Finding Optimal Triangulations Parameterized by Edge Clique Cover

Korhonen, Tuukka

2022-08

Korhonen , T 2022 , ' Finding Optimal Triangulations Parameterized by Edge Clique Cover ' , Algorithmica , vol. 84 , no. 8 , pp. 2242-2270 . https://doi.org/10.1007/s00453-022-00932-0

http://hdl.handle.net/10138/355741
https://doi.org/10.1007/s00453-022-00932-0

Algorithmica (2022) 84:2242–2270
https://doi.org/10.1007/s00453-022-00932-0

Finding Optimal Triangulations Parameterized by Edge Clique Cover

Tuukka Korhonen[1]

Received: 28 December 2020 / Accepted: 8 January 2022 / Published online: 5 February 2022

Abstract

We consider problems that can be formulated as a task of finding an optimal triangulation of a graph w.r.t. some notion of optimality. We present algorithms parameterized by the size of a minimum edge clique cover ($\mathtt{cc}$) to such problems. This parameterization occurs naturally in many problems in this setting, e.g., in the perfect phylogeny problem $\mathtt{cc}$ is at most the number of taxa, in fractional hypertreewidth $\mathtt{cc}$ is at most the number of hyperedges, and in treewidth of Bayesian networks $\mathtt{cc}$ is at most the number of non-root nodes. We show that the number of minimal separators of graphs is at most $2^{\mathtt{cc}}$, the number of potential maximal cliques is at most $3^{\mathtt{cc}}$, and these objects can be listed in times $O^*(2^{\mathtt{cc}})$ and $O^*(3^{\mathtt{cc}})$, respectively, even when no edge clique cover is given as input; the $O^*(\cdot)$ notation omits factors polynomial in the input size. These enumeration algorithms imply $O^*(3^{\mathtt{cc}})$ time algorithms for problems such as treewidth, weighted minimum fill-in, and feedback vertex set. For generalized and fractional hypertreewidth we give $O^*(4^m)$ time and $O^*(3^m)$ time algorithms, respectively, where m is the number of hyperedges. When an edge clique cover of size $\mathtt{cc}'$ is given as a part of the input we give $O^*(2^{\mathtt{cc}'})$ time algorithms for treewidth, minimum fill-in, and chordal sandwich. This implies an $O^*(2^n)$ time algorithm for perfect phylogeny, where n is the number of taxa. We also give polynomial space algorithms with time complexities $O^*(9^{\mathtt{cc}'})$ and $O^*(9^{\mathtt{cc}+O(\log^2 \mathtt{cc})})$ for problems in this framework.

Keywords Parameterized algorithms · Potential maximal cliques · Edge clique cover · Treewidth · Minimum fill-in · Fractional hypertreewidth

This work has been financially supported by Academy of Finland (Grant 322869).

✉ Tuukka Korhonen
tuukka.korhonen@uib.no

[1] Department of Computer Science, University of Helsinki, Helsinki, Finland

1 Introduction

A graph is *chordal* if it has no induced cycle of length at least four. A *triangulation* of a graph G is a chordal supergraph of G on the same vertex set. Many graph problems can be formulated as a problem of finding an optimal triangulation of the graph with respect to some notion of optimality. For example, computing the treewidth of a graph corresponds to finding a triangulation with the smallest size of a maximum clique, and computing the minimum fill-in corresponds to finding a triangulation with the least number of edges.

An *edge clique cover* of a graph G is a set of cliques of G such that each edge of G is contained in at least one of the cliques. In this article, we give fixed-parameter algorithms for optimal triangulation problems parameterized by the size of a minimum edge clique cover of the graph, denoted by $\mathtt{cc}$, and by the size of an edge clique cover given as an input, denoted by $\mathtt{cc}'$. Our algorithms are based on the framework of potential maximal cliques [5,15,17,21,35].

A *minimal triangulation* of a graph G is a triangulation of G that has no subgraph that is a triangulation of G. A *potential maximal clique* (PMC) of a graph G is a set of vertices $\Omega \subseteq V(G)$ such that there exists a minimal triangulation H of G where Ω is a maximal clique. By the results of Bouchitté and Todinca [5] and Fomin, Kratsch, Todinca, and Villanger [15], a large class of optimal triangulation problems can be solved by first enumerating all PMCs of the graph and then performing dynamic programming over them, with time complexity that is linear in the number of PMCs and polynomial in the size of the graph. Therefore the main focus of this article is on bounding the number of PMCs based on edge clique cover and giving a corresponding enumeration algorithm.

1.1 Interpretations of $\mathtt{cc}$

While in general the parameter $\mathtt{cc}$ could be considered non-standard, it has natural interpretations in at least three settings in which algorithms for finding optimal triangulations are applied: width parameters of hypergraphs, phylogenetics, and probabilistic inference. The reason that $\mathtt{cc}$ is a natural choice in these settings is that the input graph is constructed as a union of cliques, with the goal of each clique W representing a constraint of type "the triangulation must contain a maximal clique Ω with $W \subseteq \Omega$". Maximal cliques of a triangulation in turn correspond to bags of a tree decomposition. Next we discuss these three settings in more detail. The discussion is summarized in Table 1.

A *hypergraph* is a structure like a graph, but instead of edges that are sets of two vertices, a hypergraph has hyperedges that are arbitrary subsets of the vertex set. Width parameters of hypergraphs, including primal treewidth, generalized hypertreewidth, and fractional hypertreewidth are central structural parameters of constraint satisfaction problems (CSPs) [19,20]. In particular, they measure how well a hypergraph can be decomposed by cuts whose intersection with a solution has a simple characterization, allowing dynamic programming. The computation of each of these parameters can be formulated as an optimal triangulation problem on the *primal graph* of the hypergraph

Table 1 Relationships between the parameter minimum edge clique cover (cc) and natural parameters in optimal triangulation problems

Problem	Parameter	Relation
Computing width parameters of hypergraphs: primal treewidth, generalized hypertreewidth, and fractional hypertreewidth	Number of hyperedges m	$\mathtt{cc} \leq m$
Perfect phylogeny problem and its optimization variant	Number of taxa n	$\mathtt{cc} \leq n$
Computing treewidth of a moral graph of Bayesian network	Number of non-root nodes n'	$\mathtt{cc} \leq n'$

[35]. The primal graph of a hypergraph is a graph constructed by inducing a clique on each hyperedge, and therefore the size of its minimum edge clique cover is at most m, the number of hyperedges. For example, the primal treewidth of a hypergraph is the treewidth of the primal graph. Generalized and fractional hypertreewidth are more general parameters, but their definitions are technical and postponed to Sect. 2.3.

In phylogenetics a central problem is to construct an evolutionary tree of a set of n taxa (i.e. species) based on k characters (i.e. attributes) describing them [39]. For example, when the character data is drawn from molecular sequences, the number of characters k can be much larger than the number of taxa n [27]. In the weighted minimum fill-in problem the input is a graph G and a weight function on pairs of vertices of G, and the task is to find a triangulation H of G so that the total weight of edges in $E(H)\setminus E(G)$ is minimized. Deciding if the taxa admit a perfect phylogeny and an optimization version of it can be reduced to the weighted minimum fill-in problem on the partition intersection graph of the characters [4,21,39]. The partition intersection graph has a vertex for each character-state pair, and its edges are constructed by inducing a clique corresponding to each taxon [39]. Hence the size of its minimum edge clique cover is at most n, the number of taxa.

A third setting in which parameterization by cc is motivated is probabilistic inference. Given a Bayesian network, the first step of efficient probabilistic inference algorithms is to compute a tree decomposition of small width of the moral graph of the Bayesian network [25,26]. The moral graph is constructed as a union of n' cliques, where n' is the number of non-root nodes of the Bayesian network [26], and thus the size of its minimum edge clique cover is at most n'.

1.2 Connections to Practice

This article is also directly motivated by observations in practice. Starting from the Second Parameterized Algorithms and Computational Experiments challenge (PACE 2017) [11], algorithm implementations based on potential maximal cliques have been observed to outperform other exact algorithm implementations on problems formulated as finding optimal triangulations [28,30,31,36,41,42]. In particular, this work is motivated by experimental observations of the usefulness of potential maximal cliques in computing hypergraph parameters [28,30] and in phylogenetics [28,31].

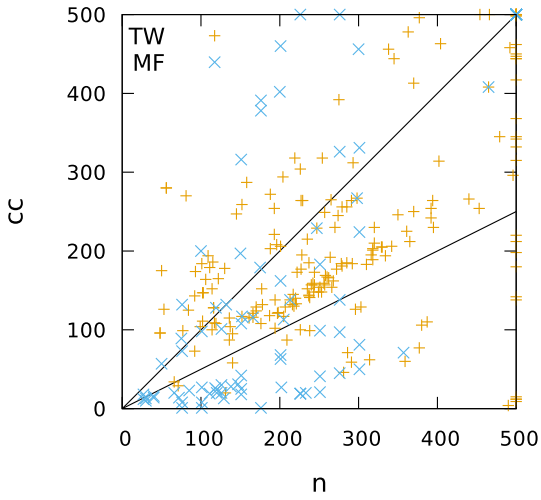

Fig. 1 Number of vertices (n) and minimum edge clique cover ($\mathtt{cc}$) of PACE 2017 treewidth (TW) and minimum fill-in (MF) instances. Both values are truncated from above at 500 and the lines $n = \mathtt{cc}$ and $n/2 = \mathtt{cc}$ are shown

Our parameterization can be justified by real-world instances with small edge clique covers. In the context of hypergraph parameters, 708 of the 3072 hypergraphs in the standard HyperBench library [12] have $m < n/2$, where n is the number of vertices and m is the number of hyperedges. In the context of phylogenetics, an instance describing mammal mitochondrial sequences [22] has 7 taxa while its partition intersection graph has 245 vertices and an instance describing Indo-European languages [37] has 24 taxa while its partition intersection graph has 864 vertices. In the context of Bayesian networks, the Bayesian network "Andes" [9], accessed from the standard BNlearn repository [38], has 223 nodes of which 134 are non-root.

To study the exact values of $\mathtt{cc}$ on well-known treewidth and minimum fill-in benchmarks, we computed minimum edge clique covers of graphs from PACE 2017 [11], which had tracks for both treewidth and minimum fill-in. The treewidth track had 200 available test instances and the minimum fill-in track 100. All of the instances are based on real-world applications [11]. We attempted to compute minimum edge clique covers of all of the 300 instances using Bron–Kerbosch algorithm [7] for maximal clique enumeration and CPLEX [24] for solving the resulting set cover problem. We managed to compute the minimum edge clique cover of 294 of the instances. Figure 1 shows the relations of minimum edge clique cover and the number of vertices in these instances. Most of the instances have minimum edge clique cover between $n/2$ and n, where n is the number of vertices. In 75 of the instances $\mathtt{cc} < n/2$, which is roughly the asymptotic threshold where our bounds for minimal separators and potential maximal cliques are better than known bounds by n [17].

1.3 Techniques

The algorithms that we design in this article are based on the framework of potential maximal cliques (PMCs) [5,15]. Algorithms in this framework typically consist of two phases. In the first phase the set $\Pi(G)$ of PMCs of the input graph G is enumerated,

and in the second phase dynamic programming over the PMCs is performed in time $O^*(|\Pi(G)|)$. The second phase of the PMC framework has already been formulated for all of the problems that we consider [15,17,18,21,35], so our $O^*(3^{\mathtt{cc}})$ time algorithms follow from an $O^*(3^{\mathtt{cc}})$ time PMC enumeration algorithm that we give. This algorithm is based on the Bouchitté–Todinca algorithm [6]. We achieve the $O^*(3^{\mathtt{cc}})$ bound by novel characterizations of minimal separators and PMCs with respect to an edge clique cover. In particular, we show that minimal separators correspond to bipartitions of an edge clique cover and almost all potential maximal cliques correspond to tripartitions of an edge clique cover.

On some of the problems we improve the time complexity to $O^*(2^{\mathtt{cc}'})$, where $\mathtt{cc}'$ is the size of an edge clique cover given as an input. The $O^*(2^{\mathtt{cc}'})$ time algorithms use the same dynamic programming states as the standard PMC framework, but instead of using PMCs for transitions we use fast subset convolution [2]. The application of fast subset convolution requires ad-hoc techniques for each problem to take into account the cost caused by the PMC implicitly selected by the convolution.

We also give algorithms that work in polynomial space and in times $O^*(9^{\mathtt{cc}'})$ and $O^*(9^{\mathtt{cc}+O(\log^2 \mathtt{cc})})$. These algorithms are based on a polynomial space and $O^*(9^{\mathtt{cc}})$ time algorithm for enumerating PMCs and on a lemma asserting that every minimal triangulation of a graph G has a maximal clique that is in a sense a balanced separator with respect to an edge clique cover of G.

1.4 Contributions

Combinatorial bounds and enumeration algorithms We start by giving bounds for the numbers of minimal separators and PMCs. We use $S(n, k)$ to denote the Stirling numbers of the second kind, i.e., the number of ways to partition an n-element set into k non-empty subsets.

Theorem 1 *If G is a graph with an edge clique cover of size $\mathtt{cc}$, then the number of minimal separators of G is at most $S(\mathtt{cc}, 2)$ and the number of potential maximal cliques of G is at most $S(\mathtt{cc}, 3) + \mathtt{cc}$.*

We also show that these bounds are exactly tight for all values of $\mathtt{cc}$. Note that $S(\mathtt{cc}, 2) \leq 2^{\mathtt{cc}}$ and $S(\mathtt{cc}, 3) + \mathtt{cc} \leq 3^{\mathtt{cc}}$.

We use $\Delta(G)$ to denote the set of minimal separators of a graph G. There are $O^*(|\Delta(G)|)$ time algorithms for enumerating the minimal separators of G [1,40], so it follows that the minimal separators of a graph can be enumerated in $O^*(2^{\mathtt{cc}})$ time. For enumerating the PMCs $\Pi(G)$ of a graph G no algorithms that are linear in the size of the output $|\Pi(G)|$ and polynomial in the size of the input are known[1]. Despite that, we are able to design an efficient algorithm for enumerating PMCs parameterized by $\mathtt{cc}$, even when no edge clique cover is given as input. The fact that the algorithm works even when no edge clique cover is given as input is crucial because there is no $O^*(2^{2^{o(\mathtt{cc})}})$ time parameterized algorithm for minimum edge clique cover assuming the exponential time hypothesis [10].

[1] Obtaining an output-linear input-polynomial time algorithm for enumerating PMCs has been explicitly stated as an open problem in 2006 [3] and to the best of our knowledge is still open.

Table 2 Parameterized algorithms obtained as corollaries of Theorem 2

Problem	Parameter	Time complexity
Computing treewidth [15]	Minimum edge clique cover cc	$O^*(3^{cc})$
Computing (weighted) minimum fill-in [15]	Minimum edge clique cover cc	$O^*(3^{cc})$
Finding an induced subgraph with treewidth $\leq t$ satisfying a CMSO formula ϕ [17]	Minimum edge clique cover cc	$O^*(3^{cc} \cdot f(t, \phi))$
Computing fractional hypertreewidth of a hypergraph [35]	Number of hyperedges m	$O^*(3^m)$

Theorem 2 *There is an algorithm that given a graph G whose minimum edge clique cover has size cc enumerates the potential maximal cliques of G in $O^*(3^{cc})$ time.*

Corollaries of Theorem 2 From Theorem 2 it follows that all problems that admit $O^*(|\Pi(G)|)$ time algorithms when the set $\Pi(G)$ of PMCs of the input graph G is given can be solved in $O^*(3^{cc})$ time, even when no edge clique cover is given as an input. Next we briefly discuss these corollaries, summarized in Table 2.

Computing treewidth and minimum fill-in are classical triangulation problems [5]. The treewidth of a graph is the minimum possible maximum clique size in a triangulation of the graph, minus one. The minimum fill-in of a graph is the minimum number of edges to add to make a graph chordal. In weighted minimum fill-in, the input, in addition to the graph, includes a weight function assigning non-negative weights to the potential edges to add, and the task is to minimize the sum of the weights of the edges added. Our $O^*(3^{cc})$ time algorithms for treewidth and (weighted) minimum fill-in follow from Theorem 2 and the dynamic programming of Fomin et al. [15].

Theorem 2 implies corollaries for all problems in a framework called *maximum induced subgraph of bounded treewidth*, including for example the problems maximum independent set, minimum feedback vertex set, and longest induced path [17]. In particular, using the dynamic programming of [17] as a black box, we obtain $O^*(3^{cc} \cdot f(t, \phi))$ time algorithms for problems that can be expressed in a following form, where t is an integer and ϕ is a counting monadic second order logic formula: Given a graph G, find a maximum size vertex subset X so that there exists a vertex subset F with $X \subseteq F \subseteq V(G)$, the treewidth of the induced subgraph $G[F]$ is at most t, and it holds that $(G[F], X) \models \phi$.

Computing the fractional hypertreewidth of a hypergraph corresponds to finding a triangulation of its primal graph, minimizing the maximum fractional edge cover of a maximal clique in the triangulation (see Sect. 2.3 for a complete definition). By combining the fact that a primal graph of a hypergraph has edge clique cover of size m, the number of hyperedges, with Theorem 2 and the dynamic programming algorithm of [35], we obtain an $O^*(3^m)$ time algorithm for fractional hypertreewidth.

Optimizing the results in special cases For some triangulation problems we obtain better results than would directly follow from Theorem 2.

A straightforward application of Theorem 2 would result in an $O^*(6^m)$ time algorithm for generalized hypertreewidth (see definition in Sect. 2.3). We optimize this a bit.

Theorem 3 *Given a hypergraph with m hyperedges, its generalized hypertreewidth can be computed in $O^*(4^m)$ time.*

When an edge clique cover of size $\mathtt{cc}'$ is given as an input, some of the algorithms can be optimized to $O^*(2^{\mathtt{cc}'})$ time. Chordal sandwich is a special case of the weighted minimum fill-in problem, where the task is to decide if there exists a fill-in with weight zero. We pay attention to it because it has application to the perfect phylogeny problem [4,21,39].

Theorem 4 *Given a graph with an edge clique cover of size $\mathtt{cc}'$, its treewidth and minimum fill-in can be computed in $O^*(2^{\mathtt{cc}'})$ time, and also the chordal sandwich problem on it can be solved in $O^*(2^{\mathtt{cc}'})$ time.*

Corollary 1 (See [21]) *The perfect phylogeny problem can be solved in $O^*(2^n)$ time, where n is the number of taxa.*

A previous parameterized algorithm for perfect phylogeny works in time $O^*(4^r)$, where $r \leq n$ is the arity of characters [27]. Our $O^*(2^n)$ time algorithm improves over it in the case when $r > n/2$. This case is motivated by the fact that the $O^*(4^r)$ algorithm works for partial characters only via a reduction that sets $r \geq nf$ for a fraction f of missing data [39].

Polynomial space algorithms We also give polynomial space algorithms for some of the problems. The algorithms work in $O^*(9^{\mathtt{cc}'})$ time when an edge clique cover of size $\mathtt{cc}'$ is given as an input and in $O^*(9^{\mathtt{cc}+O(\log^2 \mathtt{cc})})$ time when the parameter $\mathtt{cc}$ is given as an input.

Theorem 5 *There is a polynomial space $O^*(9^{\mathtt{cc}'})$ time algorithm for treewidth and weighted minimum fill-in, where $\mathtt{cc}'$ is the size of an edge clique cover given as an input. There are also polynomial space $O^*(9^m)$ time algorithms for both generalized and fractional hypertreewidth, where m is the number of hyperedges.*

Our polynomial space result for the parameter $\mathtt{cc}$ is weaker than Theorem 2 in the sense that it requires the integer $\mathtt{cc}$ as an input.

Theorem 6 *There is a polynomial space $O^*(9^{\mathtt{cc}+O(\log^2 \mathtt{cc})})$ time algorithm for treewidth and minimum fill-in, where $\mathtt{cc}$ is an integer given as an input that is at least the size of a minimum edge clique cover of the input graph.*

The present article extends the preliminary work [29] in three ways. First, Theorem 1 has been improved so that we have lower and upper bounds that match exactly on all values of $\mathtt{cc}$. Second, Theorem 3 which provides a non-trivial extension of our results to generalized hypertreewidth has been added. Third, data on minimum edge clique covers of PACE 2017 instances has been added to Sect. 1.2.

1.5 Related Work

The prior fixed-parameter (FPT) algorithms for PMC enumeration include an $O^*(4^{\mathbf{vc}})$ time algorithm, where $\mathbf{vc}$ is the size of a minimum vertex cover, and an $O^*(1.7347^{\mathbf{mw}})$ time algorithm, where $\mathbf{mw}$ is the modular width [16], extending the $O(1.7347^n)$ time algorithm, where n is the number of vertices [17]. One can see that edge clique cover and vertex cover are orthogonal parameters by considering complete graphs and star graphs. For modular width we show in Sect. 9 that the relation $\mathbf{mw} \leq 2^{\mathtt{cc}} - 2$ holds, but there are graphs with $\mathbf{mw} = 2^{\mathtt{cc}} - 2$. In the conclusion of [16] the authors mentioned that they are not aware of other FPT parameters than $\mathbf{vc}$ and $\mathbf{mw}$ for PMCs and asked whether more parameterizations could be obtained.

Other parameterized approaches on the PMC framework include an FPT modulator parameter [32] and a slicewise polynomial (XP) bound for minimal separators in H-graphs [14]. The modulator parameter is orthogonal to edge clique cover. On H-graphs, the graphs with edge clique cover of size $\mathtt{cc}$ are $K_{\mathtt{cc}}$-graphs, so the H-graph parameterization implies an $n^{O(\mathtt{cc}^2)}$ time algorithm for enumerating PMCs.

In addition to the $O^*(4^r)$ time algorithm for perfect phylogeny [27] that we discussed in Sect. 1.4, we are not aware of prior single-exponential FPT algorithms with the same parameters as our algorithms. For fractional hypertreewidth there are parameterized algorithms whose parameters depend on the sizes of intersections of hyperedges [13]. For treewidth and chordal sandwich, different techniques have been used to obtain an $O^*(3^{\mathbf{vc}})$ time algorithm for treewidth [8] and an $O^*(2^{\mathbf{vc}'})$ time algorithm for chordal sandwich, where $\mathbf{vc}'$ is the size of a minimum vertex cover of the admissible edge set [23].

1.6 Organization of the Article

In Sect. 2 we give necessary definitions and background on minimal triangulations and PMCs. In Sect. 3 we characterize minimal separators and PMCs based on edge clique cover, proving Theorem 1. In Sect. 4 we give enumeration algorithms for PMCs, proving Theorem 2. In Sect. 5 we give the algorithm for generalized hypertreewidth, proving Theorem 3. In Sect. 6 we give faster algorithms for the case when an edge clique cover is given as an input, proving Theorem 4. In Sect. 7 we give polynomial space algorithms, proving Theorems 5 and 6. In Sect. 8 we show that our combinatorial bounds are tight. In Sect. 9 we prove the relation of edge clique cover and modular width claimed in Sect. 1.5. We conclude in Sect. 10.

2 Preliminaries

We recall the standard graph notation that we use and preliminaries on minimal triangulations. We also give formal definitions of the problems that we consider and introduce our notation related to edge clique cover.

2.1 Notation on Graphs

We consider graphs that are finite, simple, and undirected. We assume that the graphs given as input are connected. For graphs with multiple connected components, the algorithms can be applied to each connected component independently. The sets of vertices and edges of a graph G are denoted by $V(G)$ and $E(G)$, respectively. The set of edges of a complete graph with vertex set X is denoted as X^2. The subgraph $G[X]$ induced by $X \subseteq V(G)$ has $V(G[X]) = X$ and $E(G[X]) = E(G) \cap X^2$. We also use the notation $G \backslash X = G[V(G) \backslash X]$. The vertex sets of connected components of a graph G are $\mathcal{C}(G)$. The set of neighbors of a vertex v is denoted by $N(v)$ and the set of neighbors of a vertex set X by $N(X) = \bigcup_{v \in X} N(v) \backslash X$. The closed neighborhood of a vertex v is $N[v] = N(v) \cup \{v\}$ and the closed neighborhood of a vertex set X is $N[X] = N(X) \cup X$. A clique of a graph G is a vertex set X such that $G[X]$ is complete. The set of inclusion maximal cliques of G is denoted by $\mathrm{MC}(G)$.

2.2 Minimal Triangulations

A graph is *chordal* if it has no induced cycle of four or more vertices. A chordal graph H is a *triangulation* of a graph G if $V(G) = V(H)$ and $E(G) \subseteq E(H)$. A triangulation H of G is a *minimal triangulation* of G if there is no other triangulation H' of G with $E(H') \subsetneq E(H)$. The edges in $E(H) \backslash E(G)$ are *fill-edges*. A vertex set $\Omega \subseteq V(G)$ is a *potential maximal clique* (PMC) of G if there is a minimal triangulation H of G such that $\Omega \in \mathrm{MC}(H)$. The set of PMCs of G is denoted by $\Pi(G)$.

A vertex set S is a *minimal a, b-separator* of graph G if the vertices a and b are in different components of $G \backslash S$, and S is inclusion minimal in this regard. A *full component* of a vertex set X is a component $C \in \mathcal{C}(G \backslash X)$ with $N(C) = X$. We note that S is a minimal a, b-separator if and only if S has distinct full components containing a and b. A vertex set S is a *minimal separator* if it is a minimal a, b-separator for some pair a, b, i.e., it has at least two full components. We denote the set of minimal separators of G with $\Delta(G)$.

A *block* of a graph G is a vertex set $C \subseteq V(G)$ such that $G[C]$ is connected and $N(C) \in \Delta(G)$, i.e., C is a full component of some minimal separator. We remark that a common notation is to call a pair $(N(C), C)$ a full block [5]. In modern formulations of the PMC framework the concept of non-full blocks is not needed [17], so we simplify the notation by identifying the block with only the vertex set C.

Next we recall a couple of required propositions on the structure of PMCs. Figure 2 provides examples giving intuition about the propositions.

Proposition 1 [5] *A vertex set $\Omega \subseteq V(G)$ is a PMC of a graph G if and only if*

1. *$N(C) \subsetneq \Omega$ for all $C \in \mathcal{C}(G \backslash \Omega)$, i.e., no component of Ω is full, and*
2. *for all pairs of distinct vertices $u, v \in \Omega$, either $\{u, v\} \in E(G)$ or there is a component $C \in \mathcal{C}(G \backslash \Omega)$ with $\{u, v\} \subseteq N(C)$.*

We will refer to condition 1 of Proposition 1 as the *no full component condition* and to condition 2 as the *cliquish condition*. Note that Proposition 1 implies an $O(nm)$ time algorithm for testing if a given vertex set is a PMC [5].

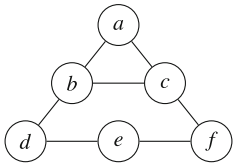

Fig. 2 An example graph G with vertex set $V(G) = \{a, b, c, d, e, f\}$. It holds that $\Omega = \{b, c, e\}$ is a PMC of G, and therefore by Proposition 2, $\{a\}$, $\{d\}$, and $\{f\}$ are blocks of Ω. As an example of Proposition 3, consider Ω and the block $C = \{a\}$. Now $N(C) = \{b, c\}$ is a minimal separator contained in Ω, and $C' = \{d, e, f\}$ is a full component of $\{b, c\}$. It holds that $\Omega \subseteq N[C'] = \{b, c, d, e, f\}$

Proposition 2 [5] *If Ω is a PMC of a graph G then all components $C \in \mathcal{C}(G \backslash \Omega)$ are blocks of G.*

We call the components $C \in \mathcal{C}(G \backslash \Omega)$ the blocks of Ω. Note that $N(C) \subsetneq \Omega$, i.e., the minimal separators of the blocks $C \in \mathcal{C}(G \backslash \Omega)$ are strict subsets of the PMC. We also need the following proposition connecting PMCs and blocks.

Proposition 3 [5] *If Ω is a PMC of a graph G and C is a block of Ω, then there is a full component C' of $N(C)$ such that $\Omega \subseteq N[C']$.*

2.3 Formal Definitions of Problems

Let G be a graph and $\mathrm{Tr}(G)$ the set of triangulations of G. The treewidth of G is $\min_{H \in \mathrm{Tr}(G)} \max_{\Omega \in \mathrm{MC}(H)} |\Omega| - 1$. Given a weight function $w : V(G)^2 \to \mathbb{R}_{\geq 0}$, the weighted minimum fill-in of G with respect to w is $\min_{H \in \mathrm{Tr}(G)} \sum_{e \in (E(H) \backslash E(G))} w(e)$. In unweighted minimum fill-in $w(e) = 1$. When a set $F \subseteq V(G)^2 \backslash E(G)$ of admissible edges is given, the chordal sandwich problem is to determine if there is a triangulation $H \in \mathrm{Tr}(G)$ with $E(H) \subseteq E(G) \cup F$. Note that chordal sandwich can be reduced to weighted minimum fill-in on the same graph G by using a weight function w with $w(e) = 0$ if $e \in F$ and $w(e) = 1$ otherwise.

A hypergraph $\mathcal{G}$ has a set of vertices $V(\mathcal{G})$ and a set of hyperedges $E(\mathcal{G})$ that are arbitrary non-empty subsets of $V(\mathcal{G})$, i.e., for $e \in E(\mathcal{G})$ it holds that $e \subseteq V(\mathcal{G})$. The primal graph $P(\mathcal{G})$ of $\mathcal{G}$ is a graph with vertices $V(P(\mathcal{G})) = V(\mathcal{G})$ and edges $E(P(\mathcal{G})) = \bigcup_{e \in E(\mathcal{G})} e^2$. An edge cover of a vertex set $X \subseteq V(\mathcal{G})$ is an assignment $c : E(\mathcal{G}) \to \{0, 1\}$ so that for each $v \in X$ it holds that $\sum_{v \in e \in E(\mathcal{G})} c(e) \geq 1$. The size of an edge cover c is $\sum_{e \in E(\mathcal{G})} c(e)$. Fractional edge cover is defined analogously to edge cover, but the function c is allowed to take any non-negative real values instead of only integers. The minimum size of an edge cover of a set $X \subseteq V(\mathcal{G})$ is denoted by $\mathrm{COV}(X)$ and fractional edge cover by $\mathrm{FCOV}(X)$. The generalized hypertreewidth of a hypergraph $\mathcal{G}$ is $\min_{H \in \mathrm{Tr}(P(\mathcal{G}))} \max_{\Omega \in \mathrm{MC}(H)} \mathrm{COV}(\Omega)$ and the fractional hypertreewidth is $\min_{H \in \mathrm{Tr}(P(\mathcal{G}))} \max_{\Omega \in \mathrm{MC}(H)} \mathrm{FCOV}(\Omega)$ [19,20,35]. Note that $\mathrm{FCOV}(X)$ can be computed in polynomial time by linear programming, but computing $\mathrm{COV}(X)$ corresponds to the NP-complete set cover problem [20].

For all of the aforementioned problems there is an optimal solution corresponding to a minimal triangulation. Furthermore, all of the problems except generalized

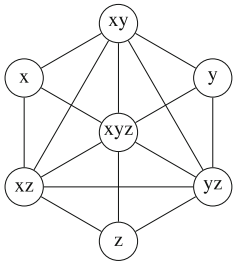

Fig. 3 An example graph G with vertex set $V(G) = \{x, y, z, xy, xz, yz, xyz\}$ and edge clique cover $\mathcal{W} = \{X, Y, Z\}$, where $X = \{x, xy, xz, xyz\}$, $Y = \{y, xy, yz, xyz\}$, and $Z = \{z, xz, yz, xyz\}$. Here, for example $\mathcal{W}[xy] = \{X, Y\}$, $\mathcal{W}[\{xy, xz\}] = \{X, Y, Z\}$, $V(G, \{X, Y\}) = \{x, y, xy\}$, $V(G, \{X, Y\}, \{Z\}) = \{x, y, xy, z\}$. Also, $\mathcal{C}(\{X, Y\}) = \{\{x, y, xy\}\}$, and because $\{x, y, xy\}$ is a block, the part $\{X, Y\}$ is good. The parts $\{X\}$ and $\{Y\}$ are not compatible because $V(G, \{X\}, \{Y\}) = \{x, y\}$ is not equal to $V(G, \{X, Y\}) = \{x, y, xy\}$

hypertreewidth can be solved in $O^*(|\Pi(G)|)$ time when $\Pi(G)$ is given as an input [15,18,21,33,35]. Generalized hypertreewidth can be solved in $O^*(|\Pi(P(\mathcal{G}))|)$ time if $\mathrm{COV}(\Omega)$ for each $\Omega \in \Pi(P(\mathcal{G}))$ is also given as input [35]. For reductions from perfect phylogeny to chordal sandwich and from maximum compatibility of binary phylogenetic characters to weighted minimum fill-in see [4,21].

2.4 Notation on Edge Clique Cover

We introduce notation for manipulating objects in a graph based on edge clique cover, with examples of the notation presented in Figure 3. An edge clique cover $\mathcal{W}$ of a graph G is a collection of subsets of $V(G)$ so that $\bigcup_{W \in \mathcal{W}} W^2 = E(G)$. We often manipulate vertex sets based on an edge clique cover $\mathcal{W}$. For a vertex $v \in V(G)$, we denote by $\mathcal{W}[v] = \{W \in \mathcal{W} \mid v \in W\}$ the set of cliques in $\mathcal{W}$ that contain v. Similarly, for a vertex set X we denote by $\mathcal{W}[X] = \bigcup_{v \in X} \mathcal{W}[v]$ the set of cliques in $\mathcal{W}$ that intersect X.

A non-empty subset $\mathcal{W}' \subseteq \mathcal{W}$ of an edge clique cover $\mathcal{W}$ is called a *part* of the edge clique cover. The vertices in $V(G, \mathcal{W}') = \{v \in V(G) \mid \mathcal{W}[v] \subseteq \mathcal{W}'\}$ are called the vertices of the part $\mathcal{W}'$. We denote the union of vertices of multiple parts with a shorthand $V(G, \mathcal{W}_1, \ldots, \mathcal{W}_p) = V(G, \mathcal{W}_1) \cup \ldots \cup V(G, \mathcal{W}_p)$. The components of a part are $\mathcal{C}(\mathcal{W}') = \mathcal{C}(G[V(G, \mathcal{W}')])$. A part is called *good* if all of its components are blocks, in which case the components of the part may be called the blocks of the part. Note that a part $\mathcal{W}'$ with $V(G, \mathcal{W}') = \emptyset$ has $\mathcal{C}(\mathcal{W}') = \emptyset$ and thus is good. Two disjoint parts $\mathcal{W}_1, \mathcal{W}_2$ are called *compatible* if $V(G, \mathcal{W}_1, \mathcal{W}_2) = V(G, \mathcal{W}_1 \cup \mathcal{W}_2)$.

3 Characterization of the Central Combinatorial Objects

In this section we show that if a graph has an edge clique cover of size $\mathtt{cc}$, then the number of blocks of the graph is at most $S(\mathtt{cc}, 2) \cdot 2$, the number of minimal separators is at most $S(\mathtt{cc}, 2)$, and the number of potential maximal cliques is at most

$S(\mathtt{cc}, 3) + \mathtt{cc}$. The characterizations of these objects will be later used in the design of the algorithms.

We start by showing that blocks correspond to parts of an edge clique cover.

Lemma 1 *Let G be a graph and $\mathcal{W}$ an edge clique cover of G. If C is a block of G then $V(G, \mathcal{W}[C]) = C$.*

Proof Clearly $C \subseteq V(G, \mathcal{W}[C])$. Note that $V(G, \mathcal{W}[C]) \subseteq N[C]$ because any vertex v intersecting a common clique with a vertex $u \in C$ must be in $N[u]$. Suppose there is a vertex $v \in (V(G, \mathcal{W}[C]) \cap N(C))$. Let C' be a full component of $N(C)$ distinct from C, implying that $v \in N(C')$. All the cliques that v intersects also intersect with C, and therefore there must be a vertex in C' that is in a clique intersecting with C which is a contradiction to the fact that $N(C)$ separates C and C'. □

By Lemma 1, any block C of G can be uniquely identified with a set $\mathcal{W}[C] \subseteq \mathcal{W}$. Moreover, there is no block with $\mathcal{W}[C] = \emptyset$ or $\mathcal{W}[C] = \mathcal{W}$, so the number of blocks is at most $2^{\mathtt{cc}} - 2$. Any minimal separator is identified as $N(C)$ of at least two blocks C, so the number of minimal separators is at most $\frac{2^{\mathtt{cc}}-2}{2} = 2^{\mathtt{cc}-1} - 1 = S(\mathtt{cc}, 2)$, proving the bound on minimal separators in Theorem 1.

Next we show that each PMC Ω is either a clique in the edge clique cover $\mathcal{W}$ or can be represented by a tripartition of edge clique cover. The high-level idea of the proof is that we can associate for each block $C \in \mathcal{C}(G \backslash \Omega)$ a part $\mathcal{W}[C]$, and then use the properties of PMCs to show that the set of these parts can be coarsed into the desired tripartition, except in the special case when $\Omega \in \mathcal{W}$. The lemma could be a bit simpler if we wanted to only prove a bound of $3^{\mathtt{cc}}$, but now it yields a tight bound.

Lemma 2 *Let G be a graph and $\mathcal{W}$ an edge clique cover of G. If Ω is a PMC of G, then either* (1) $\Omega \in \mathcal{W}$ *or* (2) $\mathcal{C}(G \backslash \Omega) = \mathcal{C}(\mathcal{W}_1) \cup \mathcal{C}(\mathcal{W}_2) \cup \mathcal{C}(\mathcal{W}_3)$*, where* $\{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3\}$ *is a partition of $\mathcal{W}$ into good parts.*

Proof First, we show that if there is a vertex $v \in \Omega$ with $|\mathcal{W}[v]| = 1$, then $\Omega = N[v]$, i.e., the sole clique containing v. Consider a vertex $u \in N(v)$ and suppose that $u \notin \Omega$ for the sake of contradiction. Note that $\mathcal{W}[v] \subseteq \mathcal{W}[u]$ and thus $N[v] \subseteq N[u]$. Now, as Ω satisfies the cliquish condition there is a path from v without intermediate vertices in Ω to all vertices in Ω. We can substitute u for v in any such path, and therefore Ω would violate the no full component condition. Therefore we have $N[v] \subseteq \Omega$, which implies $N[v] = \Omega$ because now v cannot satisfy the cliquish condition with any vertex not in $N[v]$.

We showed that case 1 applies to PMCs containing vertices v with $|\mathcal{W}[v]| = 1$. Next, we suppose that we have PMC Ω for which case 1 does not apply, and show that case 2 applies. By Proposition 2 the components $C \in \mathcal{C}(G \backslash \Omega)$ are blocks and by Lemma 1 they define a collection $P = \{\mathcal{W}[C] \mid C \in \mathcal{C}(G \backslash \Omega)\}$ of disjoint good parts of $\mathcal{W}$. We make this collection a partition of $\mathcal{W}$ by adding the missing elements $\mathcal{W} \backslash (\bigcup_{\mathcal{W}_i \in P} \mathcal{W}_i)$ as singletons. If for any such singleton $\{W\}$ there is a vertex v with $\mathcal{W}[v] \subseteq \{W\}$, then case 1 applies. Therefore we have that $V(G, \{W\}) = \emptyset$ for such singletons, and thus they are good parts. Now we have a partition P of $\mathcal{W}$ into good parts with $\bigcup_{\mathcal{W}_i \in P} \mathcal{C}(\mathcal{W}_i) = \mathcal{C}(G \backslash \Omega)$.

If the partition P would consist of only a single part then Ω would be empty. Suppose the number of parts is two, i.e., $P = \{\mathcal{W}_1, \mathcal{W}_2\}$. Now, if either $V(G, \mathcal{W}_1)$ or $V(G, \mathcal{W}_2)$ is nonempty, then the no full component condition would be violated by it. If both $V(G, \mathcal{W}_1)$ and $V(G, \mathcal{W}_2)$ are empty, then $\Omega = V(G)$, which implies that $\mathcal{W}_1$ and $\mathcal{W}_2$ are singletons and $|\mathcal{W}| = 2$, which implies that $V(G) = \mathcal{W}_1 = \mathcal{W}_2 = \Omega$, i.e., that case 1 would apply.

If the number of parts is at least three, merge arbitrary compatible pairs of parts until the number of parts is three or no pairs of parts can be merged anymore. If we end up with three parts, we are done. If we end up with more than three parts, i.e. $P = \{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3, \mathcal{W}_4, \ldots\}$, then take a vertex $u \in (V(G, \mathcal{W}_1 \cup \mathcal{W}_2) \setminus V(G, \mathcal{W}_1, \mathcal{W}_2))$ and a vertex $v \in (V(G, \mathcal{W}_3 \cup \mathcal{W}_4) \setminus V(G, \mathcal{W}_3, \mathcal{W}_4))$. Because of our assumption that we cannot continue the merging process anymore both of these vertices exist and are in Ω. However, there is no edge between u and v and there is no common component in whose neighborhood u and v are, which is a contradiction to the cliquish condition. □

We call the PMCs corresponding to case 1 of Lemma 2 type 1 PMCs and the PMCs corresponding to case 2 type 2 PMCs. The number of type 1 PMCs is at most $\mathtt{cc}$. Type 2 PMCs correspond to tripartitions of $\mathcal{W}$, so their number is at most $S(\mathtt{cc}, 3)$. Therefore, the total number of PMCs is at most $S(\mathtt{cc}, 3) + \mathtt{cc}$, completing the proof of Theorem 1.

4 Enumeration Algorithms

We modify the Bouchitté–Todinca algorithm [6] for enumerating PMCs to give an $O^*(3^{\mathtt{cc}})$ time PMC enumeration algorithm and a polynomial space $O^*(9^{\mathtt{cc}})$ time PMC enumeration algorithm. Recall that Theorem 1 with Takata's algorithm [40] implies a polynomial space $O^*(2^{\mathtt{cc}})$ time minimal separator enumeration algorithm.

The Bouchitté–Todinca algorithm characterizes PMCs based on minimal separators. We summarize this characterization in the following proposition.

Proposition 4 [6] *Let G be a connected graph with $|V(G)| > 1$ and v any vertex of G. If Ω is a PMC of G, one of the following holds.*

1. $\Omega \setminus \{v\} \in \Pi(G \setminus \{v\})$.
2. $\Omega \setminus \{v\} \in \Delta(G)$.
3. $\Omega = S \cup T$, *where* $S \in \Delta(G)$ *and* $T \in \Delta(G[C \cup \{x, y\}])$, *where C is a full component of S and x and y are non-adjacent vertices in S.*

The algorithm uses case 1 to generate n induced subgraphs of the input graph, from which PMCs are generated by cases 2 and 3. Note that the size of a minimum edge clique cover is monotone with respect to induced subgraphs. We need the following lemma, which follows from Proposition 1, to ensure that each PMC of each induced subgraph corresponds to at most one PMC of the original graph.

Lemma 3 [6] *Let G be a graph, $\Omega \in \Pi(G)$, and $v \in \Omega$. The set $\Omega \setminus \{v\}$ is not a PMC of G.*

Proof The PMC Ω satisfies the cliquish condition, so there is a path in G from v to each vertex in $\Omega \setminus \{v\}$ without intermediate vertices in Ω. Therefore $\Omega \setminus \{v\}$ violates the no full component condition. □

Now, we can just enumerate PMCs from cases 2 and 3 in each of the n induced subgraphs, and each time a PMC is found we use case 1 with Lemma 3 to generate at most one PMC of the original graph in polynomial time.

Next we complete the description of the algorithm by showing that PMCs from cases 2 and 3 can be enumerated in $O^*(3^{cc})$ time. The proof is based on Lemma 1 on the structure and the number of blocks.

Lemma 4 *There is an algorithm that given a graph G whose minimum edge clique cover has size cc enumerates the PMCs of G, possibly with duplicates, in polynomial space and $O^*(3^{cc})$ time.*

Proof As discussed above, it is sufficient to enumerate PMCs from cases 2 and 3 of Proposition 4. Case 2 simply corresponds to minimal separator enumeration, so we use the $O^*(2^{cc})$ time polynomial space minimal separator enumeration algorithm. For case 3, we do the polynomial space minimal separator enumeration in the graph G, and every time we output a minimal separator S, we do polynomial space enumeration of minimal separators in the graph $G[C \cup \{x, y\}]$ for all full components C of S and all non-adjacent pairs $x, y \in S$.

We do the inner iteration $O(n^2)$ times for each block C of G. The complexity of the inner iteration depends on the size of a minimum edge clique cover of the graph $G[C \cup \{x, y\}]$. Let $\mathcal{W}$ be a minimum edge clique cover of G. By Lemma 1, the block C corresponds to a unique subset $\mathcal{W}[C]$ of $\mathcal{W}$. The subset $\mathcal{W}[C]$ is an edge clique cover of $G[C \cup \{x, y\}]$ because all edges in it are adjacent to C because x and y are non-adjacent. Therefore, the time complexity of the inner iteration is $O^*(2^{|\mathcal{W}[C]|})$ and the total time complexity of the algorithm is $\sum_{\mathcal{W}' \subseteq \mathcal{W}} O^*(2^{|\mathcal{W}'|}) = O^*(3^{cc})$. □

Using for example sorting we can deduplicate the output of the algorithm of Lemma 4 and an $O^*(3^{cc})$ time and space algorithm for enumerating PMCs without duplicates, i.e., Theorem 2, follows. For deduplication in polynomial space, we use a simple trick that is efficient enough for our purposes.

Lemma 5 *There is an algorithm that given a graph G whose minimum edge clique cover has size cc enumerates the PMCs of G in polynomial space and $O^*(9^{cc})$ time.*

Proof Run the algorithm of Lemma 4 multiple times in succession, each time outputting the lexicographically smallest PMC that is lexicographically larger than the previous PMC outputted, until no such PMC is found. Now, using the algorithm at most 3^{cc} times we have outputted the PMCs of G in lexicographically strictly increasing order. □

5 Generalized Hypertreewidth

For generalized hypertreewidth we need to compute minimum edge covers of all PMCs, so a naive application of Theorem 2 with an $O^*(2^m)$ time set cover algorithm

results in an $O^*(6^m)$ time algorithm, where m is the number of hyperedges. In this section we give an $O^*(4^m)$ time algorithm for enumerating PMCs with their minimum edge covers, implying an $O^*(4^m)$ time algorithm for generalized hypertreewidth. We note that the naive approach with Lemma 5 is sufficient to give an $O^*(9^m)$ time polynomial space algorithm for enumerating PMCs with minimum edge covers.

In this section we let $\mathcal{W}$ be the edge clique cover of size m formed by the hyperedges of the input hypergraph. We use the characterization of PMCs given in Lemma 2. PMCs of type 1 have edge covers of size 1, so we focus on PMCs of type 2, which correspond to tripartitions of hyperedges. Let $\{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3\}$ be a tripartition corresponding to a type 2 PMC $\Omega = V(G)\setminus V(G, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3)$. We use the following notation to cover vertices of the PMC with hyperedges.

Definition 1 Let G be a graph, $\mathcal{W}$ an edge clique cover of G, and $\{D, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3\}$ a labeled four-partition of $\mathcal{W}$ with empty parts allowed. The partial PMC induced by $\{D, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3\}$ is the vertex set $\{v \in V(G) \mid D \cap \mathcal{W}[v] = \emptyset\}\setminus V(G, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3)$, which we denote by $\Gamma(G, D, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3)$.

Now we can observe that if a tripartition $\{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3\}$ corresponds to a PMC Ω, then $\Gamma(G, \emptyset, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3) = \Omega$. Furthermore, $\Gamma(G, D, \mathcal{W}_1\setminus D, \mathcal{W}_2\setminus D, \mathcal{W}_3\setminus D)$ is an empty set if and only if D is an edge cover of Ω. Therefore the task of finding a minimum edge cover of a PMC $\Omega = V(G)\setminus V(G, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3)$ reduces to finding a minimum size set $D \subseteq \mathcal{W}$ that makes $\Gamma(G, D, \mathcal{W}_1\setminus D, \mathcal{W}_2\setminus D, \mathcal{W}_3\setminus D)$ empty. We solve this with dynamic programming in $O^*(4^m)$ total time.

Lemma 6 *Let $\mathcal{G}$ be a hypergraph. The PMCs of the primal graph $P(\mathcal{G})$ with minimum edge covers $COV(\Omega)$ for each PMC Ω can be enumerated in $O^*(4^m)$ time, where m is the number of hyperedges of $\mathcal{G}$.*

Proof Let $G = P(\mathcal{G})$, $\mathcal{W} = E(\mathcal{G})$, and $\{D, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3\}$ be as in Definition 1. We denote by $\mathrm{COV}(D, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3)$ the minimum size of $D' \subseteq \mathcal{W}$ such that the set $\Gamma(G, D \cup D', \mathcal{W}_1\setminus D', \mathcal{W}_2\setminus D', \mathcal{W}_3\setminus D')$ is empty. We compute COV for all such partitions by the recursion $\mathrm{COV}(D, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3) =$

$$\begin{cases} 0 \text{ if } \Gamma(G, D, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3) = \emptyset \text{ and} \\ 1 + \min_{W \in \mathcal{W}} \mathrm{COV}(D \cup \{W\}, \mathcal{W}_1\setminus\{W\}, \mathcal{W}_2\setminus\{W\}, \mathcal{W}_3\setminus\{W\}) \text{ otherwise.} \end{cases}$$

This recursion can be evaluated and stored in $O^*(4^m)$ time. Now, we iterate over all tripartitions $\{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3\}$ of $\mathcal{W}$ and if $V(G)\setminus V(G, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3)$ is a PMC we output it and $\mathrm{COV}(\emptyset, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3)$ as its minimum edge cover. □

Theorem 3 follows.

6 Faster Algorithms When Edge Clique Cover is Given

We use fast subset convolution [2] to design $O^*(2^{cc'})$ time algorithms for treewidth, minimum fill-in, and chordal sandwich, where cc' is the size of an edge clique cover given as an input. In particular, we make use of the following result.

Proposition 5 [2] *Let X be a set, $f : 2^X \to [M]$ and $g : 2^X \to [M]$ functions from the set 2^X of all subsets of X to the set of integers up to M. The function $(f * g)$ defined as $(f * g)(Y) = \min_{Y' \subseteq Y} f(Y') + g(Y \setminus Y')$ can be computed for all subsets $Y \subseteq X$ in $O^*(2^{|X|}M)$ time.*

The algorithms we introduce are modifications of the dynamic programming phase of the PMC framework. In most of this section our presentation is general in the sense that it applies to each of the three problems. We use the term "optimal triangulation" to refer to a triangulation with the minimum size of a maximum clique in the context of treewidth, a triangulation with the least number of edges in the context of minimum fill-in, and to a triangulation that has no non-admissible fill-edges in the context of chordal sandwich (or to information that no such triangulation exists).

We start by recalling the dynamic programming phase of the PMC framework. The states of the dynamic programming correspond to *realizations* of blocks.

Definition 2 [5] Let G be a graph and C a block of G. A realization $R(C)$ of C is a graph with vertex set $V(R(C)) = N[C]$ and edge set $E(R(C)) = E(G[N[C]]) \cup N(C)^2$.

The following proposition characterizes minimal triangulations of a realization of a block.

Proposition 6 [5] *Let G be a graph and C a block of G. The graph H is a minimal triangulation of $R(C)$ if and only if* (i) $V(H) = N[C]$ *and* (ii) *there is a PMC $\Omega \in \Pi(G)$ with $N(C) \subseteq \Omega \subseteq N[C]$ and*

$$E(H) = \Omega^2 \cup \bigcup_{C_i \in \mathcal{C}(R(C) \setminus \Omega)} E(H_i),$$

where H_i is any minimal triangulation of $R(C_i)$. Moreover, each C_i is a block of G.

Proposition 6 implies dynamic programming algorithms for computing optimal triangulations of realizations of all blocks [5,15,33].

We use the following proposition for a base case.

Proposition 7 [5] *Let G be a graph that is not complete. The graph H is a minimal triangulation of G if and only if* (i) $V(H) = V(G)$ *and* (ii) *there is a minimal separator $S \in \Delta(G)$ with*

$$E(H) = \bigcup_{C_i \in \mathcal{C}(G \setminus S)} E(H_i),$$

where H_i is any minimal triangulation of $R(C_i)$.

Once we have computed optimal triangulations of realizations of all blocks, we can compute an optimal triangulation of the graph via Proposition 7 in time $O^*(2^{cc})$. For computing optimal triangulations of realizations, the bottleneck in implementing the recursion of Proposition 6 is in iterating over the PMCs.

The high-level idea of our algorithm is that we use Proposition 6 directly only with type 1 PMCs, i.e., PMCs $\Omega \in \mathcal{W} \cap \Pi(G)$. For type 2 PMCs, we simulate the

iteration over PMCs with fast subset convolution. In particular, we show that each PMC Ω of type 2 with $N(C) \subseteq \Omega \subseteq N[C]$ can be expressed in terms of two disjoint good parts $\mathcal{W}_1$ and $\mathcal{W}_2$ of $\mathcal{W}[C]$, where $\mathcal{W}$ is an edge clique cover of G. In the case of treewidth, minimum fill-in, and chordal sandwich, an optimal partition of every subset $\mathcal{W}' \subseteq \mathcal{W}$ into two good parts $\mathcal{W}_1$ and $\mathcal{W}_2$ can be computed with fast subset convolution, provided that we have first computed optimal triangulations of realizations of all blocks in $\mathcal{C}(\mathcal{W}_1)$ and $\mathcal{C}(\mathcal{W}_2)$.

We first show that each type 2 PMC can be expressed in terms of $\mathcal{W}_1$ and $\mathcal{W}_2$.

Lemma 7 *Let G be a graph, $\mathcal{W}$ an edge clique cover of G, and C a block of G. If a graph H is a minimal triangulation of $R(C)$, then either* (1) *there is a clique $\Omega \in \mathcal{W} \cap \Pi(G)$ and*

$$E(H) = \Omega^2 \cup \bigcup_{C_i \in \mathcal{C}(R(C)\setminus\Omega)} E(H_i),$$

where $N(C) \subseteq \Omega \subseteq N[C]$, and H_i is a minimal triangulation of $R(C_i)$, or (2) *there is a partition $\{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_o\}$ of $\mathcal{W}$ into good parts with $\mathcal{W}_1 \cup \mathcal{W}_2 \subseteq \mathcal{W}[C]$, a block $C' \in \mathcal{C}(\mathcal{W}_o)$ with $N(C) = N(C')$, and*

$$E(H) = \Omega^2 \cup \bigcup_{C_i \in \mathcal{C}(\mathcal{W}_1)\cup\mathcal{C}(\mathcal{W}_2)\cup(\mathcal{C}(\mathcal{W}_o)\setminus\mathcal{C}(G\setminus N(C)))} E(H_i),$$

where $\Omega = V(G)\setminus V(G, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_o)$ and H_i is a minimal triangulation of $R(C_i)$.

Proof Case 1 corresponds to Proposition 6 with PMCs of type 1. Next we prove that case 2 covers all PMCs of type 2.

Let Ω be a PMC of G with $N(C) \subseteq \Omega \subseteq N[C]$ and $\Omega \notin \mathcal{W}$. We consider the part $\mathcal{W}'_o = \mathcal{W}\setminus\mathcal{W}[C]$ and prove that $\mathcal{W}'_o$ is a good part and $\mathcal{C}(\mathcal{W}'_o) = \mathcal{C}(G\setminus N(C))\setminus\{C\}$. Observe that $\{C, N(C), V(G, \mathcal{W}'_o)\}$ is a partition of $V(G)$, and moreover there are no edges between C and $V(G, \mathcal{W}'_o)$. Now, for any component $C' \in \mathcal{C}(\mathcal{W}'_o)$, it must hold that $N(C') \subseteq N(C)$, and therefore $N(C')$ is a minimal separator and therefore $\mathcal{W}'_o$ is a good part whose blocks are the components of $G\setminus N(C)$ except C.

Similarly as in the proof of Lemma 2, consider the collection of disjoint good parts $P = \{\mathcal{W}[C_i] \mid C_i \in \mathcal{C}(G\setminus\Omega)\}$. All of the parts that intersect $\mathcal{W}'_o$ are subsets of $\mathcal{W}'_o$ because they do not intersect $\mathcal{W}[C]$, and therefore we replace the parts that intersect $\mathcal{W}'_o$ by the part $\mathcal{W}'_o$. Now by similar arguments as in Lemma 2 we add additional parts to the collection to make it a full partition of $\mathcal{W}$. Now we have a partition P of $\mathcal{W}$ with $|P| \geq 2$ and $\mathcal{W}'_o \in P$. Moreover, $\bigcup_{\mathcal{W}_i \in P} \mathcal{C}(\mathcal{W}_i) = \mathcal{C}(G\setminus\Omega)$. If $|P| = 2$, then it would hold that $P = \{\mathcal{W}'_o, \mathcal{W}[C]\}$, in which case $\Omega = N(C)$ would hold, which is a contradiction. If there are at least three parts, then merge compatible parts until we have three parts or cannot merge parts anymore. By the proof of Lemma 2, we will end up with three parts. Now let $\mathcal{W}_o$ be the part that contains $\mathcal{W}'_o$ and $\mathcal{W}_1$ and $\mathcal{W}_2$ the other two parts. Because $\mathcal{W}'_o = \mathcal{W}\setminus\mathcal{W}[C]$, we have that $\mathcal{W}_1 \cup \mathcal{W}_2 \subseteq \mathcal{W}[C]$. Moreover, because $N(C)$ has at least two full components, and all components of $N(C)$ except C are components of $\mathcal{W}'_o$, we have that there is a block $C' \in \mathcal{C}(\mathcal{W}_o)$ with $N(C') = N(C)$.

Finally, we show that $\mathcal{C}(R(C)\backslash\Omega) = \mathcal{C}(\mathcal{W}_1) \cup \mathcal{C}(\mathcal{W}_2) \cup (\mathcal{C}(\mathcal{W}_o)\backslash\mathcal{C}(G\backslash N(C)))$. By Proposition 3 we have that $\mathcal{C}(R(C)\backslash\Omega) = \mathcal{C}(G\backslash\Omega)\backslash\mathcal{C}(G\backslash N(C))$ and therefore $\mathcal{C}(R(C)\backslash\Omega) = \mathcal{C}(\mathcal{W}_1) \cup \mathcal{C}(\mathcal{W}_2) \cup \mathcal{C}(\mathcal{W}_o)\backslash\mathcal{C}(G\backslash N(C))$. All blocks of $\mathcal{W}_1$ and $\mathcal{W}_2$ are subsets of C because $\mathcal{W}_1 \cup \mathcal{W}_2 \subseteq \mathcal{W}[C]$. □

The following lemma guarantees that the characterization of PMCs of type 2 in Lemma 7 is sound in the sense that all graphs H that it defines are (not necessarily minimal) triangulations of G.

Lemma 8 *Let G be a graph, $\mathcal{W}$ an edge clique cover of G, and C a block of G. Also, let $\{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_o\}$ be a partition of $\mathcal{W}$ into good parts with $\mathcal{W}_1 \cup \mathcal{W}_2 \subseteq \mathcal{W}[C]$ and C' a block of G with $C' \in \mathcal{C}(\mathcal{W}_o)$ and $N(C) = N(C')$. Let H be any graph with (i) $V(H) = N[C]$ and (ii)*

$$E(H) = \Omega^2 \cup \bigcup_{C_i \in \mathcal{C}(\mathcal{W}_1) \cup \mathcal{C}(\mathcal{W}_2) \cup (\mathcal{C}(\mathcal{W}_o)\backslash\mathcal{C}(G\backslash N(C)))} E(H_i),$$

where $\Omega = V(G)\backslash V(G, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_o)$ and H_i is any triangulation of $R(C_i)$. The graph H is a triangulation of $R(C)$.

Proof To show that H is a triangulation of $R(C)$ we show that $N(C) \subseteq \Omega \subseteq N[C]$ and $\mathcal{C}(R(C)\backslash\Omega) = \mathcal{C}(\mathcal{W}_1) \cup \mathcal{C}(\mathcal{W}_2) \cup (\mathcal{C}(\mathcal{W}_o)\backslash\mathcal{C}(G\backslash N(C)))$.

For $\Omega \subseteq N[C]$, note that no vertex of Ω is in $V(G, \mathcal{W}_o)$, and therefore all vertices of Ω intersect a clique in $\mathcal{W}[C]$, and therefore are in $N[C]$. Furthermore, $N(C) \subseteq \Omega$ holds because there is a block $C' \in \mathcal{C}(\mathcal{W}_o)$ with $N(C) = N(C')$.

The components of Ω in G are $\mathcal{C}(\mathcal{W}_1) \cup \mathcal{C}(\mathcal{W}_2) \cup \mathcal{C}(\mathcal{W}_o)$ by definition. Because $N(C) \subseteq \Omega \subseteq N[C]$, all components of Ω in G are either in C, and thus components of Ω in $R(C)$ or are components of $N(C)$. All components in $\mathcal{C}(\mathcal{W}_1) \cup \mathcal{C}(\mathcal{W}_2)$ are subsets of C because $\mathcal{W}_1 \cup \mathcal{W}_2 \subseteq \mathcal{W}[C]$, and therefore are components of Ω in $R(C)$. All components of $N(C)$ are outside of C except C itself. It remains to show that C is not a component of Ω. Any set $\mathcal{W}'$ with $C \subseteq V(G, \mathcal{W}')$ must be a superset of $\mathcal{W}[C]$. However, because $\mathcal{W}_1$ and $\mathcal{W}_2$ are proper subsets of $\mathcal{W}[C]$, no set in $\{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_o\}$ can be a superset of $\mathcal{W}[C]$. □

In Lemmas 7 and 8 we formulated a recursion that characterizes all minimal triangulations of a realization $R(C)$ of a block C in terms of minimal triangulations of realizations $R(C')$ of blocks $C' \subsetneq C$. What remains is to integrate the computation of the optimal cost of the triangulation into this characterization. The following lemma is used for treewidth and minimum fill-in. It is simple, but we state it as a warmup for what follows.

Lemma 9 *Let G be a graph, $\mathcal{W}$ an edge clique cover of G, $\{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_o\}$ a partition of $\mathcal{W}$, and $\Omega = V(G)\backslash V(G, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_o)$ a vertex set defined by the partition. It holds that $|\Omega| = |V(G)| - |V(G, \mathcal{W}_1)| - |V(G, \mathcal{W}_2)| - |V(G, \mathcal{W}_o)|$.*

Proof The sets $V(G, \mathcal{W}_1)$, $V(G, \mathcal{W}_2)$, and $V(G, \mathcal{W}_o)$ are disjoint because $\mathcal{W}_1$, $\mathcal{W}_2$ and $\mathcal{W}_o$ are disjoint. □

Therefore, the size of Ω can be computed as a sum that considers the parts $\mathcal{W}_1$, $\mathcal{W}_2$, and $\mathcal{W}_o$ independently, and therefore we can integrate the computation of it into fast subset convolution. For treewidth, we only have to make sure that $|\Omega| \leq k+1$, where k is the upper bound for treewidth in the decision problem. For minimum fill-in, we can compute the number of edges in the triangulation of $R(C)$ as $\binom{|\Omega|}{2} + \sum_{C_i \in \mathcal{C}(R(C)\setminus\Omega)}(|E(H_i)| - \binom{|N(C_i)|}{2})$, where H_i is an optimal triangulation of the realization $R(C_i)$.

A similar lemma is used for chordal sandwich.

Lemma 10 *Let G be a graph, $\mathcal{W}$ an edge clique cover of G, $\{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_o\}$ a partition of $\mathcal{W}$ into good parts, and $\Omega = V(G)\setminus V(G, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_o)$. It holds that $\Omega^2 = \bigcup_{\mathcal{W}_i \in \{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_o\}} (V(G)\setminus V(G, \mathcal{W}_i, \mathcal{W}\setminus\mathcal{W}_i))^2$.*

Proof We first show that for all pairs $u, v \in \Omega$ there is a part $\mathcal{W}_i \in \{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_o\}$ such that $\{u, v\} \subseteq (V(G)\setminus V(G, \mathcal{W}_i, \mathcal{W}\setminus\mathcal{W}_i))$. Neither $\mathcal{W}[u]$ or $\mathcal{W}[v]$ is a subset of any of $\mathcal{W}_1, \mathcal{W}_2$, or $\mathcal{W}_o$, so there is at least of part of the partition that they both intersect but are not subsets of and that is the desired part $\mathcal{W}_i$.

To see that $(V(G)\setminus V(G, \mathcal{W}_i, \mathcal{W}\setminus\mathcal{W}_i)) \subseteq \Omega$, take any vertex $v \in (V(G)\setminus\Omega)$ and consider the set $\mathcal{W}[v]$. The set $\mathcal{W}[v]$ is a subset of one of $\mathcal{W}_1, \mathcal{W}_2$, or $\mathcal{W}_o$. Now, each of $\{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_o\}$ is either $\mathcal{W}_i$ or a subset of $\mathcal{W}\setminus\mathcal{W}_i$. □

Lemma 10 guarantees that each fill-edge caused by the PMC Ω can be "seen" from at least one of the parts $\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_o$, implying that it is sufficient to check each part independently to guarantee that Ω does not add any forbidden fill-edges. We remark that Lemma 10 appears to be difficult to generalize to count the exact number of fill-edges, which is the barrier why we are not able to give an $O^*(2^{cc'})$ time algorithm for weighted minimum fill-in.

Algorithm 1 presents the full $O^*(2^{cc'})$ time algorithm for treewidth. The algorithms for minimum fill-in and chordal sandwich are similar. The algorithm maintains a collection $\mathcal{B}$ of blocks C for which it is known that the treewidth of $R(C)$ is at most k. The invariant of the main loop of lines 2 to 18 is that after ith iteration, all blocks of size at most i and treewidth at most k have been added to $\mathcal{B}$. In each iteration of the main loop, the algorithm iterates over all good parts $\mathcal{W}'$ on lines 4 to 6, and if all realizations of blocks of the part have treewidth at most k adds the part to a collection $F_{|V(G,\mathcal{W}')|}$. These parts $\mathcal{W}'$ correspond to parts $\mathcal{W}_1$ and $\mathcal{W}_2$ of our lemmas. Then, fast subset convolution is applied on line 7 on the collections F to find for all combinations $(\mathcal{W}_1 \cup \mathcal{W}_2)$ of disjoint good parts $\mathcal{W}_1$ and $\mathcal{W}_2$ the maximum number of vertices in $V(G, \mathcal{W}_1, \mathcal{W}_2)$. Then on lines 8 to 14 the algorithm iterates through all good parts $\mathcal{W}_o$, thus determining $(\mathcal{W}_1 \cup \mathcal{W}_2)$ and all other variables that need to be taken into account. In particular, note that each part $\mathcal{W}_o$ determines only polynomially many blocks C such that there is $C' \in \mathcal{C}(\mathcal{W}_o)$ with $N(C') = N(C)$.

The analysis of the algorithm focuses on transitions via PMCs of type 2 and proceeds by induction on the main loop invariant. The time complexity follows simply from fast subset convolution, the bound 2^{cc} on the number of blocks, and the fact that each iteration of the loop of the lines 8 to 14 takes polynomial time. The correctness is shown by combining the lemmas introduced in this section.

Algorithm 1: Treewidth in $O^*(2^{cc'})$ time

Input : Connected graph G, an edge clique cover $\mathcal{W}$ of G, and an integer k
Output: Whether the treewidth of G is at most k

1 **if** G *is complete* **then return** $|V(G)| \leq k+1$ Let $\mathcal{B} \leftarrow \emptyset$ be a collection of blocks of G
2 **for** $i \leftarrow 1$ **to** n **do**
3 For each $0 \leq j \leq n$ let $F_j \leftarrow \emptyset$ be a collection of subsets of $\mathcal{W}$
4 **foreach** *good part* $\mathcal{W}' \subseteq \mathcal{W}$ **do**
5 **if** $\mathcal{C}(\mathcal{W}') \subseteq \mathcal{B}$ **then**
6 $F_{|V(G,\mathcal{W}')|} \leftarrow F_{|V(G,\mathcal{W}')|} \cup \{\mathcal{W}'\}$
7 Use fast subset convolution to compute for each subset $\mathcal{W}' \subseteq \mathcal{W}$ the maximum value of $a+b$ such that there is $\mathcal{W}'' \subseteq \mathcal{W}'$ with $\mathcal{W}'' \in F_a$ and $(\mathcal{W}' \backslash \mathcal{W}'') \in F_b$
8 **foreach** *good part* $\mathcal{W}_o \subseteq \mathcal{W}$ **do**
9 Let t be the value on $\mathcal{W} \backslash \mathcal{W}_o$ computed in line 7
10 **if** t *exists and* $n - t - |V(G, \mathcal{W}_o)| \leq k+1$ **then**
11 **foreach** *minimal separator* $N(C')$ *with* $C' \in \mathcal{C}(\mathcal{W}_o)$ **do**
12 **if** *exists* $C \in \mathcal{C}(G \backslash N(C'))$ *with* $(\mathcal{W} \backslash \mathcal{W}_o) \subseteq \mathcal{W}[C]$ **then**
13 **if** $(\mathcal{C}(\mathcal{W}_o) \backslash \mathcal{C}(N(C))) \subseteq \mathcal{B}$ **then**
14 $\mathcal{B} \leftarrow \mathcal{B} \cup \{C\}$
15 **foreach** *block* C *of* G **do**
16 **foreach** $\Omega \in \mathcal{W} \cap \Pi(G)$ *with* $|\Omega| \leq k+1$ *and* $N(C) \subseteq \Omega \subseteq N[C]$ **do**
17 **if** $\mathcal{C}(R(C) \backslash \Omega) \subseteq \mathcal{B}$ **then**
18 $\mathcal{B} \leftarrow \mathcal{B} \cup \{C\}$
19 **foreach** $S \in \Delta(G)$ **do**
20 **if** $\mathcal{C}(G \backslash S) \subseteq \mathcal{B}$ **then**
21 **return** True
22 **return** False

Lemma 11 *There is an algorithm that given a graph G with an edge clique cover $\mathcal{W}$ of size cc' determines the treewidth and minimum fill-in of G in time $O^*(2^{cc'})$. Furthermore, if also a set $F \subseteq V(G)^2 \backslash E(G)$ is given the algorithm determines if there is a triangulation H of G with $E(H) \subseteq E(G) \cup F$.*

Proof By Theorem 1 and Proposition 7 it suffices to solve each of the problems for all realizations of blocks of G. We proceed via induction, i.e., assume that the problems have been solved for all realizations of blocks C with $|C| < i$. We show that in one step taking a total of $O^*(2^{cc'})$ time the problems can be solved for all realizations of blocks C with $|C| = i$.

First we find optimal triangulations via Proposition 6 using PMCs of type 1, i.e., PMCs $\Omega \in \mathcal{W} \cap \Pi(G)$. For deciding if treewidth is at most k we have to assert that $|\Omega| \leq k+1$ and that the resulting smaller blocks have treewidth of realizations at most k. For minimum fill-in, the number of fill-edges can be computed as $|\Omega^2 \backslash E(R(C))|$ plus the numbers of fill-edges in optimal solutions of the smaller blocks. For chordal sandwich we have to assert that $\Omega^2 \subseteq E(G) \cup F$ and that the resulting smaller blocks also have positive answers.

Now we can focus on type 2 PMCs, i.e., implement the characterization of Lemmas 7 and 8 while keeping track of the optimal answer. For each of the problems we use fast subset convolution [2] to determine in $O^*(2^{cc'})$ time for each subset $\mathcal{W}' \subseteq \mathcal{W}$ an optimal partition of $\mathcal{W}'$ into two good parts $\mathcal{W}_1$ and $\mathcal{W}_2$. For treewidth we determine the maximum value of $|V(G, \mathcal{W}_1, \mathcal{W}_2)|$ such that all realizations of blocks

in $\mathcal{C}(\mathcal{W}_1) \cup \mathcal{C}(\mathcal{W}_2)$ have treewidth at most k. For minimum fill-in, we determine for each value $0 \leq p \leq n$ the minimum value of sum of minimum fill-ins of realizations of blocks in $\mathcal{C}(\mathcal{W}_1) \cup \mathcal{C}(\mathcal{W}_2)$ such that $|V(G, \mathcal{W}_1, \mathcal{W}_2)| = p$. For chordal sandwich, we decide if there is a partition such that the answer is positive for all realizations of blocks in $\mathcal{C}(\mathcal{W}_1) \cup \mathcal{C}(\mathcal{W}_2)$ and that $(V(G)\backslash V(G, \mathcal{W}_i, \mathcal{W}\backslash\mathcal{W}_i))^2 \subseteq E(G) \cup F$. Now, once we fix the set $\mathcal{W}' = (\mathcal{W}_1 \cup \mathcal{W}_2)$, by Lemmas 9 and 10 we have computed all relevant information about $\mathcal{W}_1$ and $\mathcal{W}_2$ with the convolution, and can compute the rest of the information of the PMC by $\mathcal{W}_o = \mathcal{W}\backslash(\mathcal{W}_1 \cup \mathcal{W}_2)$. Because $\mathcal{W}_o$ defines polynomially many blocks C with $C' \in \mathcal{C}(\mathcal{W}_o)$ and $N(C) = N(C')$ we can update all relevant blocks in polynomial time. □

Theorem 4 follows.

7 Polynomial Space Algorithms

We give polynomial space algorithms for treewidth, weighted minimum fill-in, and generalized and fractional hypertreewidth. The algorithms are based on the following characterization of minimal triangulations. Note that the characterization uses realizations of blocks as defined in Sect. 6.

Proposition 8 [5] *Let G be a graph, H a minimal triangulation of G, and Ω a maximal clique of H. For each $C_i \in \mathcal{C}(G\backslash\Omega)$ there exists a minimal triangulation H_i of $R(C_i)$ such that*

$$E(H) = \Omega^2 \cup \bigcup_{C_i \in \mathcal{C}(G\backslash\Omega)} E(H_i).$$

Note that by iterating over all $\Omega \in \Pi(G)$ in Proposition 8 we can indeed construct all minimal triangulations of G. Furthermore, all graphs H constructed in this manner are minimal triangulations [5].

The idea of the algorithm is to use the recursion of Proposition 8 directly, without dynamic programming. The following "balanced PMC" lemma guarantees that for any resulting minimal triangulation, we can construct it in an order in which the size of an edge clique cover roughly halves in each level of the recursion. We prove it by an edge direction argument in a tree corresponding to the minimal triangulation.

Lemma 12 *Let G be a graph with an edge clique cover $\mathcal{W}$. Any minimal triangulation H of G has a maximal clique Ω so that all blocks $C \in \mathcal{C}(G\backslash\Omega)$ have $|\mathcal{W}[C]| \leq |\mathcal{W}|/2$.*

Proof For any PMC Ω there can be at most one component $C \in \mathcal{C}(G\backslash\Omega)$ so that $|\mathcal{W}[C]| > |\mathcal{W}|/2$ because the sets $\mathcal{W}[C_i]$ over $C_i \in \mathcal{C}(G\backslash\Omega)$ correspond to disjoint subsets of $\mathcal{W}$. Let H be any minimal triangulation of G and pick arbitrary maximal clique Ω of H. While there is a component $C \in \mathcal{C}(G\backslash\Omega)$ such that $|\mathcal{W}[C]| > |\mathcal{W}|/2$, pick a maximal clique Ω of H such that $N(C) \subseteq \Omega \subseteq N[C]$. If this process stops, we have found the desired maximal clique Ω. Suppose the process does not stop. It considers an infinite sequence of blocks $C_1, C_2, \ldots$ with an associated infinite

sequence of PMCs $\Omega_1, \Omega_2, \ldots$ with $C_i \in \mathcal{C}(G \backslash \Omega_i)$. Consider two consecutive blocks C_i and C_{i+1} in this sequence such that C_{i+1} is not a subset of C_i, which exist because G is finite. Recall that $N(C_i) \subseteq \Omega_{i+1} \subseteq N[C_i]$. Because C_{i+1} is not a subset of C_i, we have that $N(C_{i+1}) \subseteq N(C_i)$, implying that C_i and C_{i+1} are two distinct components of $G \backslash N(C_i)$. Therefore the sets $\mathcal{W}[C_i]$ and $\mathcal{W}[C_{i+1}]$ are disjoint, implying that either of them has to be of size at most $|\mathcal{W}|/2$, which is a contradiction. □

We combine Proposition 8 and Lemma 12 into the following lemma.

Lemma 13 *Let G be a graph and $\mathcal{W}$ an edge clique cover of G. A graph H is a minimal triangulation of G if and only if* (1) $V(H) = V(G)$ *and* (2) *there is a PMC $\Omega \in \Pi(G)$ with $|\mathcal{W}[C_i]| \leq |\mathcal{W}|/2$ for all $C_i \in \mathcal{C}(G \backslash \Omega)$ and*

$$E(H) = \Omega^2 \cup \bigcup_{C_i \in \mathcal{C}(G \backslash \Omega)} E(H_i),$$

where H_i is a minimal triangulation of $R(C_i)$.

Proof Such a graph H is a minimal triangulation of G by standard results [5]. Let H be any minimal triangulation of G. By Lemma 12 there is a maximal clique Ω of H that satisfies $|\mathcal{W}[C_i]| \leq |\mathcal{W}|/2$ for all $C_i \in \mathcal{C}(G \backslash \Omega)$. By Proposition 8 the triangulation H can be constructed by the recursion from Ω. □

Algorithm 2 presents a polynomial space $O^*(9^{cc'})$ time algorithm for treewidth. The algorithms for other problems are similar. The algorithm implements the characterization of Lemma 13, with the observation that $\mathcal{W}[C] \cup \{N(C)\}$ is an edge clique cover of $R(C)$. The time complexity analysis of the algorithm reduces to a recursion equation resembling $t(cc') = 9^{cc'} + 3^{cc'} 2t(cc'/2)$.

Lemma 14 *There is an algorithm that given a graph G with an edge clique cover of size cc' determines the treewidth of G in polynomial space and $O^*(9^{cc'})$ time. If also a weight function $w : V(G)^2 \to \mathbb{R}_{\geq 0}$ is given, the algorithm determines the weighted minimum fill-in of G with respect to w. There is also algorithm that given a hypergraph $\mathcal{G}$ with m hyperedges determines both its generalized and fractional hypertreewidth in polynomial space and $O^*(9^m)$ time.*

Algorithm 2: Treewidth in polynomial space and $O^*(9^{cc'})$ time

```
Input  : Connected graph G, an edge clique cover W of G, and an integer k
Output: Whether the treewidth of G is at most k
for Ω ∈ Π(G) do
    if |Ω| ≤ k + 1 and |W[C_i]| ≤ |W|/2 for all C_i ∈ C(G\Ω) then
        ok ← True
        for C ∈ C(G\Ω) do
            if Treewidth(R(C), W[C] ∪ {N(C)}, k) = False then
                ok ← False
        if ok then return True
return False
```

Proof We use a recursive procedure that takes as an input a graph G and an edge clique cover $\mathcal{W}$ of it and returns the cost of an optimal triangulation of the graph. We use the characterization of minimal triangulations of Lemma 13 with the $O^*(9^{\mathtt{cc}})$ time polynomial space PMC enumeration algorithm of Lemma 5. Note that if C is a block, then $\mathcal{W}[C] \cup \{N(C)\}$ is an edge clique cover of $R(C)$. Therefore, when we recurse into a subproblem the value of $\mathtt{cc}'$ changes to at most $\mathtt{cc}'/2 + 1$.

We use $k = \mathtt{cc}'$ for clarity. We analyze the time complexity of the algorithm by induction on k. Our assumption is that the time complexity is $p(n)9^{k+3\log_2 k}k$, where $p(n)$ is a polynomial sufficiently large to cover all polynomial-time subroutines and to make the assumption true for small values of k. In particular we assume that the PMCs can be enumerated and their associated costs can be computed in $p(n)9^k$ time. We recurse from at most 3^k PMCs Ω, from each into $|\mathcal{C}(G\backslash\Omega)|$ subproblems with the new value of k being at most $k/2 + 1$. Because of the exponentiality of the algorithm the worst case is that we recurse into two subproblems from each PMC, each subproblem with the new value of k equal to $k/2 + 1$. Now, the time complexity is

$$\begin{aligned} &p(n)9^k + 3^k 2p(n)9^{k/2+1+3\log_2(k/2+1)}(k/2+1) \\ &\quad = p(n)9^k + p(n)9^{k+1+3\log_2(k+2)-3}(k+2) \\ &\quad \leq p(n)9^k + p(n)9^{k+3\log_2 k}(k+2)/9 \leq p(n)9^{k+3\log_2 k}k, \end{aligned}$$

which verifies the induction assumption. □

Theorem 5 follows.

The main idea to make the algorithm work when we do not know the edge clique cover is to just check if there are at most $3^{\mathtt{cc}}$ PMCs. The reason why the time complexity becomes a bit higher than in Lemma 14 is that we cannot assume that the worst case of branching from a PMC Ω results in only two subproblems. In particular, we cannot assume anything better than $\mathtt{cc}$ subproblems each with a minimum edge clique cover of size $\mathtt{cc}/2 + 1$.

Algorithm 3 presents a polynomial space $O^*(9^{\mathtt{cc}+O(\log^2 \mathtt{cc})})$ time algorithm for treewidth. The algorithm for weighted minimum fill-in is similar.

Algorithm 3: Treewidth in polynomial space and $O^*(9^{\mathtt{cc}+O(\log^2 \mathtt{cc})})$ time

```
Input  : Connected graph G, an integer cc, and an integer k
Output: True if the treewidth of G is at most k and G has an edge clique cover of size at most cc;
        true only if the treewidth of G is at most k
if |Π(G)| > 3^cc then return False for Ω ∈ Π(G) do
    if |Ω| ≤ k + 1 then
        if |C(G\Ω)| > cc then return False ok ← True
        for C ∈ C(G\Ω) do
            if Treewidth(R(C), cc/2 + 1, k) = False then
                ok ← False
        if ok then return True
return False
```

Lemma 15 *There is an algorithm that given a graph G and an integer $\mathtt{cc}$ uses polynomial space and $O^*(9^{\mathtt{cc}+O(\log^2 \mathtt{cc})})$ time and returns an integer t that is at least the treewidth of G. If also a weight function $w : V(G)^2 \to \mathbb{R}_{\geq 0}$ is given, the algorithm also returns a number f that is at least the weighted minimum fill-in of G with respect to w. If $\mathtt{cc}$ is at least the size of a minimum edge clique cover of G, then t is the treewidth of G and f is the weighted minimum fill-in of G with respect to w.*

Proof We use the same algorithm as in Lemma 14, but with the modification that instead of giving an edge clique cover as a parameter we give just the integer $\mathtt{cc}$. At the start we check if $|\Pi(G)| > 3^{\mathtt{cc}}$ in $O^*(9^{\mathtt{cc}})$ time, and therefore we can use the same $3^{\mathtt{cc}}$ bound on PMCs. Also, each time we recurse from a PMC Ω we check that $|\mathcal{C}(G\backslash\Omega)| \leq \mathtt{cc}$. If not, then G has no edge clique cover of size $\mathtt{cc}$.

We use $k = \mathtt{cc}$ and use induction on k. Assume that the time complexity is $p(n)9^{k+3\log_2 k}k^{3\log_2 k}$ with the same assumptions on $p(n)$ as in the proof of Lemma 14. We recurse from at most 3^k PMCs Ω, from each into $|\mathcal{C}(G\backslash\Omega)| \leq k$ subproblems with the value of k decreasing to $k/2+1$ in each. Now, the time complexity is

$$\begin{aligned}
&p(n)9^k + 3^k k p(n) 9^{k/2+1+3\log_2(k/2+1)}(k/2+1)^{3\log_2(k/2+1)} \\
&\quad = p(n)9^k + p(n)9^{k+1+3\log_2(k+2)-3}k(k/2+1)^{3\log_2(k+2)-3} \\
&\quad \leq p(n)9^k + p(n)9^{k+3\log_2 k}k^{3\log_2 k}/9 \leq p(n)9^{k+3\log_2 k}k^{3\log_2 k},
\end{aligned}$$

which verifies the induction assumption. □

Theorem 6 follows.

8 Tightness

We show that the bounds $S(\mathtt{cc}, 2)$ and $S(\mathtt{cc}, 3) + \mathtt{cc}$ for the numbers of minimal separators and PMCs are tight for all values of $\mathtt{cc}$.

Let us construct a graph $K_2^{\mathtt{cc}}$. Let $\mathcal{W}$ be a collection of size $\mathtt{cc}$ initially containing disjoint sets of size 1, that is, $\mathcal{W} = \{W_1, \ldots, W_{\mathtt{cc}}\}$ with $W_i = \{v_i\}$. For each pair $1 \leq i < j \leq \mathtt{cc}$ we insert an element $v_{i,j}$ into the sets W_i and W_j. Now, the vertex set of $K_2^{\mathtt{cc}}$ is $V(K_2^{\mathtt{cc}}) = \bigcup_{W\in\mathcal{W}} W$ and the edge set of $K_2^{\mathtt{cc}}$ is $E(K_2^{\mathtt{cc}}) = \bigcup_{W\in\mathcal{W}} W^2$. The collection $\mathcal{W}$ is therefore an edge clique cover of $K_2^{\mathtt{cc}}$.

Lemma 16 *The graph $K_2^{\mathtt{cc}}$ has at least $S(\mathtt{cc}, 2)$ minimal separators and at least $S(\mathtt{cc}, 3) + \mathtt{cc}$ PMCs.*

Proof For distinct subsets $\mathcal{W}' \subseteq \mathcal{W}$ the vertex sets $V(K_2^{\mathtt{cc}}, \mathcal{W}')$ are distinct because they can be identified by the inclusion of the v_i vertices. Let $\mathcal{W}'$ be any non-empty strict subset of $\mathcal{W}$. Both $K_2^{\mathtt{cc}}[V(K_2^{\mathtt{cc}}, \mathcal{W}')]$ and $K_2^{\mathtt{cc}}[V(K_2^{\mathtt{cc}}, \mathcal{W}\backslash\mathcal{W}')]$ are connected graphs. Any vertex in $V(K_2^{\mathtt{cc}})\backslash V(K_2^{\mathtt{cc}}, \mathcal{W}', \mathcal{W}\backslash\mathcal{W}')$ is of type $v_{i,j}$ and has a neighbor in both $V(K_2^{\mathtt{cc}}, \mathcal{W}')$ and $V(K_2^{\mathtt{cc}}, \mathcal{W}\backslash\mathcal{W}')$. Therefore, $V(K_2^{\mathtt{cc}})\backslash V(K_2^{\mathtt{cc}}, \mathcal{W}', \mathcal{W}\backslash\mathcal{W}')$ is a minimal separator and $V(K_2^{\mathtt{cc}}, \mathcal{W}')$ and $V(K_2^{\mathtt{cc}}, \mathcal{W}\backslash\mathcal{W}')$ are blocks of it. Therefore, the number of minimal separators of $K_2^{\mathtt{cc}}$ is at least the number of bipartitions of $\mathcal{W}$.

Take any tripartition $\{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3\}$ of $\mathcal{W}$. Note that distinct tripartitions define distinct sets $V(K_2^{\mathtt{cc}}, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3)$. We show that $\Omega = V(K_2^{\mathtt{cc}})\backslash(K_2^{\mathtt{cc}}, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3)$ is a PMC of $K_2^{\mathtt{cc}}$. Any vertex in Ω is of type $v_{i,j}$, with vertices v_i and v_j in different blocks $V(K_2^{\mathtt{cc}}, \mathcal{W}_p)$. Therefore, for any pair of vertices in Ω there is a common block that they are adjacent to, and therefore Ω satisfies the cliquish condition. A component $V(K_2^{\mathtt{cc}}, \mathcal{W}_1)$ cannot be full because there is a vertex $v_{i,j}$ that intersects cliques $W_i \in \mathcal{W}_2$ and $W_j \in \mathcal{W}_3$ and therefore is in the PMC but not in the neighborhood of $V(K_2^{\mathtt{cc}}, \mathcal{W}_1)$. Therefore, Ω satisfies the no full component condition. Therefore, the number of PMCs of $K_2^{\mathtt{cc}}$ that contain no vertices of type v_i is at least the number of tripartitions of $\mathcal{W}$.

We complete the proof by showing that for each v_i the set $N[v_i]$ is a PMC, contributing the term $\mathtt{cc}$ to the bound. The set $N[v_i]$ satisfies the no full component condition because v_i is not connected to any vertex outside of it. It satisfies the cliquish condition because it is a clique. □

9 Relation of Edge Clique Cover and Modular Width

In [16] an algorithm with time complexity $O^*(1.7347^{\mathtt{mw}})$, where $\mathtt{mw}$ is the modular width, was given for enumerating PMCs. In Sect. 1.5 we claimed that the parameters edge clique cover and modular width have a relation $\mathtt{mw} \leq 2^{\mathtt{cc}} - 2$, which is tight. Now we prove this claim.

A module of a graph G is a vertex set $X \subseteq V(G)$ such that for all $x \in V(G)\backslash X$, either $N(x) \cap X = X$ or $N(x) \cap X = \emptyset$. We use a recursive definition of modular width with four operations.

Definition 3 [16] A graph G has modular width $\mathtt{mw}$ if

1. $|V(G)| \leq 1$,
2. G is a disjoint union of two graphs with modular width at most $\mathtt{mw}$,
3. G is a join of two graphs G_1 and G_2 with modular width at most $\mathtt{mw}$, that is, $V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2) \cup (V(G_1) \times V(G_2))$, or
4. the vertices of G can be partitioned into $k \leq \mathtt{mw}$ modules $X_1, \ldots, X_k$, such that each $G[X_i]$ has modular width at most $\mathtt{mw}$.

Now we prove the inequality $\mathtt{mw} \leq 2^{\mathtt{cc}} - 2$.

Lemma 17 *If a graph has an edge clique cover of size $\mathtt{cc}$ then it has modular width at most $2^{cc} - 2$.*

Proof Let G be a graph with edge clique cover $\mathcal{W}$ of size $\mathtt{cc}$. If G has a vertex v with $\mathcal{W}[v] = \emptyset$, then we remove v with the operation 2. If G has a vertex v with $\mathcal{W}[v] = \mathcal{W}$, then we remove v with the operation 3. Now vertices of G can be partitioned to at most $2^{\mathtt{cc}} - 2$ classes based on $\mathcal{W}[v]$. These classes are cliques and they are modules because if $\mathcal{W}[v] = \mathcal{W}[u]$ then $N[v] = N[u]$. □

Next we prove the tightness of this inequality.

Lemma 18 *For each integer $cc \geq 3$ there is a graph with an edge clique cover of size cc and modular width $2^{cc} - 2$.*

Proof We construct a graph K^{cc}. Let $\mathcal{W}$ be a collection of size cc that will be the edge clique cover of K^{cc}. For every non-empty strict subset $\mathcal{W}' \subsetneq \mathcal{W}$ we create a vertex $v_{\mathcal{W}'}$ and insert it into each clique $W \in \mathcal{W}'$. The edges of K^{cc} are defined by $\mathcal{W}$, that is, by whether the subsets corresponding to the vertices intersect.

The graph K^{cc} is connected and its complement is also connected, so operations 2 and 3 cannot be applied. Next we prove that operation 4 cannot be applied with any integer $k < 2^{cc} - 2$. Suppose there is a non-trivial module M of K^{cc}, i.e., a module with size $2 \leq |M| < |V(K^{cc})|$. Let $v_{\mathcal{W}_1}$ and $v_{\mathcal{W}_2}$ be two distinct vertices in M with $|\mathcal{W}_1| \leq |\mathcal{W}_2|$. Now $v_{\mathcal{W}\setminus\mathcal{W}_1} \in M$, because $v_{\mathcal{W}\setminus\mathcal{W}_1} \in N(v_{\mathcal{W}_2})$, but $v_{\mathcal{W}\setminus\mathcal{W}_1} \notin N(v_{\mathcal{W}_1})$. Now, since M contains both $v_{\mathcal{W}_1}$ and $v_{\mathcal{W}\setminus\mathcal{W}_1}$ it must contain all vertices corresponding to single element subsets of $\mathcal{W}$ by a similar argument. Since M contains all vertices corresponding to single element subsets of $\mathcal{W}$ it also must contain all vertices of K^{cc} by a similar argument. □

10 Conclusion

We bounded the number of minimal separators and PMCs by the size of a minimum edge clique cover, obtaining new parameterized algorithms for problems in the PMC framework. The parameterization by edge clique cover is motivated by real applications of optimal triangulations, and our results provide theoretical corroboration on the observations of the efficiency of the PMC framework in practice. Prior to our work, only the work of Fomin et al. [16] considers FPT bounds for PMCs. Our work answers to their proposal for finding further FPT parameterizations for PMCs.

We showed that our combinatorial bounds are the best possible, implying also that our PMC enumeration algorithm is optimal up to polynomial factors with respect to the parameter cc. For individual problems it remains as an open problem to improve the algorithms or to prove conditional lower bounds assuming conjectures such as strong exponential time hypothesis [34]. We are not aware of other graph parameters whose value is always at most cc and for which single-exponential FPT bounds for minimal separators and PMCs exist. In particular, we note that graphs with vertex clique cover of size 2 can have an exponential number of minimal separators: the graph consisting of two cliques of size $n/2$ connected by a matching of $n/2$ edges has $2^{n/2} - 2$ minimal separators.

One combinatorial question closely related to our work is whether the bound $O(1.7347^n)$ for the number of PMCs can be improved in graphs where $cc \leq n$. This is motivated by moral graphs of Bayesian networks, for which the inequality holds. Also the question of finding other useful parameterizations for PMCs still remains for future work. Because of the fact that the parameter cc occurs naturally in multiple settings related to optimal triangulations, we expect that even more applications of our results could arise in future.

Acknowledgements I wish to thank Matti Järvisalo, Mikko Koivisto, Andreas Niskanen, and Juha Harviainen for helpful comments.

Funding Open access funding provided by University of Bergen (incl Haukeland University Hospital).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Berry, A., Bordat, J.P., Cogis, O.: Generating all the minimal separators of a graph. Int. J. Found. Comput. Sci. **11**(3), 397–403 (2000). https://doi.org/10.1142/S0129054100000211
2. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Fourier meets Möbius: fast subset convolution. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing, pp. 67–74. ACM (2007). https://doi.org/10.1145/1250790.1250801
3. Bodlaender, H.L., Cai, L., Chen, J., Fellows, M.R., Telle, J.A., Marx, D.: Open problems in parameterized and exact computation—IWPEC 2006. Technical Report UU-CS-2006-052, Department of Information and Computing Sciences, Utrecht University (2006). https://dspace.library.uu.nl/handle/1874/22186
4. Bordewich, M., Huber, K.T., Semple, C.: Identifying phylogenetic trees. Discrete Math. **300**(1–3), 30–43 (2005). https://doi.org/10.1016/j.disc.2005.06.015
5. Bouchitté, V., Todinca, I.: Treewidth and minimum fill-in: grouping the minimal separators. SIAM J. Comput. **31**(1), 212–232 (2001). https://doi.org/10.1137/S0097539799359683
6. Bouchitté, V., Todinca, I.: Listing all potential maximal cliques of a graph. Theor. Comput. Sci. **276**(1–2), 17–32 (2002). https://doi.org/10.1016/S0304-3975(01)00007-X
7. Bron, C., Kerbosch, J.: Algorithm 457: finding all cliques of an undirected graph. Commun. ACM **16**(9), 575–577 (1973)
8. Chapelle, M., Liedloff, M., Todinca, I., Villanger, Y.: Treewidth and pathwidth parameterized by the vertex cover number. Discrete Appl. Math. **216**, 114–129 (2017). https://doi.org/10.1016/j.dam.2014.12.012
9. Conati, C., Gertner, A.S., VanLehn, K., Druzdzel, M.J.: On-line student modeling for coached problem solving using Bayesian networks. In: User Modeling, CISM, vol. 383, pp. 231–242. Springer (1997). https://doi.org/10.1007/978-3-7091-2670-7_24
10. Cygan, M., Pilipczuk, M., Pilipczuk, M.: Known algorithms for edge clique cover are probably optimal. SIAM Journal on Computing **45**(1), 67–83 (2016). https://doi.org/10.1137/130947076
11. Dell, H., Komusiewicz, C., Talmon, N., Weller, M.: The PACE 2017 parameterized algorithms and computational experiments challenge: the second iteration. In: 12th International Symposium on Parameterized and Exact Computation, LIPIcs, vol. 89, pp. 30:1–30:12. Schloss Dagstuhl—Leibniz-Zentrum für Informatik (2017). https://doi.org/10.4230/LIPIcs.IPEC.2017.30
12. Fischl, W., Gottlob, G., Longo, D.M., Pichler, R.: HyperBench: a benchmark and tool for hypergraphs and empirical findings. In: Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, pp. 464–480 (2019). https://doi.org/10.1145/3294052.3319683
13. Fischl, W., Gottlob, G., Pichler, R.: General and fractional hypertree decompositions: hard and easy cases. In: Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, pp. 17–32 (2018). https://doi.org/10.1145/3196959.3196962
14. Fomin, F.V., Golovach, P.A., Raymond, J.: On the tractability of optimization problems on H-graphs. Algorithmica (2020). https://doi.org/10.1007/s00453-020-00692-9

15. Fomin, F.V., Kratsch, D., Todinca, I., Villanger, Y.: Exact algorithms for treewidth and minimum fill-in. SIAM J. Comput. **38**(3), 1058–1079 (2008). https://doi.org/10.1137/050643350
16. Fomin, F.V., Liedloff, M., Montealegre, P., Todinca, I.: Algorithms parameterized by vertex cover and modular width, through potential maximal cliques. Algorithmica **80**(4), 1146–1169 (2018). https://doi.org/10.1007/s00453-017-0297-1
17. Fomin, F.V., Todinca, I., Villanger, Y.: Large induced subgraphs via triangulations and CMSO. SIAM J. Comput. **44**(1), 54–87 (2015). https://doi.org/10.1137/140964801
18. Furuse, M., Yamazaki, K.: A revisit of the scheme for computing treewidth and minimum fill-in. Theor. Comput. Sci. **531**, 66–76 (2014). https://doi.org/10.1016/j.tcs.2014.03.013
19. Gottlob, G., Miklós, Z., Schwentick, T.: Generalized hypertree decompositions: NP-hardness and tractable variants. J. ACM **56**(6), 30:1-30:32 (2009). https://doi.org/10.1145/1568318.1568320
20. Grohe, M., Marx, D.: Constraint solving via fractional edge covers. ACM Trans. Algorithms **11**(1), 4:1-4:20 (2014). https://doi.org/10.1145/2636918
21. Gysel, R.: Minimal triangulation algorithms for perfect phylogeny problems. In: Language and Automata Theory and Applications—8th International Conference, LNCS, vol. 8370, pp. 421–432. Springer (2014). https://doi.org/10.1007/978-3-319-04921-2_34
22. Hasegawa, M., Kishino, H., Yano, T.: Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. J. Mol. Evol. **22**(2), 160–174 (1985)
23. Heggernes, P., Mancini, F., Nederlof, J., Villanger, Y.: A parameterized algorithm for chordal sandwich. In: Proceedings of 7th International Conference on Algorithms and Complexity, LNCS, vol. 6078, pp. 120–130. Springer (2010). https://doi.org/10.1007/978-3-642-13073-1_12
24. IBM ILOG: CPLEX optimizer 12.7.1 (2017). https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer
25. Jensen, F.V., Jensen, F.: Optimal junction trees. In: Proceedings of the 10th Annual Conference on Uncertainty in Artificial Intelligence, pp. 360–366. Morgan Kaufmann (1994)
26. Jensen, F.V., Nielsen, T.D.: Bayesian Networks and Decision Graphs. Springer, Berlin (2007). https://doi.org/10.1007/978-0-387-68282-2
27. Kannan, S., Warnow, T.: A fast algorithm for the computation and enumeration of perfect phylogenies. SIAM J. Comput. **26**(6), 1749–1763 (1997). https://doi.org/10.1137/S0097539794279067
28. Korhonen, T.: Finding optimal tree decompositions. Master's thesis, University of Helsinki (2020). http://urn.fi/URN:NBN:fi:hulib-202006173010
29. Korhonen, T.: Finding optimal triangulations parameterized by edge clique cover. In: 15th International Symposium on Parameterized and Exact Computation (IPEC 2020), Leibniz International Proceedings in Informatics (LIPIcs), vol. 180, pp. 22:1–22:18. Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.IPEC.2020.22
30. Korhonen, T., Berg, J., Järvisalo, M.: Solving graph problems via potential maximal cliques: an experimental evaluation of the Bouchitté–Todinca algorithm. ACM J. Exp. Algorithm. **24**(1), 1.9:1–1.9:19 (2019). https://doi.org/10.1145/3301297
31. Korhonen, T., Järvisalo, M.: Finding most compatible phylogenetic trees over multi-state characters. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence, pp. 1544–1551. AAAI Press (2020). https://doi.org/10.1609/aaai.v34i02.5514
32. Liedloff, M., Montealegre, P., Todinca, I.: Beyond classes of graphs with "few" minimal separators: FPT results through potential maximal cliques. Algorithmica **81**(3), 986–1005 (2019). https://doi.org/10.1007/s00453-018-0453-2
33. Lokshtanov, D.: On the complexity of computing treelength. Discrete Appl. Math. **158**(7), 820–827 (2010). https://doi.org/10.1016/j.dam.2009.10.007
34. Lokshtanov, D., Marx, D., Saurabh, S.: Lower bounds based on the exponential time hypothesis. Bull. EATCS **105**, 41–72 (2011)
35. Moll, L., Tazari, S., Thurley, M.: Computing hypergraph width measures exactly. Inf. Process. Lett. **112**(6), 238–242 (2012). https://doi.org/10.1016/j.ipl.2011.12.002
36. Ravid, N., Medini, D., Kimelfeld, B.: Ranked enumeration of minimal triangulations. In: Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, pp. 74–88. ACM (2019). https://doi.org/10.1145/3294052.3319678
37. Ringe, D., Warnow, T., Taylor, A.: Indo-European and computational cladistics. Trans. Philol. Soc. **100**(1), 59–129 (2002). https://doi.org/10.1111/1467-968X.00091
38. Scutari, M.: Learning Bayesian networks with the bnlearn R package. J. Stat. Softw. **35**(3), 1–22 (2010). https://doi.org/10.18637/jss.v035.i03

39. Semple, C., Steel, M.: Phylogenetics. Oxford University Press, Oxford (2003)
40. Takata, K.: Space-optimal, backtracking algorithms to list the minimal vertex separators of a graph. Discrete Appl. Math. **158**(15), 1660–1667 (2010). https://doi.org/10.1016/j.dam.2010.05.013
41. Tamaki, H.: Computing treewidth via exact and heuristic lists of minimal separators. In: Proceedings of the Special Event on Analysis of Experimental Algorithms, LNCS, vol. 11544, pp. 219–236. Springer (2019). https://doi.org/10.1007/978-3-030-34029-2_15
42. Tamaki, H.: Positive-instance driven dynamic programming for treewidth. J. Comb. Optim. **37**(4), 1283–1311 (2019). https://doi.org/10.1007/s10878-018-0353-z