $N(U)$, and vertices of a shortest path between x and y in another full component associated to $N(U)$. Hence, by Proposition 5, every vertex of U is incident to a fill edge in H. ∎

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

Finding moplexes with unmarked vertices. Then, the algorithm tests whether there is a moplex U that contains no marked vertices. If there is no such moplex, the algorithm returns NO. Safeness of this step comes from the following lemma, which follows immediately from Theorem 4 by simply considering the first moplex that is removed after being made simplicial.

Lemma 9 *Let G be a graph and let H be a minimum triangulation of G. There is a moplex U in G, such that no fill edge is incident to any vertex of U in H, and $N(U)$ is a clique in H.*

Proof. Consider a clique tree T of H, and let X be a leaf clique in the clique tree T. Now use $X \setminus N[V \setminus X]$ as the moplex U. ∎

We take such a moplex 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 Theorem 3 and the subsequent discussion, 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.

Moplexes missing one edge. Next, we test if $|F(S)| = 1$. By Lemma 6 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 6.

Lemma 10 *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 with $F(S) = \{\{x, y\}\}$.*

1. *If there is a unique vertex v^*, such that v^* is the last vertex on each chordless path from x to y through a full component of S not containing u, then there is a minimum triangulation of G that contains $\{x, y\}$ as a fill edge, and 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 chordless path from x to y through a full component of S not containing u, then there is a minimum triangulation of G that contains $\{x, y\}$ as a fill edge, and 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 is incident to at least one edge in $F \setminus E$. If $\{x, y\} \in F$, then there is nothing to prove, so let us assume that $\{x, y\} \notin F$. We will now analyze the triangulation $H' = (V, F')$ constructed in the proof of Lemma 6, and compare it to H. If every vertex in M is incident to a fill edge in $F' \setminus E$, then the proof of Lemma 6 can be applied directly. This corresponds to case 2 of the lemma.

Case 1 remains. Let us start by listing the vertices that in H' might loose their incident fill edges compared to H. For each connected component C_i, for $i \in \{1, ..., p\}$, fill edges incident to u in $H[N_G[C_i] \cup \{u\}]$ are moved to be incident to y instead in H'. This is unproblematic for any vertex not adjacent to y in G, but for neighbors of y, this means that they loose an incident fill edge. We used this saved edge in the proof of Lemma 6 to pay for the new and added fill edge $\{x, y\}$, and thus there exists exactly one such saved edge, one such neighbor of y, and such component containing such a neighbor, since H is an minimum triangulation. Let $v_1, ..., v_q$ be a chordless path from x to y in $H[N_G[C_1] \cup \{x, y\}]$. It follows that v_q is the single vertex that has lost one incident fill edge in H' compared to H. Vertex v_q is unique, since two vertices would allow us to save two fill edges, and would be a contradiction to H being a minimum triangulation. Taking $v^* = v_q$ completes the proof. ∎

Suppose we have a moplex U, with $F(N(U))$ consisting of the single edge $\{x, y\}$. The condition of Lemma 10 now becomes: there is a vertex v^* that is the last vertex on each chordless path in $G[V \setminus U]$ from x to y. A simple modification of standard breadth first search allows us to find v^* in linear time, if it exists. 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 8, that 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 time $\mathcal{O}(nm)$, except that the checking for all 4-cycles before any other operation is done requires $\mathcal{O}(m^2)$ time, once. 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 the results of [17] and [20] 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 $\mathcal{O}(k^2nm)$ time. By preprocessing the input by such an algorithm we get an additive time cost of $\mathcal{O}(k^2nm)$ 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 $\mathcal{O}(k^2nm + 3.0793^k)$.

Theorem 11 *The* Minimum Fill-In *problem can be solved in $\mathcal{O}(k^2nm+3.0793^k)$ time, using polynomial space.*

The algorithm can now be summarized as follows. Use the results of [17] and [20] to obtain a equivalent problem instance $(G' = (V', E'),\ k')$ where $k' \leq k$ and $|V'| = \mathcal{O}(k^2)$, in $\mathcal{O}(k^2nm)$. Use this problem to create the new initial problem instance $(G, k', 2k', \emptyset)$, where $r = 2k'$ and $M = \emptyset$. From this point on the problem instance is treated recursively. For a new instance, first check the trivial cases, and then search for 4-cycles. If a 4-cycle is found two recursive calls are made. If the graph contains a moplex with both marked and unmarked vertices, then mark all vertices of the moplex. If the graph contains a simplicial moplex where all vertices are unmarked, remove the moplex. If the graph contains a moplex U where all vertices are unmarked and $|F(N(U))| = 1$, apply Lemma 10. If every moplex contains only marked vertices, return NO, otherwise find a moplex containing unmarked vertices and make the two new recursive calls.

5 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 to the algorithm of Section 4 is that the mark is a vertex set containing the vertices which are candidates to add a fill edge incident to. I.e., marks are *annotated.* This is to properly execute the steps where we partition on clique separators. 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 [11]. We also allow the algorithm to return 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 step where we use the exact algorithm by Fomin and Villanger [11], as we ignore the marks in that step.

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 $T(k, r) \leq a^k \cdot b^k \cdot o(k)$ 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.

5.1 Polynomial time reduction rules

Trivial cases. Exactly like in the algorithm of Section 4, 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 chordless 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 by Theorem 3.

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

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 by Theorem 3 and its subsequent discussion. First, we test for each subproblem if there is a solution with at most two fill edges. If so, we solve this subproblem in polynomial time and use the obtained solution 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. Each of these subinstances is solved with parameter $k-3\alpha+3$; we answer yes if the total number of all minimum triangulation for all subinstances is at most k.

Thus, we need to choose a and b such that $T(k, r) \leq \alpha \cdot T(k - 3\alpha + 3, r)$ 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 use Lemma 10, 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 $S \subseteq N(v)$, 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 10 exists, if v^* exists, we 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: $a \geq b$.

5.2 Branching rules

Above, we have given a number of reduction steps. We repeat applying reductions, until none of the above reduction steps applies, and then consider, in the given order, the following branching steps.

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

Before continuing we have to strengthen the claim in Lemma 8, to only talk about single vertices and minimal separators.

Lemma 12 *Let $G = (V, E)$ be a graph and for a vertex $v \in V$, let S be a minimal separator in $N(v)$, and let W be a connected component of $G[V \setminus N[v]]$ such that $N(W) = S$. Then for any minimal triangulation(including any minimum triangulation) of G, S is a clique or $\{v, w\}$ is a fill edge for $w \in W$.*

Proof. Between any pair x, y of non adjacent vertices in $G[S]$, there exists a path x, v, y, and a path from x to y only passing through vertices of W. Using these two paths and the fact that S separates v from W, enable us to conclude that there exists a chordless cycle $C_{x,y} = x, v, y, u_1, u_2, \ldots, u_p, x$ where $u_i \in W$ for $i \in \{1, \ldots, p\}$. At this point we can apply Proposition 5. Graph $G[x, v, y, u_1, u_2, \ldots, u_p]$ is an induced cycle, and thus by Proposition 5 we know that any triangulation of $G[x, v, y, u_1, u_2, \ldots, u_p]$ either contains $\{v, u_i\}$ for $i \in \{1, \ldots, p\}$ or $\{x, y\}$ as a fill edge. Running over all pairs of non adjacent vertices in $G[S]$ we can conclude that the statement holds. ∎

Minimal separator S with $|F(S)| \geq 3$. Next we 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 each of 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 which is safe by Lemma 12, 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. It is sufficient for our purpose to only annotated vertices in one connected component. 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 a chordless path $u_1, u_2, ..., u_r$ from x to y only containing vertices of W. Vertex set $\{y, v, x, u_1, u_2, ..., u_r\}$ induces a cycle in G. By Proposition 5, v has a fill edge to one of the vertices in $\{u_1, u_2, ..., u_r\}$ 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 13 *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. In the first subinstance, we have k reduced by $|F(S)| \geq 3$, and r reduced by at least two. This can be seen as follows. There are no chordless 4-cycles in G, and thus the edges of $F(S)$ form a connected graph on vertices in S that are endpoints of edges in $F(S)$. A consequence of this is that at most $|F(S)|+1$ vertices will be incident to the edges in $F(S)$. As there are $2|F(S)|$ endpoints of edges in $F(S)$, $|F(S)| \geq 3$, and edges of $F(S)$ are incident to at most $|F(S)| + 1$ vertices, there is at least 2 more endpoints of edges than there are marked vertices with incident fill edges. In the second subinstance, r is decreased by one, as we place one mark. This gives: $T(k, r) \leq T(k-3, r-2) + T(k, r-1)$, leading to the condition $a^3b^2 \geq 1 + a^3b$.

Notice that after this step has been carried out, 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 elements of $F(S)$ are added to G as fill edges. 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. This test can be carried out in polynomial time [24].

If this test fails for all unmarked v and minimal separators in $N(v)$, we continue with the next step. Suppose now that we have found a vertex v and two connected components W_1 and W_2 of $G[V \setminus S]$ with $G'[N[W_1]]$ and $G'[N[W_2]]$ 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. We can use any FPT-algorithm for MINIMUM FILL-IN to do this in polynomial time.

There are now two cases, either $G'[N[W_1]]$ or $G'[N[W_2]]$ can be triangulated by adding at most three fill edges, or not. For the first case let us assume without loss of generality, that $G'[N[W_1]]$ can be triangulated by adding $k_1 \leq 3$ fill edges. We recurse on two subinstances. The first subinstance handles the case where S is completed into a clique. Here we need to recurse on $G'[N[W_2]]$(since this is the unsolved subgraph), where k is reduced by $|F(S)|+k_1 \geq 3$. Similar as above, the at least three fill edges form a connected graph on the set of its endpoints, and hence two vertices will get two incident fill edges and thus, r decreases with at least two. In the second subinstance(under the condition that $G'[N[W_1]]$ can be triangulated by adding $k_1 \leq 3$ fill edges), we mark v, which is safe due to Lemma 12. Thus, the first case results in the recursive condition: $T(k,r) \leq T(k-3,r-2)+T(k,r-1)$, giving $a^3b^2 \geq 1+a^3b$.

Now for the second case, where neither of $G'[N[W_1]]$ and $G'[N[W_2]]$ can be triangulated by adding at most three fill edges. We consider again two subinstances, but directly split the first instance again. In the first main subinstance, we complete S into a clique. Now, S is a clique separator. Thus, we can instead solve two smaller subinstances, and recurse on $G'[N[W_1]]$ and $G'[V \setminus N[W_1]]$. In each of these subinstances, the parameter k is reduced by six: we used two fill edges for completing S and must use at least four fill edges in the other subinstance. As the two fill edges in S share an endpoint, we decrease r by one. In the second main subinstance, we mark v, which is safe due to Lemma 12. This gives the recursive condition: $T(k,r) \leq 2T(k-6,r-1)+T(k,r-1)$, giving $a^6b \geq 2+a^6$.

5.3 Using a list of potential maximal cliques

Suppose now that none of the steps above could be applied. 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 [11].

This step consists of a number of substeps:

1. We return NO if there are more than $k+1$ marked vertices.
2. We partition the unmarked vertices into groups, and return NO if there are more than $3k/2+1$ groups.
3. We make a list of minimal separators of G: this list will in general not contain all minimal separators, but it contains all that are needed to make step 5 succeed.
4. We make a list of potential maximal cliques of G. Again, this list does not need all potential maximal cliques, but it contains all that are needed to make step 5 succeed.
5. We use the two lists above as input to an algorithm by Fomin et al. [10] and determine if there is a triangulation with fill at most k, and if so, find a triangulation with minimum fill.

Each of these steps, with details and correctness, will be discussed extensively below, but we start with deriving a number of graph theoretic lemmas.

From the fact that none of the above steps can be applied, several conclusions can be drawn. 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. As a consequence, we can make the following observations and definitions on the provided graph instance.

- G is 4-cycle free,
- for an unmarked vertex v,
  - $N(v)$ contains a unique minimal separator S_v,
  - $|F(S_v)| = 2$,
  - one vertex $x \in S_v$ is an endpoint of both edges $\{x, y\}, \{x, z\} \in F(S_v)$,
  - $G[V \setminus N[v]]$ contains one connected component W_v,
  - adding the edges $F(S_v)$ to $G[N[v]]$ results in a chordal graph,
  - adding the edges $F(S_v)$ to $G[N[W_v]]$ results in a non-chordal graph,
  - let C_v be the connected component of $G[V \setminus S_v]$ containing v.

Lemma 14 *Let u and v be two unmarked vertices such that $S_u \subset N[v]$. Then u and v are contained in the same connected component C_v of $G[V \setminus S_v]$.*

Proof. Vertices u and v are adjacent by definition if $v \in S_u$. If $v \notin S_u$, then both u and v have the endpoints of the edges in $F(S_u)$ in their neighborhood, and thus u and v are adjacent, since there are no chordless 4-cycles.

Vertex v is contained in C_v by definition, and by the edge $\{u, v\}$ vertex u is contained in $N[C_v] = C_v \cup S_v$. If $u \in C_v$ then the lemma follows. Let us on the contrary assume that $u \in S_v$. Since $u \in S_v$ and S_v is a minimal separator, then there exists a neighbor w of u in W_v. This implies that $W_v \subset C_u$, since $S_u \subset N[v]$, $G[W_v]$ is connected, $N[v] \cap W_v = \emptyset$, and $w \in (N(u) \cap W_v) \setminus S_u$.

From the conditions of the lemma $G[N[W_u]]$ is not chordal when edge set $F(S_u)$ is added. Let $C_{x,y} = x, v_1, v_2, \ldots, v_p, y$ be a chordless cycle of length at least four in $G[N[W_u]]$ where edges $F(S_u)$ are added, such that $x, y \in S_u$ and $v_1, v_2, \ldots, v_p \in W_u$ where $p \geq 2$. Observe now that $C_{x,u,y} = u, x, v_1, v_2, \ldots, v_p, y$ is an induced cycle in $G[N[v]]$ since, $W_u \cup S_u \subseteq N[v]$, $x, y \in S_u \subset N(u)$, $v_1, v_2, \ldots, v_p \in W_u$, and $W_u \cap N(u) = \emptyset$. Finally $W_u \subseteq C_v$, since $W_v \subset C_u$, $S_v \subset N[C_u]$ (because $W_v \subset C_u$), and thus $W_u \subseteq C_v$ since $C_v \subseteq V \setminus (S_v \cup W_v)$, $W_u \subseteq V \setminus N[C_u]$, and $S_v \cup W_v \subset N[C_u]$.

We can now argue for the contradiction. Graph $G[N[v]]$ is chordal when edges $F(S_v)$ are added, and $G[N[v]]$ contains the induced cycle $u, x, v_1, v_2, \ldots, v_p, y$ for $p \geq 2$ and $v_1, v_2, \ldots, v_p \notin S_v$ and thus the only possible fill edge of this cycle contained in $F(S_v)$ is $\{x, y\}$. Adding edge $\{x, y\}$ to cycle $u, x, v_1, v_2, \ldots, v_p, y$ makes $x, v_1, v_2, \ldots, v_p, y$ to an induced cycle of length at least four which is a contradiction to $G[N[v]]$ being a chordal graph when edges $F(S_v)$ are added. ∎

Lemma 15 *Given the conditions above and two unmarked vertices u and v, $N[u] = N[v]$ or $F(S_u) \cap F(S_v) = \emptyset$.*

Proof. Let $|F(S_u) \cap F(S_v)| = 1$, and let $\{x, y\}$ be the edge in $F(S_u) \cap F(S_v)$. The edge $\{u, v\} \in E$, since the graph otherwise would contain a 4-cycle.

Vertex u is not contained in S_v. On the contrary if $u \in S_v$, then $S_v \subset N[u]$, since an edge $\{u, z'\} \in F(S_v)$ for $z' \in \{x, y\}$ would make either x or y non adjacent to u. By Lemma 14 u and v are contained in C_u.

For simplicity let us assume that $u \in C_v$, the opposite case is symmetric. Since $C_v \cup S_v = N[v]$ and $u \in C_v$ we have that $C_u \cup S_u \subset N[v]$.

Let $\{x, z\}$ be the second edge in $F(S_u)$, and let $x, u, z, w_1, w_2, \ldots, w_p$ for $w_1, w_2, \ldots, w_p \in W_u$ be an induced cycle in G, where $1 \leq p$. If $z \in S_v$ we have a contradiction since $|F(S_u) \cap F(S_v)| = 2$ in this case. We know that $G[N[C_v]]$ is chordal when the two edges in $F(S_v)$ are added, and we know that $x \in S_v$, $v, z \in C_v$. Let i be the smallest number such that $w_i \in S_v$. This is a contradiction since $x, u, z, w_1, w_2, \ldots, w_i$ is an induced cycle of length at least four in $G[N[C_u]]$ when the two edges in $F(S_u)$ are added.

It remains to consider the case where $|F(S_u) \cap F(S_v)| = 2$. We will now show that $N[u] = N[v]$. Like in the first half of the proof we can notice that $\{u, v\} \in E$, 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 $F(S_u)$. Applying Lemma 14 to this observation provide us with the knowledge that $u \in C_v$ and $v \in C_u$. On the contrary assume that $w \in N[v] \setminus N[u]$ (equivalently $w \in N[u] \setminus N[v]$). By the observation above, $v \in C_u$ and thus $N[v] \subseteq N[u] = C_u \cup S_u$. ∎

Lemma 16 *Given the conditions above, and 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 $w_i \in S_v$ is unmarked for each $i \in \{1, 2, 3, 4\}$. From the conditions of the lemma we know that edges of $F(S_v)$ are only incident to three vertices in S_v. Let w_i for $i \in \{1, 2, 3, 4\}$, be the vertex such that $S_v \subset N[w_i]$. The existence of vertex $w_i \in S_v$ is now a contradiction to Lemma 14 ∎

Lemma 17 *Given a graph $G = (V, E)$ with no chordless 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, 13, 26], 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, such that $u \notin \{x, y\}$ and $v \notin \{x, y\}$. There 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. Hence the first substep: we return NO at this point if there are more than $k + 1$ marked vertices. So assume there are at most $k + 1$ marked vertices.

The next step of the algorithm is to control the unmarked vertices. What we will do is to partition the unmarked vertices into groups, bound the number of groups, and show that an algorithm by Fomin et. al. [10] has a running time that can be bounded by a function of the number of groups and marked vertices.

For each unmarked vertex v we know that $|F(S_v)| = 2$, and by Lemma 15, we know that for pairs of unmarked vertices u and v, $F(S_v) = F(S_u)$, 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.

Lemma 18 *If G has a triangulation with at most k fill edges, then there are at most $3k/2+1$ groups of unmarked vertices.*

Proof. Consider a minimum triangulation H, which respects all given marks. By Lemma 17 there are 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, this means that the group can be considered as a single unit or vertex. There are at most $k+1$ groups that have fill edges incident to its vertices. The number of groups that do not have fill edges incident to its vertices is at most $k/2$, since each such group has two private fill edges in its neighborhood. ∎

So, if we have more than $3k/2+1$ groups, we return NO. The final step of our algorithm is to apply an algorithm by Fomin et al. [10] to compute a minimum triangulation, with one important modification concerning the groups of unmarked vertices. Fomin et al. [10] have shown that 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 [10] is more powerful than stated. Inspection of this algorithm shows that it also can be used to obtain the following result.

Theorem 19 (See Fomin et al. [10]) *There is an algorithm that, given a graph $G = (V, E)$, a list L_1 of potential maximal cliques in G, and a list L_2 of minimal separators in G, returns in time $\mathcal{O}(n^3(|L_1| + |L_2|))$ a triangulation H of G, such that*

- *each maximal clique in H is an element of L_1*
- *each minimal separator in H is an element of L_2*
- *the fill of H is minimal amongst all triangulations that fulfill the above two criteria.*

We can use this result as follows: we list potential maximal cliques and minimal separators of G, but we can avoid listing such potential maximal cliques and minimal separators when they would imply more than k fill edges. As minimal separators in a chordal graph are cliques, we only have to list minimal separators S with $|F(S)| \leq k$ and potential maximal cliques Ω with $|F(\Omega)| \leq k$, when we want to solve the MINIMUM FILL-IN problem with parameter k.

It is not hard to see that if u and v have the same closed neighborhood, then each potential maximal clique that contains u also contains v; the same holds for each minimal separator in a minimal triangulation of G. 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 direct edges or paths 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. Thus, when listing minimal separators and potential maximal cliques, we can treat vertices with the same closed neighborhood as a single vertex.

Consider a collection of $4(\sqrt{2k}+1)$ of the $3k/2+1$ groups of unmarked vertices. By Lemma 16, for each group of unmarked vertices, there are at most three unmarked vertices not in this group that are incident to some vertices in the group, thus there exists at least $\sqrt{2k}+1$ groups with no edges between vertices of the groups. Such a set of vertices can never belong to a clique in a triangulation with at most k fill edges, as there are more than k pairs of vertices in this set that are non-adjacent. Thus, we can limit our lists of minimal separators and potential maximal cliques to sets that contain at most $4(\sqrt{2k}+1)$ of the $3k/2+1$ groups of unmarked vertices.

We need the following result by Bouchitté and Todinca.

Proposition 20 ([3]) *Let Ω be a potential maximal clique of $G=(V,E)$, let u be a vertex of G, and let $G' = G[V \setminus \{u\}]$. Then one of the following holds:*

1. *Either Ω, or $\Omega \setminus \{u\}$ is a potential maximal clique of G';*
2. *$\Omega = S \cup \{u\}$, where S is a minimal separator of G;*
3. *Ω is a nice potential maximal clique.*

Lemma 21 *Given a graph G, let M be the set of marked vertices, let q be the number of groups of unmarked vertices, and let $k \le |M|-1$. 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. Let Ω be a potential maximal clique of a graph G and let $S \subset \Omega$ be a minimal separator of G. We say that S is an *active separator for* Ω, if Ω is not a clique in the graph obtained from G by completing all the minimal separators contained in Ω, except S. A potential maximal clique Ω containing an active separator (for Ω) is called a *nice potential maximal clique*.

To prove Lemma 21, we show first that all minimal separators and nice potential maximal cliques containing $W \subset (V \setminus M)$ where $|W| \le \ell$ can be listed in $\mathcal{O}^*(1.7549^{k+1})$ time and space. For the first part of this proof we assume that W is the intersection between $V \setminus M$ and the minimal separator or nice potential maximal clique we are searching for.

Due to Proposition 1 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 $|C_1 \cap M| \le |C_2 \cap M|$. Fomin and Villanger [11] 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 [11] 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 nice potential maximal cliques is done in the same way, but requires some more detailes. By [25](Lemma 2), there exists a partitioning of $G[V \setminus \Omega]$ into three sets A_1, A_2, A_3, such that any two of these sets can be used along with an additional vertex z to obtain a connected vertex set Z, such that $\Omega = N(Z) \cup \{z\}$. Without loss of generality let $|A_1 \cap M| \leq |A_2 \cap M| \leq |A_3 \cap M|$. Use $A_1 \cup A_2 \cup \{z\}$ as the set Z, and and the algorithm in [11] to list the nice potential maximal cliques containing W and not $V \setminus (M \cup W)$ in $\mathcal{O}^*(1.7549^{k+1})$ time and space.

In the second part we will consider the problem of listing non-nice potential maximal cliques. The remaining potential maximal cliques can be generated from the set of minimal separators and nice potential maximal cliques by the following procedure (see [10, 11]). Let $v_1, v_2, ..., v_n$ be an ordering of V and let $V_i = \bigcup_{j=1}^{i} v_j$. By Proposition 20, every potential maximal clique Ω in $G[V_{i+1}]$ that is not yet listed (i.e. is not obtained by adding a vertex to a minimal separator or which is not nice) is either also potential maximal clique in $G[V_i]$ or obtained from a potential maximal clique in $G[V_i]$ by adding v_{i+1}. Thus by performing dynamic programming one can list all remaining potential maximal cliques in time used to list minimal separators and nice potential cliques for $G[V_i]$ for every $i \in \{1, \ldots, n\}$.

In our case the situation the goal is to list only potential maximal cliques where the intersection with $V \setminus M$ contains at most ℓ vertices. Fortunately for us the algorithm for general graphs only require the presence of minimal separators and nice potential maximal cliques containing fewer vertices than the target potential maximal clique. Thus, we can use the same dynamic programming, and just ignore potential maximal cliques which has to big intersection. A similar technique is used in [11] to list potential maximal cliques of size at most k. ∎

As discussed above, minimal separators and potential maximal cliques that contain vertices from more than $4(\sqrt{2k}+1)$ groups cannot belong to a triangulation with at most k fill edges. Hence, we apply Lemma 21 with $\ell = 4(\sqrt{2k}+1)$. Thus, in $\mathcal{O}^*(1.7549^k)$ time, we list all minimal separators and potential maximal cliques that can contribute to a triangulation with at most k fill edges. Given these lists, we now employ the algorithm of Fomin et al. [10] (see Theorem 19) 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.

Note that the algorithm in Theorem 19 does not need to respect the marks. Thus, we possibly obtain for the subproblem a solution while there is no solution for this subproblem respecting the mark; as discussed at the start of this section, this does not harm the overall correctness of the algorithm.

We obtain the following condition for the running time: $T(k,r) \leq 1.7549^k$, and thus $a \geq 1.7549$.

Analyzing the running time. We derived the following conditions on a and b, such that, if these hold, then by induction it follows that $T(k,r) \leq a^k b^k \cdot o(k)$: for all integers $\alpha \geq 2$: $a^{3\alpha-3} \geq \alpha$, $a \geq b$, $ab \geq 2$, $a^3b^2 \geq 1 + a^3b$, $a^6b \geq 2 + a^6$, $a \geq 1.7549$. 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 us 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 $\mathcal{O}(k^2nm + 2.35965^k)$, and requires $\mathcal{O}^*(1.7549^k)$ space.

Theorem 22 *The* Minimum Fill-In *problem can be solved in* $\mathcal{O}(k^2nm + 2.35965^k)$ *time,* *using* $\mathcal{O}^*(1.7549^k)$ *space.*

6 Conclusions

In this paper, we have presented new 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. It would be an interesting study to experimentally evaluate the first algorithm, and variants of it where some of the steps of the second algorithm are added to it.

The bounding conditions in the analysis of the running time of the second algorithm are $a \geq 1.7549$, and $a^3b^2 \geq 1 + a^3b$. Thus, a faster algorithm could be possibly obtained by either finding a faster version of the algorithm that lists the minimal separators and potential maximal cliques (i.e., speeding up the algorithm from Fomin and Villanger [11], see Lemma 21), or the two steps that give the condition $a^3b^2 \geq 1 + a^3b$: minimal separators S with $|F(S)| = 3$ and the first case where we split into two non-chordal graphs.

We obtain running times of the form $O(k^2nm + c^k)$ time by first applying the kernelization algorithm for Minimum Fill-In by Kaplan et al. [17], see also [20]. To obtain an algorithm with a better asymptotic running time, it would be useful to have an algorithm that obtains a kernel of size polynomial in k for Minimum Fill-In whose running time is $o(k^2nm)$.

Acknowledgements

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.*, Theoretical Computer Science, 276 (2002), pp. 17–32.

[4] P. Buneman, *A characterization of rigid circuit graphs*, Discrete Mathematics, 9 (1974), pp. 205–212.

[5] L. Cai, *Fixed-parameter tractability of graph modification problems for hereditary properties*, Information Processing Letters, 58 (1996), pp. 171–176.

[6] F. R. K. Chung and D. Mumford, *Chordal completions of planar graphs*, Journal of Combinatorial Theory, Series B, 31 (1994), pp. 96–106.

[7] G. A. Dirac, *On rigid circuit graphs*, Anh. Math. Sem. Univ. Hamburg, 25 (1961), pp. 71–76.

[8] R. G. Downey and M. R. Fellows, *Parameterized complexity*, Springer-Verlag, New York, 1999.

[9] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Springer-Verlag, New York, 2006.

[10] F. V. Fomin, D. Kratsch, and I. Todinca, *Exact (exponential) algorithms for treewidth and minimum fill-in*, in Proceedings of the 31st International Colloquium on Automata, Languages and Programming, ICALP 2004, vol. 3142 of Lecture Notes in Computer Science, Springer, 2004, pp. 568–580.

[11] F. V. Fomin and Y. Villanger, *Treewidth computation and extremal combinatorics*, in Proceedings of the 35th International Colloquium on Automata, Languages and Programming, ICALP 2008, vol. 5125 of Lecture Notes in Computer Science, Springer, 2008, pp. 210–221.

[12] D. R. Fulkerson and O. A. Gross, *Incidence matrices and interval graphs*, Pacific Journal of Mathematics, 15 (1965), pp. 835—855.

[13] F. Gavril, *The intersection graphs of subtrees in trees are exactly the chordal graphs*, Journal of Combinatorial Theory, Series B, 16 (1974), pp. 47–56.

[14] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[15] P. Heggernes, *Minimal triangulations of graphs: A survey*, Discrete Mathematics, 306 (2006), pp. 297–317.

[16] 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 Proceedings of the 35th annual symposium on Foundations of Computer Science, FOCS'94, IEEE Computer Science Press, 1994, pp. 780–791.

[17] ——, *Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs*, SIAM Journal on Computing, 28 (1999), pp. 1906–1922.

[18] S. L. Lauritzen and D. J. Spiegelhalter, *Local computations with probabilities on graphical structures and their applications to expert systems*, Journal of the Royal Statistical Society, Series B, 50 (1988), pp. 157-224.

[19] C. Lekkerkerker and J. Boland, *Representation of a finite graph by a set of intervals on the real line*, Fund. Math., 51 (1962), pp. 45–64.

[20] A. Natanzon, R. Shamir, and R. Sharan, *A polynomial approximation algorithm for the minimum fill-in problem*, SIAM Journal on Computing, 30 (2000), pp. 1067–1079.

[21] R. Niedermeier, *Invitation to Fixed-Parameter Algorithms*, Oxford University Press, 2006.

[22] A. Parra and P. Scheffler, *Characterizations and algorithmic applications of chordal graph embeddings*, Discrete Applied Mathematics, 79 (1997) pp. 171–188.

[23] D. J. Rose, *Triangulated graphs and the elimination process*, Journal of Mathematical Analysis and Applications, 32 (1970), pp. 597–609.

[24] R. E. Tarjan and M. Yannakakis, *Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs*, SIAM Journal on Computing, 13 (1984), pp. 566–579.

[25] Y. Villanger, *Improved exponential-time algorithms for treewidth and minimum fill-in.*, in Proceedings of the 7th Latin American Theoretical Informatics Symposium, LATIN 2006, vol. 3887 of Lecture Notes in Computer Science, Springer, 2006, pp. 800–811.

[26] J. Walter, *Representations of rigid cycle graphs*, PhD thesis, Wayne State University, USA, 1972.

[27] M. Yannakakis, *Computing the minimum fill-in is NP-complete*, SIAM Journal on Algebraic and Discrete Methods, 2 (1981), pp. 77–79.