# Subexponential Parameterized Algorithm for Minimum Fill-in *

Fedor V. Fomin † Yngve Villanger †

**Abstract**

The MINIMUM FILL-IN problem is to decide if a graph can be triangulated by adding at most $k$ edges. Kaplan, Shamir, and Tarjan [FOCS 1994] have shown that the problem is solvable in time $\mathcal{O}(2^{\mathcal{O}(k)}+k^2nm)$ on graphs with $n$ vertices and $m$ edges and thus is fixed parameter tractable. Here, we give the first subexponential parameterized algorithm solving MINIMUM FILL-IN in time $\mathcal{O}(2^{\mathcal{O}(\sqrt{k}\log k)}+k^2nm)$. This substantially lower the complexity of the problem. Techniques developed for MINIMUM FILL-IN can be used to obtain subexponential parameterized algorithms for several related problems including MINIMUM CHAIN COMPLETION, CHORDAL GRAPH SANDWICH, and TRIANGULATING COLORED GRAPH.

## 1 Introduction

A graph is *chordal* (or triangulated) if every cycle of length at least four contains a chord, i.e. an edge between nonadjacent vertices of the cycle. The MINIMUM FILL-IN problem (also known as MINIMUM TRIANGULATION and CHORDAL GRAPH COMPLETION) is

> MINIMUM FILL-IN
> *Input:* A graph $G = (V, E)$ and a non-negative integer $k$.
> *Question:* Is there $F \subseteq [V]^2$, $|F| \leq k$, such that graph $H = (V, E \cup F)$ is chordal?

The name fill-in is due to the fundamental problem arising in sparse matrix computations which was studied intensively in the past. During Gaussian eliminations of large sparse matrices new non-zero elements called fill can replace original zeros thus increasing storage requirements and running time needed to solve the system. The problem of finding the right elimination ordering minimizing the amount of fill elements can be expressed as the MINIMUM FILL-IN problem on graphs [45, 47]. See also [15, Chapter 7] for a more recent overview

*This research was partially supported by the Research Council of Norway.
†Department of Informatics, University of Bergen, Bergen, Norway, {fedor.fomin,yngve.villanger}@ii.uib.no

of related problems and techniques. Besides sparse matrix computations, applications of Minimum Fill-in can be found in database management [2], artificial intelligence, and the theory of Bayesian statistics [13, 27, 40, 51]. The survey of Heggernes [30] gives an overview of techniques and applications of minimum and minimal triangulations.

Minimum Fill-in (under the name Chordal Graph Completion) was one of the 12 open problems presented at the end of the first edition of Garey and Johnson's book [26] and it was proved to be NP-complete by Yannakakis [52]. Kaplan et al. proved that Minimum Fill-in is fixed parameter tractable by giving an algorithm of running time $\mathcal{O}(m16^k)$ in [36] and improved the running time to $\mathcal{O}(k^6 16^k + k^2 mn)$ in [37], where $m$ is the number of edges and $n$ is the number of vertices of the input graph. There were several algorithmic improvements resulting in decreasing the constant in the base of the exponents. In 1996, Cai [11], reduced the running time to $\mathcal{O}((n+m)\frac{4^k}{k+1})$. The fastest parameterized algorithm known prior to our work is the recent algorithm of Bodlaender et al. with running time $\mathcal{O}(2.36^k + k^2 mn)$ [4].

In this paper we give the first subexponential parameterized algorithm for Minimum Fill-in. The last chapter of Flum and Grohe's book [21, Chapter 16] concerns subexponential fixed parameter tractability, the complexity class SUBEPT, which, loosely speaking—we skip here some technical conditions—is the class of problems solvable in time $2^{o(k)} n^{O(1)}$, where $n$ is the input length and $k$ is the parameter. Subexponential fixed parameter tractability is intimately linked with the theory of exact exponential algorithms for hard problems, which are better than the trivial exhaustive search, though still exponential [22]. Based on the fundamental results of Impagliazzo et al. [33], Flum and Grohe established that most of the natural parameterized problems are not in SUBEPT unless Exponential Time Hypothesis (ETH) fails. Until recently, the only notable exceptions of problems in SUBEPT were the problems on planar graphs, and more generally, on graphs excluding some fixed graph as a minor [16]. In 2009, Alon et al. [1] used a novel application of color coding to show that parameterized Feedback Arc Set in Tournaments is in SUPEPT. Minimum Fill-in is the first problem on general graphs which appeared to be in SUBEPT.

**General overview of our approach.** Our main tool in obtaining subexponential algorithms is the theory of minimal triangulations and potential maximal cliques of Bouchitté and Todinca [8]. This theory was developed in contest of computing the treewidth of special graph classes and was used later in exact exponential algorithms [23, 24]. A set of vertices $\Omega$ of a graph $G$ is a potential maximal clique if there is a minimal triangulation such that $\Omega$ is a maximal clique in this triangulation. Let $\Pi$ be the set of all potential maximal cliques in graph $G$. The importance of potential maximal cliques is that if we are given the set $\Pi$, then by using the machinery from [8, 23], it is possible compute an optimum triangulation in time up to polynomial factor proportional to $|\Pi|$.

Let $G$ be an $n$-vertex graph and $k$ be the parameter. If $(G, k)$ is a YES instance of the Minimum Fill-in problem, then every maximal clique of every optimum triangulation is obtained from some potential maximal clique of $G$ by adding at most $k$ fill edges. We call such potential maximal clique *vital*. To give a general overview of our algorithm, we start with the approach that does not work directly, and then explain what has to be changed to suceed. The algorithm consists of three main steps.

Step A. Apply a kernelization algorithm that in time $n^{O(1)}$ reduces the problem an instance to instance of size polynomial in $k$;

Step B. Enumerate all vital potential maximal cliques of an $n$-vertex graph in time $n^{o(k/\log k)}$. By Step A, $n = k^{O(1)}$, and thus the running time of enumeration algorithm and the number of vital potential maximal cliques is $2^{o(k)}$;

Step C. Apply the theory of potential maximal clique to solve the problem in time proportional to the number of vital potential maximal cliques, which is $2^{o(k)}$.

Step A, kernelization for Minimum Fill-in, was known prior to our work. In 1994, Kaplan et al. gave a kernel with $\mathcal{O}(k^5)$ vertices. Later the kernelization was improved to $O(k^3)$ in[37] and then to $2k^2 + 4k$ in [43]. Step C, with some modifications, is similar to the algorithm from [8, 23]. This is Step B which does not work and instead of enumerating vital potential maximal cliques we make a "detour". We use branching (recursive) algorithm that in subexponential time outputs subexponential number of graphs avoiding a specific combinatorial structure, non-reducible graphs. In non-reducible graphs we are able to enumerate vital potential maximal cliques. Thus Step B is replaced with

Step B1. Apply branching algorithm to generate $n^{\mathcal{O}(\sqrt{k})}$ non-reducible instances such that the original instance is a YES instance if and only if at least one of the generated non-reducible instances is a YES instance;

Step B2. Show that if $G$ is non-reducible, then all vital potential maximal cliques of $G$ can be enumerated in time $n^{\mathcal{O}(\sqrt{k})}$.

Putting together Steps A, B1, B2, and C, we obtain the subexponential algorithm.

It follows from our results that several other problems belong to SUBEPT. We show that within time $\mathcal{O}(2^{\mathcal{O}(\sqrt{k}\log k)} + k^2nm)$ it is possible to solve Minimum Chain Completion, and Triangulating Colored Graph. For Chordal Graph Sandwich, and we show that deciding if a sandwiched chordal graph $G$ can be obtained from $G_1$ by adding at most $k$ fill edges, is possible in time $\mathcal{O}(2^{\mathcal{O}(\sqrt{k}\log k)} + k^2nm)$.

A *chain* graph is a bipartite graph where the sets of neighbors of vertices form an inclusion chain. In the Minimum Chain Completion problem, we are asked if a bipartite graph can be turned into a chain graph by adding at most $k$ edges. The problem was introduced by Golumbic [28] and Yannakakis

[52]. The concept of chain graph has surprising applications in ecology [41, 46]. Feder et al. in [20] gave approximation algorithms for this problem.

The TRIANGULATING COLORED GRAPH problem is a generalization of MINIMUM FILL-IN. The instance is a graph with some of its vertices colored; the task is to add at most $k$ fill edges such that the resulting graph is chordal and no fill edge is monochromatic. We postpone the formal definition of the problem till Section 7. The problem was studied intensively because of its close relation to PERFECT PHYLOGENY PROBLEM—fundamental and long-standing problem for numerical taxonomists [7, 10, 35]. The TRIANGULATING COLORED GRAPH problem is $NP$-complete [6] and $W[t]$-hard for any $t$, when parametrized by the number of colours [5]. However, the problem is fixed parameter tractable when parameterized by the number of fill edges.

In CHORDAL GRAPH SANDWICH we are given two graphs $G_1$ and $G_2$ on the same vertex set, and the question is if there is a chordal graph $G$ which is a supergraph of $G_1$ and a subgraph of $G_2$. The problem is a generalization of TRIANGULATING COLORED GRAPH. We refer to the paper of Golumbic et al. [29] for a general overview of graph sandwich problems.

The remaining part of the paper is organized as follows. Section 2 contains definitions and preliminary results. In Section 3, we give branching algorithm, Step B1. Section 4 provides algorithm enumerating vital potential maximal cliques in non-reducible graphs, Step B2. This is the most important part of our algorithm. It is based on a new insight into the combinatorial structure of potential maximal cliques. In Section 5, we show how to adapt the algorithm from [8, 23] to implement Step C. The main algorithm is given in Section 6. In Section 7, we show how the ideas used for MINIMUM FILL-IN can be used to obtain subexponential algorithms for other problems. To implement our strategy for CHORDAL GRAPH SANDWICH, we have to provide a polynomial kernel for this problem. We conclude with open problems in Section 8.

# 2 Preliminaries

We denote by $G = (V, E)$ a finite, undirected and simple graph with vertex set $V(G) = V$ and edges set $E(G) = E$. We also use $n$ to denote the number of vertices and $m$ the number of edges in $G$. For a nonempty subset $W \subseteq V$, the subgraph of $G$ induced by $W$ is denoted by $G[W]$. We say that a vertex set $W \subseteq V$ is *connected* if $G[W]$ is connected. The *open neighborhood* of a vertex $v$ is $N(v) = \{u \in V : uv \in E\}$ and the *closed neighborhood* is $N[v] = N(v) \cup \{v\}$. For a vertex set $W \subseteq V$ we put $N(W) = \bigcup_{v \in W} N(v) \setminus W$ and $N[W] = N(W) \cup W$. Also for $W \subset V$ we define $\mathbf{fill}_G(W)$, or simple $\mathbf{fill}(W)$, to be the number of non-edges of $W$, i.e. the number of pairs $u \neq v \in W$ such that $uv \notin E(G)$. We use $G_W$ to denote the graph obtained from graph $G$ by completing its vertex subset $W$ into a clique. We refer to Diestel's book [17] for basic definitions from Graph Theory.

**Chordal graphs and minimal triangulations.** *Chordal* or *triangulated* graphs is the class of graphs containing no induced cycles of length more than three. In other words, every cycle of length at least four in a chordal graph contains a chord. Graph $H = (V, E \cup F)$ is said to be a *triangulation* of $G = (V, E)$ if $H$ is chordal. The triangulation $H$ is called *minimal* if $H' = (V, E \cup F')$ is not chordal for every edge subset $F' \subset F$ and $H$ is a *minimum* triangulation if $H' = (V, E \cup F')$ is not chordal for every edge set $F'$ such that $|F'| < |F|$. The edge set $F$ for the chordal graph $H$ is called the *fill* of $H$, and if $H$ is a minimum triangulation of $G$, then $|F|$ is the minimum fill-in for $G$.

Minimal triangulations can be described in terms of vertex eliminations (also known as Elimination Game) [45, 25]. Vertex elimination procedure takes as input a vertex ordering $\pi\colon \{1, 2, \ldots, n\} \to V(G)$ of graph $G$ and outputs a chordal graph $H = H_n$. We put $H_0 = G$ and define $H_i$ to be the graph obtained from $H_{i-1}$ by completing all neighbours $v$ of $\pi(i)$ in $H_{i-1}$ with $\pi^{-1}(v) > i$ into a clique. An elimination ordering $\pi$ is called *minimal* if the corresponding vertex elimination procedure outputs a minimal triangulation of $G$.

**Proposition 2.1** ([44])**.** *Graph $H$ is a minimal triangulation of $G$ if and only if there exists a minimal elimination ordering $\pi$ of $G$ such that the corresponding procedure outputs $H$.*

We will also need the following description of the fill edges introduced by vertex eleminations.

**Proposition 2.2** ([48])**.** *Let $H$ be the chordal graph produced by vertex elimination of graph $G$ according to ordering $\pi$. Then $uv \notin E(G)$ is a fill edge of $H$ if and only if there exists a path $P = uw_1w_2 \ldots w_\ell v$ such that $\pi^{-1}(w_i) < \min(\pi^{-1}(u), \pi^{-1}(v))$ for each $1 \leq i \leq \ell$.*

**Minimal separators.** Let $u$ and $v$ be two non adjacent vertices of a graph $G$. A set of vertices $S \subseteq V$ is an *$u, v$-separator* if $u$ and $v$ are in different connected components of the graph $G[V \setminus S]$. We say that $S$ is a *minimal $u, v$-separator* of $G$ if no proper subset of $S$ is an $u, v$-separator and that $S$ is a *minimal separator* of $G$ if there are two vertices $u$ and $v$ such that $S$ is a minimal $u, v$-separator. Notice that a minimal separator can be contained in another one. If a minimal separator is a clique, we refer to it as to a *clique minimal separator*. A connected component $C$ of $G[V \setminus S]$ is a *full* component associated to $S$ if $N(C) = S$. The following proposition is an exercise in [28].

**Proposition 2.3** (Folklore)**.** *A set $S$ of vertices of $G$ is a minimal $a, b$-separator if and only if $a$ and $b$ 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 $S$.*

**Potential Maximal Cliques** are combinatorial objects which properties are crucial for our algorithm. A vertex set $\Omega$ is defined as a *potential maximal clique* in graph $G$ if there is some minimal triangulation $H$ of $G$ such that $\Omega$ is a maximal clique of $H$. Potential maximal cliques were defined by Bouchitté and Todinca in [8, 9].

The following proposition was proved by Kloks et al. for minimal separators [38] and by Bouchitté and Todinca for potential maximal cliques [8].

**Proposition 2.4** ([8, 38])**.** *Let $X$ be either a potential maximal clique or a minimal separator of $G$, and let $G_X$ be the graph obtained from $G$ by completing $X$ into a clique. Let $C_1, C_2, \ldots, C_r$ be the connected components of $G \setminus X$. Then graph $H$ obtained from $G_X$ by adding a set of fill edges $F$ is a minimal triangulation of $G$ if and only if $F = \bigcup_{i=1}^{r} F_i$, where $F_i$ is the set of fill edges in a minimal triangulation of $G_X[N[C_i]]$.*

The following result about the structure of potential maximal cliques is due to Bouchitté and Todinca.

**Proposition 2.5** ([8])**.** *Let $\Omega \subseteq V$ be a set of vertices of the graph $G$. Let $\{C_1, C_2, \ldots, C_p\}$ be the set of the connected components of $G \backslash \Omega$ and $\{S_1, S_2, \ldots, S_p\}$, where $S_i = N(C_i)$ for $i \in \{1, 2, \ldots, p\}$. Then $\Omega$ is a potential maximal clique of $G$ if and only if:*

1. *$G \setminus \Omega$ has no full component associated to $\Omega$, and*
2. *the graph on the vertex set $\Omega$ obtained from $G[\Omega]$ by completing each $S_i$, $i \in \{1, 2, \ldots, p\}$, into a clique, is a complete graph.*

*Moreover, if $\Omega$ is a potential maximal clique, then $\{S_1, S_2, \ldots, S_p\}$ is the set of minimal separators of $G$ contained in $\Omega$.*

We will need also the following proposition from [23].

**Proposition 2.6** ([23])**.** *Let $\Omega$ be a potential maximal clique of $G$. Then for every $y \in \Omega$, $\Omega = N_G(Y) \cup \{y\}$, where $Y$ is the connected component of $G \setminus (\Omega \setminus \{y\})$ containing $y$.*

A naive approach of deciding if a given vertex subset is a potential maximal clique would be to try all possible minimal triangulations. There is a much faster approach of recognizing potential maximal cliques due to Bouchitté and Todinca based on Proposition 2.5.

**Proposition 2.7** ([8])**.** *There is an algorithm that, given a graph $G = (V, E)$ and a set of vertices $\Omega \subseteq V$, verifies if $\Omega$ is a potential maximal clique of $G$ in time $\mathcal{O}(nm)$.*

**Parameterized complexity.** A parameterized problem $\Pi$ is a subset of $\Gamma^* \times \mathbb{N}$ for some finite alphabet $\Gamma$. An instance of a parameterized problem consists of $(x, k)$, where $k$ is called the parameter. A central notion in parameterized complexity is *fixed parameter tractability (FPT)* which means, for a given instance $(x, k)$, solvability in time $f(k) \cdot p(|x|)$, where $f$ is an arbitrary function of $k$ and $p$ is a polynomial in the input size. We refer to the book of Downey and Fellows [19] for further reading on Parameterized Complexity.
**Kernelization.** A *kernelization algorithm* for a parameterized problem $\Pi \subseteq \Gamma^* \times \mathbb{N}$ is an algorithm that given $(x, k) \in \Gamma^* \times \mathbb{N}$ outputs in time polynomial

in $|x|+k$ a pair $(x',k') \in \Gamma^* \times \mathbb{N}$, called *kernel* such that $(x,k) \in \Pi$ if and only if $(x',k') \in \Pi$, $|x'| \leq g(k)$, and $k' \leq k$, where $g$ is some computable function. The function $g$ is referred to as the size of the kernel. If $g(k) = k^{O(1)}$ then we say that $\Pi$ admits a polynomial kernel.

There are several known polynomial kernels for the MINIMUM FILL-IN problem [36, 37]. The best known kernelization algorithm is due to Natanzon et al. [42, 43], which for a given instance $(G,k)$ outputs in time $\mathcal{O}(k^2nm)$ an instance $(G',k')$ such that $k' \leq k$, $|V(G')| \leq 2k^2+4k$, and $(G,k)$ is a YES instance if and only if $(G',k')$ is.

**Proposition 2.8** ([42, 43])**.** MINIMUM FILL-IN *has a kernel with vertex set of size $\mathcal{O}(k^2)$. The running time of the kernelization algorithm is $\mathcal{O}(k^2nm)$.*

# 3 Branching

In our algorithm we apply branching procedure (Rule 1) whenever it is possible. To describe this rule, we need some definitions. Let $u,v$ be two nonadjacent vertices of $G$, and let $X = N(u) \cap N(v)$ be the common neighborhood of $u$ and $v$. Let also $P = uw_1w_2 \dots w_\ell v$ be a chordless $uv$-path. In other words, any two vertices of $P$ are adjacent if and only if they are consecutive vertices in $P$. We say that *visibility of $X$ from $P$ is obscured* if $|X \setminus N(w_i)| \geq \sqrt{k}$ for every $i \in \{1,\dots,\ell\}$. Thus every internal vertex of $P$ is nonadjacent to at least $\sqrt{k}$ vertices of $X$.

The idea behind branching is based on the observation that every fill-in of $G$ with at most $k$ edges should either contain fill edge $uv$, or should make at least one internal vertex of the path to be adjacent to all vertices of $X$.

**Rule 1** (Branching Rule)**.** *If instance $(G=(V,E),k)$ of* MINIMUM FILL-IN *contains a pair of nonadjacent vertices $u,v \in V$ and a chordless $uv$-path $P = uw_1w_2 \dots w_\ell v$ such that visibility of $X = N(u) \cap N(v)$ from $P$ is obscure, then branch into $\ell+1$ instances $(G_0,k_0),(G_1,k_1),\dots,(G_\ell,k_\ell)$. Here*

- *$G_0 = (V, E \cup \{uv\})$, $k_0 = k-1$;*
- *For $i \in \{1,\dots,\ell\}$, $G_i = (V, E \cup F_i)$, $k_i = k - |F_i|$, where $F_i = \{w_ix | x \in X \wedge w_ix \notin E\}$.*

**Lemma 3.1.** *Rule 1 is sound, i.e. $(G,k)$ is a YES instance if and only if $(G_i,k_i)$ is a YES instance for some $i \in \{0,\dots,\ell\}$.*

*Proof.* If for some $i \in \{0,\dots,\ell\}$, $(G_i,k_i)$ is a YES instance, then $G$ can be turned into a chordal graph by adding at most $k_i + |F_i| = k$ edges, and thus $(G,k)$ is a YES instance.

Let $(G,k)$ be a YES instance, and let $F \subseteq [V]^2$ be such that graph $H = (V, E \cup F)$ is chordal and $|F| \leq k$. By Proposition 2.1, there exists an ordering $\pi$ of $V$, such that the Elimination Game algorithm on $G$ and $\pi$ outputs $H$. Without loss of generality, we can assume that $\pi^{-1}(u) < \pi^{-1}(v)$. If for some

$x \in X$, $\pi^{-1}(x) < \pi^{-1}(u)$, then by Proposition 2.2, we have that $uv \in F$. Also by Proposition 2.2, if $\pi^{-1}(w_i) < \pi^{-1}(u)$ for each $i \in \{1, \ldots, \ell\}$, then again $uv \in F$. In both cases $(G_0, k_0)$ is a YES instance.

The only remaining case is when $\pi^{-1}(u) < \pi^{-1}(x)$ for all $x \in X$, and there is at least one vertex of $P$ placed after $u$ in ordering $\pi$. Let $i \geq 1$ be the smallest index such that $\pi^{-1}(u) < \pi^{-1}(w_i)$. Thus for every $x \in X$, in the path $xuw_1, w_2, \ldots, w_i$ all internal vertices are ordered by $\pi$ before $x$ and $w_i$. By Proposition 2.2, this imply that $w_i$ is adjacent to all vertices of $X$, and hence $(G_i, k_i)$ is a YES instance. □

The following lemma shows that every branching step of Rule 1 can be performed in polynomial time.

**Lemma 3.2.** *Let $(G, k)$ be an instance of* Minimum Fill-in. *It can be identified in time $\mathcal{O}(n^4)$ if there is a pair $u, v \in V(G)$ satisfying conditions of Rule 1. Moreover, if conditions of Rule 1 hold, then a pair $u, v$ of two nonadjacent vertices and a chordless $uv$-path $P$ such that visibility of $N(u) \cap N(v)$ from $P$ is obscured, can be found in time $\mathcal{O}(n^4)$.*

*Proof.* For each pair of nonadjacent vertices $u, v$, we compute $X = N(u) \cap N(v)$. We compute the set of all vertices $W \subseteq V(G) \setminus \{u, v\}$, such that every vertex of $W$ is nonadjacent to at least $\sqrt{k}$ vertices of $X$. Then conditions of Rule 1 do not hold for $u$ and $v$ if in the subgraph $G_{uv}$ induced by $W \cup \{u, v\}$, $u$ and $v$ are in different connected components. If $u$ and $v$ are in the same connected component of $G_{uv}$, then a shortest (in $G_{uv}$) $uv$-path $P$ is a chordless path and the visibility of $X$ from $P$ is obscured. Clearly, all these procedures can be performed in time $\mathcal{O}(n^4)$. □

We say that instance $(G, k)$ is *non-reducible* if conditions of Rule 1 do not hold. Thus for each pair of vertices $u, v$ of non-reducible graph $G$, there is no $uv$-path with obscure visibility of $N(u) \cap N(v)$.

**Lemma 3.3.** *Let $t(n, k)$ be the maximum number of non-reducible problem instances resulting from recursive application of Rule 1 starting from instance $(G, k)$ with $|V(G)| = n$. Then $t(n, k) = n^{\mathcal{O}(\sqrt{k})}$ and all generated non-reducible instances can be enumerated within the same time bound.*

*Proof.* Let us assume that we branch on the instances corresponding to a pair $u, v$ and path $P = uw_1w_2 \ldots w_\ell v$ such that the visibility of $N(u) \cap N(w)$ is obscure from $P$. Then the value of $t(n, k)$ is at most $\sum_{i=0}^{\ell} t(n, k_i)$. Here $k_0 = 1$ and for all $i \geq 1$, $k_i = k - |F_i| \leq k - \sqrt{k}$. Since the number of vertices in $P$ does not exceed $n$, we have that $t(n, k) \leq t(n, k-1) + n \cdot t(n, k - \sqrt{k})$. By making use of standard arguments on the amount of leaves in branching trees (see, for example [34, Theorem 8.1]) it follows that $t(n, k) = n^{\mathcal{O}(\sqrt{k})}$. By Lemma 3.2, every recursive call of the branching algorithm can be done in time $\mathcal{O}(n^4)$, and thus all non-reducible instances are generated in time $\mathcal{O}(n^{\mathcal{O}(\sqrt{k})} \cdot n^4) = n^{\mathcal{O}(\sqrt{k})}$. □

# 4 Listing vital potential maximal cliques

Let $(G, k)$ be a YES instance of MINIMUM FILL-IN. It means that $G$ can be turned into a chordal graph $H$ by adding at most $k$ edges. Every maximal clique in $H$ corresponds to a potential maximal clique of $G$. The observation here is that if a potential maximal clique $\Omega$ needs more than $k$ edges to be added to become a clique, then no solution $H$ can contain $\Omega$ as a maximal clique. In Section 5 we prove that the only potential maximal cliques that are essential for a fill-in with at most $k$ edges are the ones that miss at most $k$ edges from a clique.

A potential maximal clique $\Omega$ is *vital* if the number of edges in $G[\Omega]$ is at least $|\Omega|(|\Omega|-1)/2-k$. In other words, the subgraph induced by vital potential maximal clique can be turned into a complete graph by adding at most $k$ edges. In this section we show that all vital potential maximal cliques of an $n$-vertex non-reducible graph can be enumerated in time $n^{\mathcal{O}(\sqrt{k})}$.

We will first show how to enumerate potential maximal cliques which are, in some sense, almost cliques. This enumeration algorithm will be used as a subroutine to enumerate vital potential maximal cliques. A potential maximal clique $\Omega$ is *quasi-clique*, if there is a set $Z \subseteq \Omega$ of size at most $5\sqrt{k}$ such that $\Omega \setminus Z$ is a clique. In particular, if $|\Omega| \leq 5\sqrt{k}$, then $\Omega$ is also a quasi-clique. The following lemma gives an algorithm enumerating all quasi-cliqes.

**Lemma 4.1.** *Let $(G, k)$ be a problem instance on $n$ vertices. Then all quasi-cliques in $G$ can be enumerated within time $n^{\mathcal{O}(\sqrt{k})}$.*

*Proof.* We will prove that while a quasi-clique can be very large, it can be reconstructed in polynomial time from a small set of $\mathcal{O}(\sqrt{k})$ vertices. Hence all quasi-cliques can be generated by enumerating vertex subsets of size $\mathcal{O}(\sqrt{k})$. Because the amount of vertex subsets of size $\mathcal{O}(\sqrt{k})$ is $n^{\mathcal{O}(\sqrt{k})}$, this will prove the lemma.

Let $\Omega$ be a potential maximal clique which is a quasi-clique, and let $Z \subseteq \Omega$ be the set of size at most $5\sqrt{k}$ such that $X = \Omega \setminus Z$ is a clique. Depending on the amount of full components associated to $X$ in $G \setminus (Z \cup X)$, we consider three cases: There are at least two full components, there is exactly one, and there is no full component.

Consider first the case when $X$ has at least two full components, say $C_1$ and $C_2$. In this case, by Proposition 2.3, $X$ is a minimal clique separator of $G \setminus Z$. Let $H$ be some *minimal* triangulation of $G \setminus Z$. By Proposition 2.4, clique minimal separators remain clique minimal separators in every minimal triangulation. Therefore, $X$ is a minimal separator in $H$. It is well known that every chordal graph has at most $n-1$ minimal separators and that they can be enumerated in linear time [12]. To enumerate quasi-cliques we implement the following algorithm. We construct a minimal triangulation $H$ of $G \setminus Z$. A minimal triangulation can be constructed in time $\mathcal{O}(nm)$ or $\mathcal{O}(n^{\omega} \log n)$, where $\omega < 2.37$ is the exponent of matrix multiplication and $m$ is the number of edges in $G$ [32, 48]. For every minimal separator $S$ of $H$, where $G[S]$ is a clique, we check if $S \cup Z$ is a potential maximal clique in $G$. This can be done in $\mathcal{O}(km)$ time by Proposition 2.7. Therefore, in this case, the time required to enumerate all quasi-cliques of the form $X \cup Z$, up to polynomial multiplicative factor is proportional to the amount of sets $Z$ of size at most $5\sqrt{k}$. The total running time to enumerate quasi-cliques of this type is $n^{\mathcal{O}(\sqrt{k})}$.

Now we consider the case when no full component in $G \setminus (Z \cup X)$ associated to $X$. It means that for every connected component $C$ of $G \setminus (Z \cup X)$, there is $x \in X \setminus N(C)$. By Proposition 2.5, $X$ is also a potential maximal clique in $G \setminus Z$. We construct a minimal triangulation $H$ of $G \setminus Z$. By Proposition 2.4, $X$ is also a potential maximal clique in $H$. By the classical result of Dirac [18] chordal graph $H$ contains at most $n$ maximal cliques and all the maximal cliques of $H$ can be enumerated in linear time [3]. For every maximal clique $K$ of $H$ such that $K$ is also a clique in $G$, we check if $K \cup Z$ is a potential maximal clique in $G$, which can be done in $\mathcal{O}(nm)$ time by Proposition 2.7. As in the previous case, the enumeration of all such quasi-cliques boils down to enumerating sets $Z$, which takes time $n^{\mathcal{O}(\sqrt{k})}$.

Last case, vertex set $X$ has unique full component $C_r$ in $G \setminus (Z \cup X)$ associated to $X$. Since $\Omega = Z \cup X$, we have that each of the connected components $C_1, C_2, \ldots, C_r$ of $G \setminus (Z \cup X)$ is also a connected component of $G \setminus \Omega$. Then for every $i \in \{1, \ldots, r-1\}$, $S_i = N_{G \setminus Z}(C_i)$ is a clique minimal separator in $G \setminus Z$ because $S_i \subseteq X$ is a clique, and $C_i$ together with the component of $G \setminus (Z \cup S_i)$ containing $X \setminus S_i$, are full components associated to $S_i$. Let $H$ be a minimal triangulation of $G \setminus Z$. Vertex set $X$ is a clique in $G \setminus Z$ and thus is a clique in $H$. Let $K$ be a maximal clique of $H$ containing $X$. By Proposition 2.4, for every $i \in \{1, \ldots, r-1\}$, $S_i$ is a minimal separator in $H$. By Proposition 2.4, $G \setminus Z$ has no fill edges between vertices separated by $S_i$ and thus $C_i$ is a connected component of $H \setminus K$.

Because $\Omega$ is a potential maximal clique in $G$, by Proposition 2.5, there is $y \in \Omega$ such that $y \notin N_G(C_r)$. Since $C_r$ is a full component for $X$, we have that $y \in Z$. Moreover, every connected component $C \neq C_r$ of $G \setminus (Z \cup X)$ is also a connected component of $H \setminus K$. Thus every connected component of $H \setminus K$ containing a neighbor of $y$ in $G$ is also a connected component of $G \setminus \Omega$ containing a neighbor of $y$.

Let $B_1, B_2, \ldots, B_\ell$ be the set of connected components in $G \setminus (K \cup Z)$ with $y \in N_G(B_i)$. We define

$$Y = \bigcup_{1 \leq i \leq \ell} B_i \cup \{y\}.$$

By Proposition 2.6, $\Omega = N_G(Y) \cup \{y\}$ and in this case potential maximal clique is characterized by $y$ and $Y$.

To summarize, to enumerate all quasi-cliques corresponding to the last case, we do the following. For every set $Z$ of size at most $5\sqrt{k}$, we construct a minimal triangulation $H$ of $G \setminus Z$. The amount of maximal cliques in a chordal graph $H$ is at most $n$, and for every maximal clique $K$ of $H$ and for every $y \in Z$, we compute the set $Y$. We use Propositions 2.7 to check if $N_G(Y) \cup \{y\}$ is a

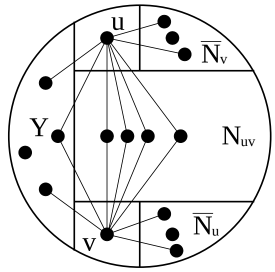


Figure 1: Partitioning of potential maximal clique $\Omega$ into sets $\overline{N}_u, \overline{N}_v, N_{uv}, \{u\}, \{v\}$, and $Y$.

potential maximal clique. The total running time to enumerate quasi-cliques in this case is bounded, up to polynomial factor, by the amount of subsets of size $\mathcal{O}(\sqrt{k})$ in $G$ which is $n^{\mathcal{O}(\sqrt{k})}$. $\square$

Now we are ready to prove the result about vital potential maximal cliques in non-reducible graphs.

**Lemma 4.2.** *Let $(G, k)$ be a non-reducible instance of the problem. All vital potential maximal cliques in $G$ can be enumerated within time $n^{\mathcal{O}(\sqrt{k})}$, where $n$ is the number of vertices in $G$.*

*Proof.* We start by enumerating all vertex subsets of $G$ of size at most $5\sqrt{k}+2$ and apply Proposition 2.7 to check if each such set is a vital potential maximal clique or not.

Let $\Omega$ be a vital potential maximal clique with at least $5\sqrt{k}+3$ vertices and let $Y \subseteq \Omega$ be the set of vertices of $\Omega$ such that each vertex of $Y$ is adjacent in $G$ to at most $|\Omega| - 1 - \sqrt{k}$ vertices of $\Omega$. To turn $\Omega$ into a complete graph, for each vertex $v \in Y$, we have to add at least $\sqrt{k}$ fill edges incident to $v$. Hence $|Y| \le 2\sqrt{k}$. If $\Omega \setminus Y$ is a clique, then $\Omega$ is a quasi-clique. By Lemma 4.1, all quasi-cliques can be enumerated in time $n^{\mathcal{O}(\sqrt{k})}$.

If $\Omega \setminus Y$ is not a clique, there is at least one pair of non-adjacent vertices $u, v \in \Omega \setminus Y$. By Proposition 2.5, there is a connected component $C$ of $G \setminus \Omega$ such that $u, v \in N(C)$.

**Claim 1.** *There is $w \in C$ such that $|\Omega \setminus N(w)| \le 5\sqrt{k}+2$.*

*Proof.* Targeting towards a contradiction, we assume that the claim does not hold. We define the following subsets of $\Omega \setminus Y$.

- $\overline{N}_u \subseteq \Omega \setminus Y$ is the set of vertices which are not adjacent to $u$,
- $\overline{N}_v \subseteq \Omega \setminus Y$ is the set of vertices which are not adjacent to $v$, and
- $N_{uv} = \Omega \setminus (Y \cup \overline{N}_u \cup \overline{N}_v)$ is the set of vertices adjacent to $u$ and to $v$.

See Fig. 1 for an illustration. Let us note that

$$\Omega = \overline{N}_u \cup \overline{N}_v \cup N_{uv} \cup \{u\} \cup \{v\} \cup Y.$$

Since $u, v \notin Y$, we have that $\max\{|\overline{N}_u|, |\overline{N}_v|\} \leq \sqrt{k}$.

We claim that $|N_{uv}| \leq \sqrt{k}$. Targeting towards a contradiction, let us assume that $|N_{uv}| > \sqrt{k}$. By our assumption, every vertex $w \in C$ is not adjacent to at least $5\sqrt{k}+2$ vertices of $\Omega$. Since $|Y \cup \overline{N}_u \cup \overline{N}_v \cup \{u\} \cup \{v\}| \leq 2\sqrt{k}+\sqrt{k}+\sqrt{k}+2 = 4\sqrt{k}+2$, we have that each vertex of $C$ is nonadjacent to at least $\sqrt{k}$ vertices of $N_{uv}$. We take a shortest $uv$-path $P$ with all internal vertices in $C$. Because $C$ is a connected component and $u, v \in N(C)$, such a path exists. Every internal vertex of $P$ is nonadjacent to at least $\sqrt{k}$ vertices of $N_{uv} \subseteq N(u) \cap N(v)$, and thus the visibility of $N_{uv}$ from $P$ is obscured. But this is a contradiction to the assumption that $(G, k)$ is non-reducible. Hence $|N_{uv}| \leq \sqrt{k}$.

Thus if the claim does not hold, we have that

$$|\Omega| = |\overline{N}_u \cup \overline{N}_v \cup N_{uv} \cup \{u\} \cup \{v\} \cup Y| \leq 5\sqrt{k} + 2,$$

but this contradicts to the assumption that $|\Omega| \geq 5\sqrt{k}+3$. This concludes the proof of the claim. ☐

We have shown that for every vital potential maximal clique $\Omega$ of size at least $5\sqrt{k}+3$, there is a connected component $C$ and $w \in C$ such that $|\Omega \setminus N(w)| \leq 5\sqrt{k}+2$. Let $H$ be the graph obtained from $G$ by completing $N(w)$ into a clique. The graph $H[\Omega]$ consist of a clique plus at most $5\sqrt{k}+2$ vertices. We want to show that $\Omega$ is a quasi-clique in $H$, by arguing that $\Omega$ is a potential maximal clique in $H$. Vertex set $\Omega$ is a potential maximal clique in $G$, and thus by Proposition 2.5, there is no full component associated to $\Omega$ in $G \setminus \Omega$. Because $N(w) \cap \Omega \subseteq N(C) \subset \Omega$, there is no full component associated to $\Omega$ in $H$. We use Proposition 2.5 to show that $\Omega$ is a potential maximal clique in $H$ as well. Hence $\Omega$ is a quasi-clique in $H$.

To conclude, we use the following strategy to enumerate all vital potential maximal cliques. We enumerate first all quasi-cliques in $G$ in time $n^{\mathcal{O}(\sqrt{k})}$ by making use of Lemma 4.1, and for each quasi-clique we use Proposition 2.7 to check if it is a vital potential maximal clique. We also try all vertex subsets of size at most $5\sqrt{k}+2$ and check if each of such sets is a vital potential maximal clique. All vital potential maximal cliques which are not enumerated prior to this moment should satisfy the condition of the claim. As we have shown, each such vital potential maximal clique is a quasi-clique in the graph $H$ obtained from $G$ by selecting some vertex $w$ and turning $N_G(w)$ into clique. Thus for every vertex $w$ of $G$, we construct graph $H$ and then use Lemma 4.1 to enumerate all quasi-cliques in $H$. For each quasi-clique of $H$, we use Proposition 2.7 to check if it is a vital potential maximal clique in $G$. The total running time of this procedure is $n^{\mathcal{O}(\sqrt{k})}$. ☐

# 5 Exploring remaining solution space

For an instance $(G, k)$ of Minimum Fill-in, let $\Pi_k$ be the set of all vital potential maximal cliques. In this section, we give an algorithm of running time $\mathcal{O}(nm|\Pi_k|)$, where $n$ is the number of vertices and $m$ the number of edges in $G$. The algorithm receives $(G, k)$ and $\Pi_k$ as an input, and decides if $(G, k)$ is a YES instance. The algorithm is a modification of the algorithm from [23]. The only difference is that the algorithm from [23] computes an optimum triangulation from the set of all potential maximal cliques while here we have to work only with vital potential maximal cliques. For reader's convenience we provide the full proof, but first we need the following lemma.

**Lemma 5.1.** *Let $S$ be a minimal separator in $G$ and let $C$ be a full connected component of $G \setminus S$ associated to $S$. Then every minimal triangulation $H$ of $G_S$ contains a maximal clique $K$ such that $S \subset K \subseteq S \cup C$.*

*Proof.* By Proposition 2.4, $H$ is a minimal triangulation of $G_S$ if and only if $H[S \cup C]$ is a minimal triangulation of $G_S[S \cup C]$. Because $S$ is a clique $G_S[S]$ in, $S$ is a subset of some maximal clique $K$ of $H[S \cup C]$. By definition $K$ is a potential maximal clique in $G_S[S \cup C]$, and by Proposition 2.5 $K$ is a potential maximal clique in $G$. Since $G_S[S \cup C] \setminus S$ has a full component associated to $S$, we have that by Proposition 2.5, $S$ is not a potential maximal clique in $G_S[S \cup C]$ and thus $S \subset K$. ☐

**Lemma 5.2.** *Given a set of all vital potential maximal cliques $\Pi_k$ of $G$, it can be decided in time $\mathcal{O}(nm|\Pi_k|)$ if $(G, k)$ is a YES instance of* Minimum Fill-in.

*Proof.* Let $\mathbf{mfi}(G)$ be the minimum number of fill edges needed to triangulate $G$. Let us remind that by $\mathbf{fill}_G(\Omega)$ we denote the number of non-edges in $G[\Omega]$ and by $G_\Omega$ the graph obtained from $G$ by completing $\Omega$ into a clique. If $\mathbf{mfi}(G) \le k$, then by Proposition 2.4, we have that

$$\mathbf{mfi}(G) = \min_{\Omega \in \Pi_k} [\mathbf{fill}_G(\Omega) + \sum_{C \text{ is a component of } G \setminus \Omega} \mathbf{mfi}(G_\Omega[C \cup N_G(C)])]. \quad (1)$$

Formula (1) can be used to compute $\mathbf{mfi}(G)$, however by making use of this formula we are not able to obtained the claimed running time. To implement the algorithm in time $\mathcal{O}(nm|\Pi_k|)$, we compute $\mathbf{mfi}(G_\Omega[C \cup N_G(C)])]$ by dynamic programming.

By Proposition 2.5, for every connected component $C$ of $G \setminus \Omega$, where $\Omega \in \Pi_k$, $S = N_G(C) \subset \Omega$ is a minimal separator. We define the set $\Delta_k$ as the set of all minimal separators $S$, such that $S = N(C)$ for some connected component $C$ in $G \setminus \Omega$ for some $\Omega \in \Pi_k$. Since for every $\Omega \in \Pi_k$ the number of components in $G \setminus \Omega$ is at most $n$, we have that $|\Delta_k| \le n|\Pi_k|$.

For $S \in \Delta_k$ and a full connected component $C$ of $G \setminus S$ associated to $S$. We define $\Pi_{S,C}$ as the set of potential maximal cliques $\Omega \in \Pi_k$ such that $S \subset \Omega \subseteq S \cup C$. The triple $(S, C, \Omega)$ was called a good triple in [23].

For every $\Omega \in \Pi_k$, connected component $C$ of $G \setminus \Omega$, and $S = N(C)$, we compute $\mathbf{mfi}(F)$, where $F = G_\Omega[C \cup S]$. We start dynamic programming by computing the values for all sets $(S, C)$ such that $\Omega = C \cup S$ is an inclusion-minimal potential maximal clique. In this case we put $\mathbf{mfi}(F) = \mathbf{fill}(C \cup S)$. Observe that $G_S[C \cup S] = G_\Omega[C \cup S]$. Hence by Lemma 5.1, for every minimal triangulation $H$ of $G_S$, there exists a potential maximal clique $\Omega$ in $G$ such that $\Omega$ is a maximal clique in $H$ and $S \subset \Omega \subseteq S \cup C$. Thus $\Omega \in \Pi_{S,C}$. Using this observation, we write the following formula for dynamic programming.

$$\mathbf{mfi}(F) = \min_{\Omega' \in \Pi_{S,C}} [\mathbf{fill}_F(\Omega') + \sum_{C' \text{ is a component of } F \setminus \Omega'} \mathbf{mfi}(F_{\Omega'}[C' \cup N(C')])]. \quad (2)$$

The fact $S \subset \Omega'$ ensures that the solution in (2) can be reconstructed from instances with $|S \cup C|$ of smaller sizes. By (1) and (2) we can decide if there exists a triangulation of $G$ using at most $k$ fill edges, and to construct such a triangulation. It remains to argue for the running time.

Finding connected components in $G \setminus \Omega$ and computing $\mathbf{fill}(\Omega)$ can easily been done in $\mathcal{O}(n + m)$ time. Furthermore, (1) is applied $|\Pi_k|$ times in total. The running time of dynamic programming using (2) is proportional to the amount of states of dynamic programming, which is

$$\sum_{S \in \Delta_k} \sum_{C \in G \setminus S} |\Pi_{S,C}|$$

The graph $G \setminus \Omega$ contains at most $n$ connected components and thus for every minimal separator, each potential maximal clique is counted at most $n$ times, and thus the amount of the elements in the sum does not exceed $n|\Pi_k|$. The total running time is $\mathcal{O}(nm|\Pi_k|)$. ☐

# 6 Putting things together

Now we are in the position to prove the main result of this paper.

**Theorem 6.1.** *The* Minimum Fill-in *problem is solvable in time* $\mathcal{O}(2^{\mathcal{O}(\sqrt{k} \log k)} + k^2 nm)$.

*Proof.* **Step A.** Given instance $(G, k)$ of the Minimum Fill-in problem, we use Proposition 2.8 to obtain a kernel $(G', k')$ on $\mathcal{O}(k^2)$ vertices and with $k' \leq k$. Let us note that $(G, k)$ is a YES instance if and only if $(G', k')$ is a YES instance. This step is performed in time $\mathcal{O}(k^2 nm)$.

**Step B1.** We use Branching Rule 1 on instance $(G', k')$. Since the number of vertices in $G'$ is $\mathcal{O}(k^2)$, we have that by Lemma 3.3, the result of this procedure is the set of $(k^2)^{\mathcal{O}(\sqrt{k})} = 2^{\mathcal{O}(\sqrt{k} \log k)}$ non-reducible instances $(G_1, k_1), \ldots, (G_p, k_p)$. For each $i \in \{1, 2, \ldots, p\}$, graph $G_i$ has $\mathcal{O}(k^2)$ vertices and $k_i \leq k$. Moreover, by Lemma 3.1, $(G', k')$, and thus $(G, k)$, is a YES instance if and only if at least one $(G_i, k_i)$ is a YES instance. By Lemma 3.3, the running time of this step is $2^{\mathcal{O}(\sqrt{k}\log k)}$.

**Step B2.** For each $i \in \{1, 2, \ldots, p\}$, we list all vital potential maximal cliques of graph $G_i$. By Lemma 4.2, the amount of all vital potential maximal cliques in non-reducible graph $G_i$ is $2^{\mathcal{O}(\sqrt{k}\log k)}$ and they can be listed within the same running time.

**Step C.** At this step for each $i \in \{1, 2, \ldots, p\}$, we are given instance $(G_i, k_i)$ together with the set $\Pi_{k_i}$ of vital potential maximal cliques of $G_i$ computed in Step B2. We use Lemma 5.2 to solve the Minimum Fill-in problem for instance $(G_i, k_i)$ in time $\mathcal{O}(k^6|\Pi_{k_i}|) = 2^{\mathcal{O}(\sqrt{k}\log k)}$. If at least one of the instances $(G_i, k_i)$ is a YES instance, then by Lemma 3.1, $(G, k)$ is a YES instance. If all instances $(G_i, k_i)$ are NO instances, we conclude that $(G, k)$ is a NO instance. Since $p = 2^{\mathcal{O}(\sqrt{k}\log k)}$, we have that Step C can be performed in time $2^{\mathcal{O}(\sqrt{k}\log k)}$. The total running time required to perform all steps of the algorithm is $\mathcal{O}(2^{\mathcal{O}(\sqrt{k}\log k)} + k^2nm)$.

$\square$

Let us remark that our decision algorithm can be easily adapted to output the optimum fill-in of size at most $k$.

# 7 Applications to other problems

The algorithm described in the previous section can be modified to solve several related problems. Problems considered in this section are Minimum Chain Completion, Chordal Graph Sandwich, and Triangulating Colored Graph.

**Minimum Chain Completion.** Bipartite graph $G = (V_1, V_2, E)$ is a chain graph if the neighbourhoods of the nodes in $V_1$ forms a chain, that is there is an ordering $v_1, v_2, \ldots, v_{|V_1|}$ of the vertices in $V_1$, such that $N(v_1) \subseteq N(v_2) \subseteq \ldots \subseteq N(v_{|V_1|})$.

| |
|---|
| Minimum Chain Completion<br>*Input:* A bipartite graph $G = (V_1, V_2, E)$ and integer $k \geq 0$.<br>*Question:* Is there $F \subseteq V_1 \times V_2$, $\lvert F\rvert \leq k$, such that graph $H = (V_1, V_2, E \cup F)$ a chain graph? |

Yannakakis in his famous NP-completeness proof of Minimum Fill-in [52] used the following observation. Let $G$ be a bipartite graph with bipartitions $V_1$ and $V_2$, and let $G'$ be cobipartite (the complement of bipartite) graph formed by turning $V_1$ and $V_2$ into cliques. Then $G$ can be transformed into a chain graph by adding $k$ edges if and only if $G'$ can be triangulated with $k$ edges. By Theorem 6.1, we have that Minimum Chain Completion is solvable in $\mathcal{O}(2^{\mathcal{O}(\sqrt{k}\log k)} + k^2nm)$ time.

**Chordal graph sandwich.** In the chordal graph sandwich problem we are given two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ on the same vertex set $V$, and with $E_1 \subset E_2$. The CHORDAL GRAPH SANDWICH problem asks if there exists a chordal graph $H = (V, E_1 \cup F)$ sandwiched in between $G_1$ and $G_2$, that is $E_1 \cup F \subseteq E_2$.

> CHORDAL GRAPH SANDWICH
> *Input:* Two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ such that $E_1 \subset E_2$, and $k = |E_2 \setminus E_1|$.
> *Question:* Is there $F \subseteq E_2 \setminus E_1$, $|F| \leq k$, such that graph $H = (V, E_1 \cup F)$ is a triangulation of $G_1$?

Let us remark that the CHORDAL GRAPH SANDWICH problem is equivalent to asking if there is a minimal triangulation of $G_1$ sandwiched between $G_1$ and $G_2$.

To solve CHORDAL GRAPH SANDWICH we cannot use the algorithm from Section 6.1 directly. The reason is that we are only allowed to add edges from $E_2$ as fill edges. We need a kernelization algorithm for this problem as well.

**Lemma 7.1.** CHORDAL GRAPH SANDWICH *has a kernel with vertex set of size $\mathcal{O}(k^3)$. The running time of kernelization algorithm is $\mathcal{O}(k^2 nm)$.*

*Proof.* To simplify notations, we denote an instance of the CHORDAL GRAPH SANDWICH problem by $(G, E', k)$, where $G_1 = G = (V, E)$, $E' \cap E = \emptyset$, $k \leq |E'|$, and $G_2 = (V, E \cup E')$.

We first define two reduction rules and prove their correctness.

**No-cycle-vertex Rule.** *If instance $(G, E', k)$ has $u \in V$ such that for each connected component $C$ of $G[V \setminus N[u]]$, $N(C)$ is a clique, then replace instance $(G, E', k)$ with instance $(G \setminus \{u\}, E'', k')$, where $E'' \subseteq E'$ are the edges not incident to $u$ and $k' = k - |E' \setminus E''|$.*

**Claim 2.** *No-cycle-vertex Rule is sound, i.e. $(G \setminus \{u\}, E'', k')$ is an YES instance of the chordal sandwich problem if and only if $(G, E', k)$ is an YES instance.*

*Proof.* Let $G_u$ be the graph $G[V \setminus \{u\}]$. Chordallity is a hereditary property so if $H = (V, E \cup F)$ is a triangulation of $G$ where $|F| \leq k$, then $H[V \setminus \{u\}]$ is a triangulation of $G_u$ where the set of fill edges are the edges of $F$ not incident to $u$.

For the opposite direction assume that $H_u = (V(G_u), E(G_u) \cup F_u)$ is a minimal triangulation of $G_u$ where $|F_u| \leq k$. Our objective now is to argue that $H = (V, E \cup F_u)$ is a triangulation of $G$. Targeting towards a contradiction, let us assume that $W = uaw_1w_2 \dots w_\ell bu$ is a chordless cycle of length at least four in $H$. Then notice that $u$ is a vertex of $W$ as $H_u = H[V \setminus \{u\}]$ is a chordal graph by our assumption. Vertices $a$ and $b$ of $W$ are not adjacent by definition, and let vertices $w_1w_2 \dots w_\ell$ be contained in the connected component $C$ of $G[V \setminus N[u]$.

Nonadjacent vertices $a, b$ are now contained in $N(C)$ which is a contradiction to the condition of applying No-cycle-vertex Rule. $\square$

For each pair of nonadjacent vertices $x, y \in V$, we define $A_{x,y}$ as the set of vertices such that $w \in A_{x,y}$ if $x, y \in N_G(w)$ and vertices $x, y$ are contained in the same connected component of $G[(V \setminus N[w]) \cup \{x, y\}]$.

**Safe-edge Rule.** *If $2k < |A_{x,y}|$ for some pair of vertices $x, y$ in a problem instance $(G, E', k)$ then*

- *replace instance $(G, E', k)$ with a trivial NO instance if $xy \notin E'$, and*
- *otherwise with instance $(G = (V, E \cup \{xy\}), E' \setminus \{xy\}, k - 1)$.*

**Claim 3.** *Safe-edge Rule is sound, i.e. the instance outputted by the rule is a YES instance of the chordal sandwich problem if and only if $(G, E', k)$ is an YES instance.*

*Proof.* By the definition of $A_{x,y}$, it is not hard to see that there exists an induced cycle of length at least four consisting of $x, w, y$ and a shortest induced path from $x$ to $y$ in $G[(V \setminus N[w]) \cup \{x, y\}]$. A trivial observation is that every triangulation of $G$ either has $xy$ as a fill edge, or there exists a fill edge incident to $w$. Since $2k < |A_{xy}|$, we have that every minimal triangulation not using $xy$ as a fill edge has at least one fill edge incident to each vertices in $A_{uv}$, and thus $(G, E', k)$ is a NO instance if $xy \notin E'$ and $xy \in F$ for every edge set $F \subseteq E'$ such that $H = (V, E \cup F)$ is chordal. $\square$

It is possible to show that exhausting application of both No-cycle-vertex and Safe-edge rules either results in a polynomial kernel or a trivial recognizable NO instance. However, the running time of the algorithm would be $\mathcal{O}(kn^2m)$. In what follow, we show that with much more careful implementation of rules, it is possible to obtain a kernel of size $\mathcal{O}(k^3)$ in time $\mathcal{O}(k^2nm)$. The algorithm uses the same approach as the kernel algorithms given in [37] and [43] for Minimum Fill-in.

Let $(G, E', k)$ be an instance of the problem. We give an algorithm with running time $\mathcal{O}(k^2nm)$ that outputs an instance $(G', E'', k')$ such that $|V(G')| \leq 32k^3 + 4k$ and $(G', E'', k')$ is a YES instance if and only if $(G, E', k)$ is a YES instance. Let us remark that we put no efforts to optimize the size of the kernel.

Let $A, B$ be a partitioning of the vertices of graph $G$ in the given instance $(G, E', k)$. Initially we put $A = \emptyset$ and $B = V(G)$. There is a sequence of procedures. To avoid a confusing nesting of if-then-else statements, we use the following convention: The first case which applies is used first in the procedure. Thus, inside a given procedure, the hypotheses of all previous procedures are assumed false.

P1: If $4k < |A|$ then return a trivial NO instance, else if $G[B]$ contains a chordless cycle $W$ of length at least four, move $V(W)$ from $B$ to $A$.

P2: If $4k < |A|$ then return a trivial NO instance, else if $G[B]$ contains a chordless path $P$ of at least two vertices which is also an induced subgraph of a chordless cycle $W$ of $G$ of length at least four, move $V(P)$ from $B$ to $A$.

P3: Compute $A_{x,y}$ for each pair of vertices in $A$.

P4: If $|A_{x,y}| \leq 2k$ then move $A_{x,y}$ from $B$ to $A$, else if $xy \notin E'$ return a trivial NO instance, else add edge $xy$ to $F$.

P5: Delete every vertex of $B$.

Now we argue on the correctness and running time required to implement the procedures. P1: Every cordless cycle of length $4 \leq \ell$ requires at least $\ell - 3$ fill edges to be triangulated [37]. Thus, at least one fill edge has to be added for each 4 vertices moved to $A$, and we have a NO instance if $4k < |A|$. A chordless cycle of a non chordal graph can be obtained in $\mathcal{O}(n+m)$ time [49]. Total running time is $\mathcal{O}(k(n+m))$.

P2: Let $P$ be an induced path which is an induced subgraph of a chordless cycle $W$ in a graph $G$. Then any triangulation of $G$ will add at least $|V(P)|-1$ fill edges incident to vertices in $P$ [37]. Thus, 2 end points of a fill edge(equivalently to one fill edge) has to be added for each 4 vertices moved to $A$, and we have a NO instance if $4k < |A|$. A path $P$ can be obtained as follows: Let $u$ be an end vertex of $P$ and let $x$ be the unique neighbour of $u$ in $A$ on the cycle $W$. We have a chordless cycle $W$ satisfying the conditions if there is a path from $(N(u) \cap B) \setminus N(x)$ to a vertex in $N(x) \setminus N(u)$ not using vertices of $N(x) \cap N(u)$. Such a path can trivially be found in $\mathcal{O}(m)$ time. Thus the time required to this step is $\mathcal{O}(knm)$.

P3: Let $x, y$ be nonadjacent vertices of $A$. For each vertex $w \in N(x) \cap N(y) \cap B$ check if there is a path from $x$ to $y$ avoiding $N[w]$, if so there is a chordless cycle containing $xwy$ as consecutive vertices. Running time is $\mathcal{O}(k^2nm)$ because $|A|$ is $O(k)$.

P4: If $|A_{x,y}| > 2k$ then by the Safe-edge Rule, it is safe to add edge $xy$ if $xy \in E'$, and to return a trivial NO instance if $xy \notin E'$. Running time for this step is $\mathcal{O}(k^2n)$.

P5: For every remaining vertex $u \in B$, No-cycle-vertex Rule can be applied, and vertex $u$ can safely be deleted from the graph $M = (V, E \cup F)$. Let us on the contrary assume that this is not the case. Then one can find two nonadjacent vertices $a, b \in N_M(u)$ and connected component $C$ of $M \setminus N_M[w]$ such that $a, b \in N_M(C)$. Let $W$ be the chordless cycle of length at least four, obtained by $yux$ and a shortest chordless path from $x$ to $y$ in $M[C \cup \{x, y\}]$. By P1, at least one vertex of $W$ is contained in $A$ and by P2, no two consecutive vertices of $W$ are contained in $B$. By our assumption $u \in B$, so $x, y \in A$ by P2. Vertex $u$ is contained in $N(x) \cap N(y)$ because during P3 we add edges only between vertices in $A$. Furthermore, $u \in A_{x,y}$ because otherwise by [37, Theorem 2.10], $u$ would already have been moved to $A$ by P2. Since $xy \notin F$ and $u \in A_{x,y} \cap B$, we have a contradiction to P4.

Let $G' = (A, E(G[A] \cup F)$ and let $E''$ be the edges in $E' \setminus F$ where both endpoints are incident to vertices of $A$. Since $(G', E'', k - |F|)$ is obtained by applying Safe-edge Rule on each edge in $F$ and No-cycle-vertex Rule on each vertex in $B$, we have that instance $(G', E'', k - |F|)$ is a YES instance if and only if $(G, E', k)$ is a YES instance, Finally, $|A| \leq 32k^3 + 4k$ because during P1 and P2 we move in total at most $4k$ vertices from $B$ to $A$, and during P3 at most $2k \cdot (4k)^2$ vertices to $A$.

$\square$

**Theorem 7.2.** Chordal Sandwich Problem *is solvable in time* $\mathcal{O}(2^{\mathcal{O}(\sqrt{k}\log k)} + k^2nm)$.

*Proof.* Let $(G_1, G_2, k)$ be an instance of the problem. We sketch the proof by following the steps of the proof of Theorem 6.1 and commenting on the differences. *Step A:* We use Lemma 7.1, to obtain in time $\mathcal{O}(k^2nm)$ kernel $(G_1', G_2', k')$ such that $|V(G_1')| = \mathcal{O}(k^3)$ and $k' \leq k$. *Step B1:* On kernel we use Branching Rule 1 exhaustively, with the adaptation that every instance defined by fill edge set $F_i$ where $F_i \not\subseteq E_2$ is discarded. Thus we obtain $2^{\mathcal{O}(\sqrt{k}\log k)}$ non-reducible instances. *Step B2:* For each non-reducible instance $(G_1^i, G_2^i, k_i)$, we enumerate vital potential maximal cliques of $G_1^i$ but discard all potential maximal cliques that are not cliques in $G_2^i$. *Step C:* Solve the remaining problem in time proportional to the number of vital potential maximal cliques in $G_1'$ that are also cliques in $G_2^i$. This step is almost identical to Step C of Theorem 6.1. $\square$

**Triangulating Colored Graph.** In the Triangulating Colored Graph problem we are given a graph $G = (V, E)$ with a partitioning of $V$ into sets $V_1, V_2, \ldots, V_c$, a colouring of the vertices. Let us remark that this coloring is not necessarily a proper coloring of $G$. The question is if $G$ can be triangulated without adding edges between vertices in the same set (colour).

Triangulating Colored Graph
*Input:* A graph $G = (V, E)$, a partitioning of $V$ into sets $V_1, V_2, \ldots, V_c$, and an integer $k$.
*Question:* Is there $F \subseteq [V]^2$, $|F| \leq k$, and such that for each $uv \in F$, $u, v \notin V_i$, $1 \leq i \leq c$, and graph $H = (V, E_1 \cup F)$ is a triangulation of $G$?

Triangulating Colored Graph can be trivially reduced to Chordal Graph Sandwich by defining $G_1 = G$, and the edge set of graph $G_2$ as the edge set of $G_1$ plus the set of all vertex pairs of different colors. Thus by Theorem 7.2, Triangulating Colored Graph is solvable in time $\mathcal{O}(2^{\mathcal{O}(\sqrt{k}\log k)} + k^2nm)$.

# 8 Conclusions and open problems

In this paper we gave the first parameterized subexponential time algorithm solving Minimum Fill-in in time $\mathcal{O}(2^{\mathcal{O}(\sqrt{k}\log k)} + k^2nm)$. It would be interesting to find out how tight is the exponential dependence, up to some complexity assumption, in the running time of our algorithm. We would be surprised to hear about time $2^{o(\sqrt{k})}n^{\mathcal{O}(1)}$ algorithm solving Minimum Fill-in. For example, such an algorithm would be able to solve the problem in time $2^{o(n)}$. However, the only results we are aware in this direction is that Minimum Fill-in cannot be solved in time $2^{o(k^{1/6})}n^{\mathcal{O}(1)}$ unless the ETH fails [14]. See [33] for more information on ETH. Similar uncertainty occurs with a number of other graph problems expressible in terms of vertex orderings. Is it possible to prove that unless ETH fails, there are no $2^{o(n)}$ algorithms for Treewidth, Minimum Interval Completion, and Optimum Linear Arrangement? Here the big gap between what we suspect and what we know is frustrating.

On the other hand, for the Triangulated Colored Graph problem, which we are able to solve in time $\mathcal{O}(2^{\mathcal{O}(\sqrt{k}\log k)} + k^2nm)$, Bodlaender et al. [6] gave a polynomial time reduction that from a 3-SAT formula on $p$ variables and $q$ clauses constructs an instance of Triangulated Colored Graph. This instance has $2 + 2p + 6q$ vertices and a triangulation of the instance respecting its colouring can be obtained by adding of at most $(p + 3q) + (p + 3q)^2 + 3pq$ edges. Thus up to ETH, Triangulated Colored Graph and Chordal Graph Sandwich cannot be solved in time $2^{o(\sqrt{k})}n^{\mathcal{O}(1)}$.

The possibility of improving the $nm$ factor in the running time $\mathcal{O}(2^{\mathcal{O}(\sqrt{k}\log k)}+k^2nm)$ of the algorithm is another interesting open question. The factor $nm$ appears from the running time required by the kernelization algorithm to identify simplicial vertices. Identification of simplicial vertices can be done in time $\mathcal{O}(\min\{mn, n^{\omega}\log n\})$, where $\omega < 2.37$ is the exponent of matrix multiplication [32, 39]. Is the running time required to obtain a polynomial kernel for Minimum Fill-in is at least the time required to identify a simplicial vertex in a graph and can search of a simplicial vertex be done faster than finding a triangle in a graph?

Finally, there are various problems in graph algorithms, where the task is to find a minimum number of edges or vertices to be changed such that the resulting graph belongs to some graph class. For example, the problems of completion to interval and proper interval graphs are fixed parameter tractable [31, 36, 37, 50]. Can these problems be solved by subexponential parameterized algorithms? Are there any generic arguments explaining why some FPT graph modification problems can be solved in subexponential time and some can't?

**Acknowledgement** We are grateful to Marek Cygan and Saket Saurabh for discussions and useful suggestions.

# References

[1] N. Alon, D. Lokshtanov, and S. Saurabh, *Fast FAST*, in ICALP, vol. 5555 of LNCS, Springer, 2009, pp. 49–58.

[2] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis, *On the desirability of acyclic database schemes*, J. ACM, 30 (1983), pp. 479–513.

[3] J. R. S. Blair and B. W. Peyton, *An introduction to chordal graphs and clique trees*, in Graph Theory and Sparse Matrix Computations, Springer, 1993, pp. 1–30. IMA Volumes in Mathematics and its Applications, Vol. 56.

[4] H. Bodlaender, P. Heggernes, and Y. Villanger, *Faster parameterized algorithms for minimum fill-in*, Algorithmica, (to appear).

[5] H. L. Bodlaender, M. R. Fellows, M. T. Hallett, T. Wareham, and T. Warnow, *The hardness of perfect phylogeny, feasible register assignment and other problems on thin colored graphs*, Theor. Comput. Sci., 244 (2000), pp. 167–188.

[6] H. L. Bodlaender, M. R. Fellows, and T. Warnow, *Two strikes against perfect phylogeny*, in ICALP, vol. 623 of LNCS, Springer, 1992, pp. 273–283.

[7] M. L. Bonet, C. A. Phillips, T. Warnow, and S. Yooseph, *Constructing evolutionary trees in the presence of polymorphic characters*, in STOC, ACM, 1996, pp. 220–229.

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

[9] ———, *Listing all potential maximal cliques of a graph*, Theor. Comput. Sci., 276 (2002), pp. 17–32.

[10] P. Buneman, *A characterisation of rigid circuit graphs*, Discrete Math., 9 (1974), pp. 205–212.

[11] L. Cai, *Fixed-parameter tractability of graph modification problems for hereditary properties*, Inf. Process. Lett., 58 (1996), pp. 171–176.

[12] L. S. Chandran and F. Grandoni, *A linear time algorithm to list the minimal separators of chordal graphs*, Discrete Mathematics, 306 (2006), pp. 351–358.

[13] F. R. K. Chung and D. Mumford, *Chordal completions of planar graphs*, J. Comb. Theory, Ser. B, 62 (1994), pp. 96–106.

[14] M. Cygan, *Private communication*. 2011.

[15] T. A. Davis, *Direct methods for sparse linear systems*, vol. 2 of Fundamentals of Algorithms, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006.

[16] E. D. Demaine, F. V. Fomin, M. Hajiaghayi, and D. M. Thilikos, *Subexponential parameterized algorithms on graphs of bounded genus and $H$-minor-free graphs*, J. Assoc. Comput. Mach., 52 (2005), pp. 866–893.

[17] R. Diestel, *Graph theory*, vol. 173 of Graduate Texts in Mathematics, Springer-Verlag, Berlin, third ed., 2005.

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

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

[20] T. Feder, H. Mannila, and E. Terzi, *Approximating the minimum chain completion problem*, Inf. Process. Lett., 109 (2009), pp. 980–985.

[21] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, Berlin, 2006.

[22] F. V. Fomin and D. Kratsch, *Exact Exponential Algorithms*, An EATCS Series: Texts in Theoretical Computer Science, Springer, 2010.

[23] F. V. Fomin, D. Kratsch, I. Todinca, and Y. Villanger, *Exact algorithms for treewidth and minimum fill-in*, SIAM J. Comput., 38 (2008), pp. 1058–1079.

[24] F. V. Fomin and Y. Villanger, *Finding induced subgraphs via minimal triangulations*, in STACS, vol. 5 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010, pp. 383–394.

[25] D. R. Fulkerson and O. A. Gross, *Incidence matrices and interval graphs*, Pacific J. Math., 15 (1965), pp. 835–855.

[26] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, 1979.

[27] D. Geman, *Random fields and inverse problems in imaging*, in École d'été de Probabilités de Saint-Flour XVIII—1988, vol. 1427 of Lecture Notes in Math., Springer, Berlin, 1990, pp. 113–193.

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

[29] M. C. Golumbic, H. Kaplan, and R. Shamir, *Graph sandwich problems*, J. Algorithms, 19 (1995), pp. 449–473.

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

[31] P. Heggernes, C. Paul, J. A. Telle, and Y. Villanger, *Interval completion with few edges*, in STOC, 2007, pp. 374–381.

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

[33] R. Impagliazzo, R. Paturi, and F. Zane, *Which problems have strongly exponential complexity?*, J. Comput. Syst. Sci., 63 (2001), pp. 512–530.

[34] M. Jurdzinski, M. Paterson, and U. Zwick, *A deterministic subexponential algorithm for solving parity games*, SIAM J. Comput., 38 (2008), pp. 1519–1532.

[35] S. Kannan and T. Warnow, *Triangulating 3-colored graphs*, SIAM J. Discrete Math., 5 (1992), pp. 249–258.

[36] 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 FOCS, IEEE, 1994, pp. 780–791.

[37] H. Kaplan, R. Shamir, and R. E. Tarjan, *Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs*, SIAM J. Comput., 28 (1999), pp. 1906–1922.

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

[39] D. Kratsch and J. Spinrad, *Between $O(nm)$ and $O(n^\alpha)$*, SIAM J. Comput., 36 (2006), pp. 310–325.

[40] S. L. Lauritzen and D. J. Spiegelhalter, *Local computations with probabilities on graphical structures and their application to expert systems*, J. Royal Statist. Soc. B, 50 (1988), pp. 157–224.

[41] H. Mannila and E. Terzi, *Nestedness and segmented nestedness*, in KDD '07, New York, NY, USA, 2007, ACM, pp. 480–489.

[42] A. Natanzon, R. Shamir, and R. Sharan, *A polynomial approximation algorithm for the minimum fill-in problem*, in STOC, 1998, pp. 41–47.

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

[44] T. Ohtsuki, L. K. Cheung, and T. Fujisawa, *Minimal triangulation of a graph and optimal pivoting ordering in a sparse matrix*, J. Math. Anal. Appl., 54 (1976), pp. 622–633.

[45] S. Parter, *The use of linear graphs in Gauss elimination*, SIAM Review, 3 (1961), pp. 119–130.

[46] B. Patterson and W. Atmar, *Nested subsets and the structure of insular mammalian faunas and archipelagos*, Biological Journal of the Linnean Society, 28 (1986).

[47] D. J. Rose, *A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations*, in Graph Theory and Computing, R. C. Read, ed., Academic Press, New York, 1972, pp. 183–217.

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

[49] 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.

[50] Y. Villanger, P. Heggernes, C. Paul, and J. A. Telle, *Interval completion is fixed parameter tractable*, SIAM J. Comput., 38 (2009), pp. 2007–2020.

[51] S. Wong, D. Wu, and C. Butz, *Triangulation of bayesian networks: A relational database perspective*, in Rough Sets and Current Trends in Computing, vol. 2475 of LNCS, Springer, 2002, pp. 950–950.

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