Exploring the Subexponential Complexity of Completion Problems

PÅL GRØNÅS DRANGE, FEDOR V. FOMIN, MICHAŁ PILIPCZUK, and YNGVE VILLANGER, Department of Informatics, University of Bergen

Let $\mathcal{F}$ be a family of graphs. In the $\mathcal{F}$-COMPLETION problem, we are given an n-vertex graph G and an integer k as input, and asked whether at most k edges can be added to G so that the resulting graph does not contain a graph from $\mathcal{F}$ as an induced subgraph. It was shown recently that two special cases of $\mathcal{F}$-COMPLETION, namely, (i) the problem of completing into a chordal graph known as MINIMUM FILL-IN (SIAM J. Comput. 2013), which corresponds to the case of $\mathcal{F} = \{C_4, C_5, C_6, \ldots\}$, and (ii) the problem of completing into a split graph (Algorithmica 2015), that is, the case of $\mathcal{F} = \{C_4, 2K_2, C_5\}$, are solvable in parameterized subexponential time $2^{\mathcal{O}(\sqrt{k}\log k)}n^{\mathcal{O}(1)}$. The exploration of this phenomenon is the main motivation for our research on $\mathcal{F}$-COMPLETION.

In this article, we prove that completions into several well-studied classes of graphs without long induced cycles and paths also admit parameterized subexponential time algorithms by showing that:

—The problem TRIVIALLY PERFECT COMPLETION, which is $\mathcal{F}$-COMPLETION for $\mathcal{F} = \{C_4, P_4\}$, a cycle and a path on four vertices, is solvable in parameterized subexponential time $2^{\mathcal{O}(\sqrt{k}\log k)}n^{\mathcal{O}(1)}$.
—The problems known in the literature as PSEUDOSPLIT COMPLETION, the case in which $\mathcal{F} = \{2K_2, C_4\}$, and THRESHOLD COMPLETION, in which $\mathcal{F} = \{2K_2, P_4, C_4\}$, are also solvable in time $2^{\mathcal{O}(\sqrt{k}\log k)}n^{\mathcal{O}(1)}$.

We complement our algorithms for $\mathcal{F}$-COMPLETION with the following lower bounds:

—For $\mathcal{F} = \{2K_2\}$, $\mathcal{F} = \{C_4\}$, $\mathcal{F} = \{P_4\}$, and $\mathcal{F} = \{2K_2, P_4\}$, $\mathcal{F}$-COMPLETION cannot be solved in time $2^{o(k)}n^{\mathcal{O}(1)}$ unless the Exponential Time Hypothesis (ETH) fails.

Our upper and lower bounds provide a complete picture of the subexponential parameterized complexity of $\mathcal{F}$-COMPLETION problems for any $\mathcal{F} \subseteq \{2K_2, C_4, P_4\}$.

Categories and Subject Descriptors: G.2.2 [**Graph Theory**]: Graph Algorithms

General Terms: Algorithms, Parameterized, Subexponential, Completion

Additional Key Words and Phrases: Edge completion, modification, subexponential parameterized complexity

ACM Reference Format:
Pål Grønås Drange, Fedor V. Fomin, Michał Pilipczuk, and Yngve Villanger. 2015. Exploring the subexponential complexity of completion problems. ACM Trans. Comput. Theory 7, 4, Article 14 (August 2015), 38 pages.
DOI: http://dx.doi.org/10.1145/2799640

A preliminary version of this article appeared in the proceedings of STACS 2014 [Drange et al. 2014].
The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement No. 267959.
Authors' address: P. G. Drange (corresponding author), F. V. Fomin, M. Pilipczuk, and Y. Villanger, Department of Informatics, University of Bergen, Postboks 7803, 5020 Bergen, Norway; emails: {Pal.Drange, Fedor.Fomin}@ii.uib.no, michal.pilipczuk@mimuw.edu.pl, Yngve.Villanger@uib.no.

ACM 1942-3454/2015/08-ART14 $15.00
DOI: http://dx.doi.org/10.1145/2799640

1. INTRODUCTION

Let $\mathcal{F}$ be a family of graphs. In this article, we study the following $\mathcal{F}$-COMPLETION problem.

$\mathcal{F}$-COMPLETION	
Input:	A graph $G = (V, E)$ and a nonnegative integer k.
Parameter:	k
Question:	Does there exist a supergraph $H = (V, E \cup S)$ of G, such that $\lvert S\rvert \leq k$ and H contains no graph from $\mathcal{F}$ as an induced subgraph?

$\mathcal{F}$-COMPLETION problems form a subclass of graph modification problems for which one is asked to apply a bounded number of changes to an input graph to obtain a graph with some specified property. Graph modification problems arise naturally in many branches of science and have been studied extensively during the past 40 years. Interestingly enough, despite the long history of the problem, there is no known classification of graph classes $\mathcal{F}$-free into those for which $\mathcal{F}$-COMPLETION is solvable in polynomial time and those for which the problem is NP-complete [Yannakakis 1981b; Mancini 2008; Burzyn et al. 2006].

One of the motivations for studying algorithms for completion problems in graphs comes from their intimate connections to different width parameters. For example, the treewidth of a graph, one of the most fundamental graph parameters, is the minimum, over all possible completions into a chordal graph, of the maximum clique size minus one [Bodlaender 1998]. The treedepth of a graph—also known as the vertex ranking number, the ordered chromatic number, and the minimum elimination tree height—plays a crucial role in the theory of sparse graphs developed by Nešetřil and Ossona de Mendez [2012]. Mirroring the connection between treewidth and chordal graphs, the treedepth of a graph can be defined as the largest clique size in a completion to a trivially perfect graph. This can be observed by recalling that the definition of the treedepth of G is the minimum height of a rooted forest whose closure contains G as a subgraph. Similarly, the vertex cover number of a graph is equal to the minimum of the largest clique size taken over all completions to a threshold graph, minus one.

Recent developments have led to subexponential parameterized algorithms for the problems INTERVAL COMPLETION [Bliznets et al. 2014a] and PROPER INTERVAL COMPLETION [Bliznets et al. 2014b]. Both problems have strong connections to width parameters just like the ones mentioned earlier: The pathwidth of a graph is the minimum of the maximum clique size over all interval completions of the graph, minus one, whereas the bandwidth mirrors this relation for proper interval completions of the graph [Kaplan and Shamir 1996].

Parameterized algorithms for completion problems. For a long time in parameterized complexity, the main focus of studies in $\mathcal{F}$-COMPLETION was for the case when $\mathcal{F}$ was an infinite family of graphs, for example, MINIMUM FILL-IN or INTERVAL COMPLETION [Kaplan et al. 1999; Natanzon et al. 2000; Villanger et al. 2009]. This was mainly due to the fact that when $\mathcal{F}$ is a finite family, $\mathcal{F}$-COMPLETION is solvable on an n-vertex graph in time $f(k) \cdot n^{\mathcal{O}(1)}$ for some function f by a simple branching argument; this was first observed by Cai [1996]. More precisely, if the maximum number of nonedges in a graph from $\mathcal{F}$ is d, then the corresponding $\mathcal{F}$-COMPLETION is solvable in time $d^k \cdot n^{\mathcal{O}(1)}$. The interest in $\mathcal{F}$-COMPLETION problems started to increase with the advance of kernelization. It appeared that from the perspective of kernelization, even for the case of finite families $\mathcal{F}$, the problem is far from trivial. Guo [2007] initiated the study of kernelization algorithms for $\mathcal{F}$-COMPLETION in the case when the forbidden set $\mathcal{F}$ contains the graph

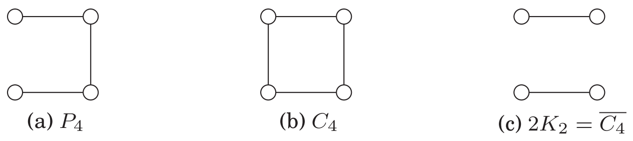


Fig. 1. Forbidden induced subgraphs. Trivially perfect graphs are $\{C_4, P_4\}$-free, threshold graphs are $\{2K_2, P_4, C_4\}$-free, and cographs are P_4-free.

C_4; see Figure 1. (In fact, Guo considered edge deletion problems, but they are polynomial time equivalent to completion problems to the complements of the forbidden induced subgraphs.) In the literature, the most studied graph classes containing no induced C_4 are the split graphs, that is, $\{2K_2, C_4, C_5\}$-free graphs, threshold graphs, that is, $\{2K_2, P_4, C_4\}$-free graphs, and $\{C_4, P_4\}$-free graphs, that is, trivially perfect graphs [Brandstädt et al. 1999]. Guo obtained polynomial kernels for the completion problems for chain graphs, split graphs, threshold graphs, and trivially perfect graphs, concluding that, as a consequence of his polynomial kernelization, the corresponding $\mathcal{F}$-Completion problems: Chain Completion, Split Completion, Threshold Completion, and Trivially Perfect Completion are solvable in times $\mathcal{O}(2^k + mnk)$, $\mathcal{O}(5^k + m^4 n)$, $\mathcal{O}(4^k + kn^4)$, and $\mathcal{O}(4^k + kn^4)$, respectively.

The work on kernelization of $\mathcal{F}$-Completion problems was continued by Kratsch and Wahlström [2013], who showed that there exists a set $\mathcal{F}$ consisting of one graph on seven vertices for which $\mathcal{F}$-Completion does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly. Guillemot et al. [2013] showed that Cograph Completion, that is, the case $\mathcal{F} = \{P_4\}$, admits a polynomial kernel, while for $\mathcal{F} = \{\overline{P_{13}}\}$, the complement of a path on 13 vertices, $\mathcal{F}$-Completion has no polynomial kernel. These results were significantly improved by Cai and Cai [2015]: For $\mathcal{F} = \{P_\ell\}$ or $\mathcal{F} = \{C_\ell\}$, the problems $\mathcal{F}$-Completion and $\mathcal{F}$-Edge Deletion admit a polynomial kernel if and only if the forbidden graph has at most three edges.

It was shown recently that for some choices of $\mathcal{F}$, the $\mathcal{F}$-Completion problem is solvable in *subexponential* time. The exploration of this phenomenon is the main motivation for our research on this problem. The last chapter of Flum and Grohe's textbook on parameterized complexity theory [Flum and Grohe 2006, Chapter 16] concerns subexponential fixed-parameter tractability and the complexity class SUBEPT, which, loosely speaking—we skip here some technical conditions—is the class of problems solvable in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$, where n is the input length and k is the parameter. Until recently, the only notable examples of problems in SUBEPT were problems on planar graphs, and more generally, on graphs excluding some fixed graph as a minor [Demaine et al. 2005]. In 2009, Alon et al. [2009] used a novel application of color coding, dubbed chromatic coding, to show that parameterized Feedback Arc Set in Tournaments is in SUBEPT. As Flum and Grohe [2006] observed, for most natural parameterized problems, already the classical NP-hardness reductions can be used to refute the existence of subexponential parameterized algorithms unless the following well-known complexity hypothesis formulated by Impagliazzo et al. [2001] fails.

Exponential Time Hypothesis *(ETH). There exists a positive real number s such that 3-CNF-SAT with n variables cannot be solved in time* 2^{sn}.

Thus, it is very likely that the majority of parameterized problems are not solvable in subexponential parameterized time, and until very recently no natural parameterized problem solvable in subexponential parameterized time on general graphs was known. A subset of the authors recently showed that Minimum Fill-in, also known as Chordal Completion, which is equivalent to $\mathcal{F}$-Completion with $\mathcal{F}$ consisting of cycles of length at least four, is in SUBEPT [Fomin and Villanger 2013], simultaneously establishing

Obstruction set $\mathcal{F}$	Graph class name	Complexity
$C_4, C_5, C_6, \ldots$	Chordal	SUBEPT [Fomin and Villanger 2013]
C_4, P_4	Trivially Perfect	SUBEPT (Theorem 2.1)
$2K_2, C_4, C_5$	Split	SUBEPT [Ghosh et al. 2015]
$2K_2, C_4, P_4$	Threshold	SUBEPT (Theorem 3.1)
$2K_2, C_4$	Pseudosplit	SUBEPT (Theorem 4.3)
$\overline{P_3}, K_t, t = o(k)$	Co-t-cluster	SUBEPT [Fomin et al. 2014]
$\overline{P_3}$	Co-cluster	E [Komusiewicz and Uhlmann 2012]
$2K_2$	$2K_2$-free	E (Theorem 5.1)
C_4	C_4-free	E (Theorem 5.4)
P_4	Cograph	E (Theorem 5.9)
$2K_2, P_4$	Co-Trivially Perfect	E (Theorem 5.15)

Fig. 2. Known subexponential complexity of $\mathcal{F}$-COMPLETION for different sets $\mathcal{F}$. All problems in this table are NP-hard and in FPT. The entry SUBEPT means that the problem is solvable in subexponential time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$, whereas E means that the problem is not solvable in subexponential time unless ETH fails.

that CHAIN COMPLETION (completing to a chain graph) is solvable in subexponential time. Later, Ghosh et al. [2015] showed that SPLIT COMPLETION is solvable in subexponential time. In contrast, Komusiewicz and Uhlmann [2012] showed that an edge modification problem known as CLUSTER DELETION does not belong to SUBEPT unless ETH fails. Note that CLUSTER DELETION is equivalent to $\mathcal{F}$-COMPLETION when $\mathcal{F} = \{\overline{P_3}\}$, the complement of the path P_3. However, it is interesting to note that by a result of Fomin et al. [2014], CLUSTER DELETION INTO t CLUSTERS, that is, the complement problem for $\mathcal{F}$-COMPLETION for $\mathcal{F} = \{\overline{P_3}, K_t\}$, can be solved in time $2^{\mathcal{O}(\sqrt{tk})} \cdot n^{\mathcal{O}(1)}$, thus is in SUBEPT for $t = o(k)$.

Our results. In this work, we extend the class of $\mathcal{F}$-COMPLETION problems admitting subexponential time algorithms (see Figure 2). Our main algorithmic result is the following:

TRIVIALLY PERFECT COMPLETION is solvable in time $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$ and is thus in SUBEPT.

This problem is the $\mathcal{F}$-COMPLETION problem for $\mathcal{F} = \{C_4, P_4\}$.

At a very high level, our algorithm is based on the same strategy as the algorithm for completion into chordal graphs [Fomin and Villanger 2013]. As in that algorithm, we enumerate subexponentially many special objects, here called *trivially perfect potential maximal cliques*; these are the maximal cliques in some minimal completion into a trivially perfect graph. Once we enumerate these objects, we apply dynamic programming in order to find an optimal completion. Here the similarities end, however. To enumerate trivially perfect potential maximal cliques (henceforth referred to as only *potential maximal cliques*) for trivially perfect graphs, we have to use completely different structural properties from those used for the case of chordal graphs.

We also show that, within the same running time, the $\mathcal{F}$-COMPLETION problem is solvable for $\mathcal{F} = \{2K_2, C_4\}$ and $\mathcal{F} = \{2K_2, P_4, C_4\}$. This corresponds to completion into threshold and pseudosplit graphs, respectively. Combined with the results of Fomin and Villanger [2013] and Ghosh et al. [2015], this implies that all four problems considered by Guo [2007] are in SUBEPT, in addition to admitting a polynomial kernel. We finally complement our algorithmic findings by showing the following:

For $\mathcal{F} = \{2K_2\}$, $\mathcal{F} = \{C_4\}$, $\mathcal{F} = \{P_4\}$, and $\mathcal{F} = \{2K_2, P_4\}$, the $\mathcal{F}$-COMPLETION problem cannot be solved in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ unless ETH fails.

Thus, we obtain a complete classification for all $\mathcal{F} \subseteq \{2K_2, P_4, C_4\}$.

Organization of the article. In Section 2, we present some structural results about trivially perfect graphs and their completions, and describe the main result of the article: an algorithm that solves TRIVIALLY PERFECT COMPLETION in subexponential time. This section also contains some structural results on trivially perfect graphs that might be interesting on their own. In Sections 3 and 4, we present subexponential time algorithms for THRESHOLD COMPLETION and PSEUDOSPLIT COMPLETION, respectively.

In Section 5, we present lower bounds on $\mathcal{F}$-COMPLETION when $\mathcal{F}$ is $\{2K_2\}$, $\{C_4\}$, $\{P_4\}$, and $\{2K_2, P_4\}$. In Section 6, we give some concluding remarks and state some interesting remaining questions and future directions.

Notation and preliminaries on parameterized complexity. We consider only finite simple undirected graphs. We use n_G to denote the number of vertices and m_G the number of edges in a graph G. If $G = (V, E)$ is a graph, and $A, B \subseteq V$, we write $E(A, B)$ for the set of edges with one endpoint in A and the other in B. We write $E(A) = E(A, A)$ for the set of edges with both endpoints in A. We denote by $m_A = m_{G[A]}$ the number of edges inside A, that is, $m_A = |E(A)|$. By $G[A]$ for a set $A \subseteq V(G)$, we mean the induced subgraph G' of G on vertex set A, where for u and v in A, we have that $uv \in E(G')$ when $uv \in E(G)$. If a graph is a bipartite graph, we use $G = (A, B, E)$ to denote that G has bipartitions A and B.

Given a graph $G = (V, E)$, we use $N_G(v)$ for a vertex $v \in V$ to denote the set of neighbors of v in G. We write $N_G[v]$ to mean the set $N_G(v) \cup \{v\}$. For sets of vertices $U \subseteq V$, we write $N_G(U)$ to denote the open neighborhood $\bigcup_{v \in U}(N_G(v)) \setminus U$, and $N_G[U] = N_G(U) \cup U$ to denote the closed neighborhood. For a set of pairs of vertices S, we write $G + S = (V, E \cup S)$ and if $U \subseteq V$ is a set of vertices, then $G - U = G[V \setminus U]$. We will skip the subscripts when this will not cause any confusion. We will write $G \cong H$ to denote that G is isomorphic to H. For a natural number $n \in \mathbb{N}$, we write $[n]$ to denote the set $[n] = \{1, 2, 3, \dots\}$.

A *universal vertex* in a graph G is a vertex v such that $N[v] = V(G)$. Let (G) denote the set of universal vertices of G. Observe that (G), when nonempty, is always a clique, and we will refer to it as the *(maximal) universal clique*. Maximal universal cliques in trivially perfect graphs play an important role in the design of our algorithm; they are the main building blocks that we will use to achieve the algorithm.

We state here a simplified definition of parameterized problems, kernels, and the class of parameterized subexponential time algorithms. A *parameterized problem* Π is a problem whose input is a pair (x, k), where $k \in \mathbb{N}$. The problem Π is *fixed-parameter tractable*, thus belongs to the class FPT if there is an algorithm that solves this problem in time $f(k) \cdot x^{O(1)}$ for some function f that depends only on k. A *kernelization algorithm* for Π is a polynomial time algorithm that on input (x, k) gives an output (x', k') such that $|x'| \leq g(k)$ and $k' \leq g(k)$ for some function g that depends only on k, and such that (x, k) is a yes instance for Π if and only if (x', k') is a yes instance for Π. We call the output the *kernel*. We say a problem admits a *polynomial kernel* if the function g is polynomial. We refer to the seminal book on parameterized complexity theory of Flum and Grohe [2006] for more on parameterized complexity.

The complexity class SUBEPT is a subset of FPT; it is the class of problems Π for which there exists an algorithm with running time $2^{o(k)} \cdot n^{O(1)}$. That is, the parameter function f is *subexponential*. Note that if the exponential time hypothesis is true, then SUBEPT $\subsetneq$ FPT.

2. COMPLETION TO TRIVIALLY PERFECT GRAPHS

In this section, we study the TRIVIALLY PERFECT COMPLETION problem, which is $\mathcal{F}$-COMPLETION for $\mathcal{F} = \{C_4, P_4\}$. The decision version of this problem was shown to be NP-complete by Yannakakis [1981a]. Since trivially perfect graphs are characterized by a finite set of forbidden induced subgraphs, it follows from Cai [1996] that the problem also is fixed-parameter tractable, that is, it belongs to the class FPT.

The main result of this section is the following theorem:

THEOREM 2.1. *For an input* (G, k)*,* TRIVIALLY PERFECT COMPLETION *is solvable in time* $2^{\mathcal{O}(\sqrt{k}\log k)} + n^{\mathcal{O}(1)}$*, where* $n = |V(G)|$*.*

Throughout this section, an edge set S is called a *completion* for G if $G+S$ is trivially perfect. Furthermore, a completion S is called a *minimal completion* for G if no proper subset of S is a completion for G. An outline of our algorithm for TRIVIALLY PERFECT COMPLETION is

Step A: On input (G, k), we first apply a kernelization algorithm to obtain in polynomial time an equivalent instance with $k^{\mathcal{O}(1)}$ vertices. Due to this preprocessing step, we may assume without loss of generality that we work on an instance (G, k) with $|V(G)| = k^{\mathcal{O}(1)}$.
Step B: Assuming that our input instance has $k^{\mathcal{O}(1)}$ vertices, we show how to generate all special vertex subsets of the kernel, which we call *vital potential maximal cliques* in time $2^{\mathcal{O}(\sqrt{k}\log k)}$. A vital potential maximal clique $\Omega \subseteq V(G)$ is a vertex subset that is a maximal clique in some minimal completion of size at most k.
Step C: Using dynamic programming, we show how to compute an optimal solution or to conclude that (G, k) is a no instance, in time polynomial in the number of vital potential maximal cliques.

2.1. Structure of Trivially Perfect Graphs

Apart from the aforementioned characterization by forbidden induced subgraphs, which is an inherently local characterization, several other equivalent definitions of trivially perfect graphs are known. These definitions reveal more structural properties of this graph class that are essential for developing our algorithm. Therefore, before proceeding with the proof of Theorem 2.1, we establish a number of results on the global structure of trivially perfect graphs and minimal completions.

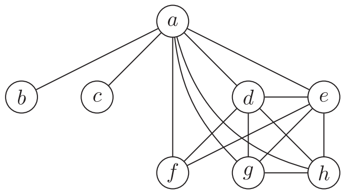
Trivially perfect graphs have a rooted decomposition tree, which we call a *universal clique decomposition*, in which each node corresponds to a maximal set of vertices that are all universal for the graph induced by the vertices in the subtree rooted at this node. This decomposition is similar to that of a *treedepth decomposition*. We refer to Figure 3 for an example of the concepts that we introduce next. The following recursive definition is often used as an alternative definition of trivially perfect graphs.

PROPOSITION 2.2 [JING-HO ET AL. 1996]. *The class of trivially perfect graphs can be defined recursively as follows:*

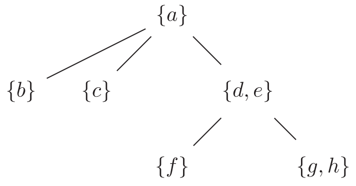
—K_1 *is a trivially perfect graph.*
—*Adding a universal vertex to a trivially perfect graph results in a trivially perfect graph.*
—*The disjoint union of two trivially perfect graphs is a trivially perfect graph.*

Let T be a rooted tree and t be a node of T. We denote by T_t the maximal subtree of T rooted in t. We can now use the universal clique uni(G) of a trivially perfect graph $G = (V, E)$ to make a decomposition structure.

Definition 2.3 (*Universal Clique Decomposition*). A *universal clique decomposition* of a connected trivially perfect graph $G = (V, E)$ is a pair $(T = (V_T, E_T), \mathcal{B} = \{B_t\}_{t \in V_T})$, where T is a rooted tree and $\mathcal{B}$ is a partition of the vertex set V into disjoint nonempty subsets, such that



(a) A trivially perfect graph G.



(b) The universal clique decomposition of G with bags as labels.

Label (bag)	Block	Tail
$\{a\}$	$(\{a\}, V)$	$\{a\}$
$\{b\}$	$(\{b\}, \{b\})$	$\{a, b\}$
$\{c\}$	$(\{c\}, \{c\})$	$\{a, c\}$
$\{d, e\}$	$(\{d, e\}, \{d, e, f, g, h\})$	$\{a, d, e\}$
$\{f\}$	$(\{f\}, \{f\})$	$\{a, d, e, f\}$
$\{g, h\}$	$(\{g, h\}, \{g, h\})$	$\{a, d, e, g, h\}$

(c) Table of the bags with corresponding blocks and tails.

Fig. 3. In the first figure, we have a trivially perfect graph, and in the second, the *universal clique decomposition* of the graph with bags as labels. Finally, we have a table of the bags as well as corresponding blocks and tails. Notice that for a block (B, D) and tail Q, $B \subseteq D$ and $B \subseteq Q$. Furthermore, for any leaf block, it holds that $B = D$, and for the root block it holds that $D = V$.

—if $vw \in E(G)$ and $v \in B_t$ and $w \in B_s$, then s and t are on a path from a leaf to the root, with possibly $s = t$, and
—for every node $t \in V_T$, the set of vertices B_t is the maximal universal clique in the subgraph $G[\bigcup_{s\in V(T_t)} B_s]$.

We call the vertices of T *nodes* and the sets in $\mathcal{B}$ *bags* of the universal clique decomposition $(T, \mathcal{B})$. By slightly abusing the notation, we often do not distinguish between a node and its corresponding bag. Note that, in a universal clique decomposition, every nonleaf node t has at least two children, since otherwise the universal clique contained in the bag corresponding to t would not be maximal.

LEMMA 2.4. *A connected graph G admits a universal clique decomposition if and only if it is trivially perfect. Moreover, such a decomposition is unique up to isomorphisms.*

PROOF. In the reverse direction, we proceed by induction on the structure of trivially perfect graphs using Proposition 2.2. The base case is when we have one vertex K_1, which is a trivially perfect graph and also admits a unique universal clique decomposition. The induction step is when we add a vertex v, and by the structure of trivially perfect graphs, v is a universal vertex. Either we add a universal vertex to a connected trivially perfect graph, in which case we simply add the vertex to the root bag, or we add a universal vertex to the disjoint union of two or more trivially perfect graphs. In the latter case, we create a new tree, with r_v being the root connected to the root of each of the trees for the disjoint union, and with $B_v = \{v\}$ its corresponding bag. Since v is the only universal vertex in the graph, the constructed structure is a universal clique decomposition. Observe that the constructed decompositions are unique (up to isomorphisms).

In the forward direction, we proceed by induction on the height of the universal clique decomposition. Suppose $(T, \mathcal{B})$ is a universal clique decomposition of a graph G. Consider the case when T has height 1, that is, we have only one single tree node (and one bag). Then this bag, by Proposition 2.2, is a clique (every vertex in the bag is universal), and since a complete graph is trivially perfect, the base case holds. Consider now the case when T has a height of at least 2. Let r be the root of T, and let $x_1, x_2, \dots, x_p$ be children of r in T. Observe that the tree T_{x_i} is a universal clique decomposition for the graph $G[\bigcup_{t \in V(T_{x_i})} B_t]$ for each $i = 1, 2, \dots, p$. Hence, by the induction hypothesis, we have that $G[\bigcup_{t \in V(T_{x_i})} B_t]$ is trivially perfect. To see that G is trivially perfect as well, observe that G can be obtained by taking the disjoint union of graphs $G[\bigcup_{t \in V(T_{x_i})} B_t]$ for $i = 1, 2, \dots, p$, and adding $|B_r|$ universal vertices. □

We define the following notion for use in describing the dynamic programming procedure.

Definition 2.5 (*Block*). Let $(T = (V_T, E_T), \mathcal{B} = \{B_t\}_{t \in V_T})$ be the universal clique decomposition of a connected trivially perfect graph $G = (V, E)$. For each node $t \in V_T$, we associate a *block* $L_t = (B_t, D_t)$, where

—B_t is the subset of V contained in the bag corresponding to t, and
—D_t is the set of vertices of V contained in the bags corresponding to the nodes of the subtree T_t.
—The *tail* of a block L_t is the set of vertices Q_t contained in the bags corresponding to the nodes of the path from t to the root r of T, including B_t and B_r.

When t is a leaf of T, we have that $B_t = D_t$, and we call the block $L_t = (B_t, D_t)$ a *leaf block*. If t is the root, we have that $D_t = V(G)$, and we call L_t the *root block*. Otherwise, we call L_t an *internal block*. Observe that for every block $L_t = (B_t, D_t)$ with tail Q_t, we have that $B_t \subseteq Q_t$, $B_t \subseteq D_t$, and $D_t \cap Q_t = B_t$ (see Figure 3). Note also that Q_t is a clique and the vertices of Q_t are universal to $D_t \setminus B_t$. The following lemma summarizes the properties of universal clique decompositions, maximal cliques, and blocks used in our proof.

LEMMA 2.6. *Let $(T, \mathcal{B})$ be the universal clique decomposition of a connected trivially perfect graph G and let $L = (B, D)$ be a block with Q as its tail.*

(*i*) *The following are equivalent:*
  (*1*) *L is a leaf block,*
  (*2*) *$D = B$,*
  (*3*) *Q is a maximal clique of G, and*
  (*4*) *$Q = N_G[v]$ for every $v \in B$.*
(*ii*) *If L is a nonleaf block, then for every two vertices u, v from different connected components of $G[D \setminus B]$, we have that $Q = N_G(u) \cap N_G(v)$.*
(*iii*) *For every maximal clique Q' of G, there exists a leaf block L' in $(T, \mathcal{B})$ with tail Q'.*

PROOF. *(i)* We first prove the chain (1) → (2) → (3) → (1). If L is a leaf block and D is the set of vertices in all the bags of the subtree rooted at L, then $B = D$. If $D = B$, we have that $N_G[v] = Q$ for any $v \in B$ (thus also (2) → (4)). Hence, Q is maximal. If L is not a leaf block, then there is a vertex v that belongs to a child node of L, and for any such vertex v the set $Q \cup \{v\}$ is a clique; thus Q is not a maximal clique. Hence, if Q is a maximal clique, then L is a leaf block.

Finally, we show that (4) → (2): Since Q is a clique and $v \in Q$, we have that $Q \subseteq N_G[v]$. On the other hand, since $v \in B$ and L is a leaf block, we have that $Q \supseteq N_G[v]$ by the definition of universal clique decomposition.

(ii) Suppose that $L = (B, D)$ is a nonleaf block and D_1 and D_2 are two connected components of $G' = G[D \setminus B]$. Let $v \in D_1$ and $u \in D_2$, and observe that since they are in

different connected components of $G[D \setminus B]$, $N_{G'}(v) \cap N_{G'}(u) = \emptyset$. By the universality of Q, the result follows: $Q = N_G(v) \cap N_G(u)$.

(iii) Let $(T, \mathcal{B})$ be a UCD and consider the following construction of a tail of a block L_t. Obviously, $\mathrm{uni}(G) \subseteq Q'$. Let $L_1 = (\mathrm{uni}(G), V(G))$. Consider the disconnected trivially perfect graph $G - \mathrm{uni}(G)$. This has exactly one component containing vertices from Q'. Let G_2 be the connected component of $G - \mathrm{uni}(G)$ containing a vertex from Q'. Again $\mathrm{uni}(G_2) \subseteq B'$, thus let $L_2 = (\mathrm{uni}(G_2), V(G_2))$. Continue until L_t is a leaf bag. Since $\mathrm{uni}(G_t) = V(G_t)$ and since $\mathrm{uni}(G_t) \subseteq Q'$ (again by maximality of Q'), we have that $Q' = \bigcup_{1 \leq 1 \leq t} \mathrm{uni}(G_i)$, which is exactly the tail of L_t. Since $(T, \mathcal{B})$ is unique, L_t is a leaf bag of $(T, \mathcal{B})$, which proves the claim. □

2.2. Structure of Minimal Completions

Before we proceed with the algorithm, we prove some properties of minimal completions. The following lemma gives insight to the structure of a yes instance.

LEMMA 2.7. *Let $G = (V, E)$ be a connected graph, S a minimal completion of G, and let $H = G + S$. Suppose $L = (B, D)$ is a block in the universal clique decomposition of H and let $D_1, D_2, \ldots, D_\ell$ be the connected components of $H[D] - B$.*

(*i*) *If L is not a leaf block, then $\ell > 1$;*
(*ii*) *If $\ell > 1$, then even in the original graph, G, every vertex $v \in B$ has at least one neighbor in each of the sets $D_1, D_2, \ldots, D_\ell$;*
(*iii*) *The graph $G[D_i]$ is connected for every $i \in \{1, \ldots, \ell\}$;*
(*iv*) *For every $i \in \{1, \ldots, \ell\}$, $B \subseteq N_G(D \setminus (B \cup D_i))$.*

PROOF. *(i)* Let (B, D) be a nonleaf block. Recall that, by the definition of a universal clique decomposition, B is the maximal universal clique of $H[D]$. From Lemma 2.6 (i), we get that $B \neq D$. Since $B = \mathrm{uni}(H[D])$, there must be two nonadjacent vertices in $D \setminus B$ and no universal vertex in $D \setminus B$. Since $H[D \setminus B]$ is trivially perfect, it must have several connected components, that is, $\ell > 1$.

(ii) Suppose, without loss of generality, that there exists a vertex $v \in B$ such that $N_G(v) \cap D_1 = \emptyset$. Let $S' = S \setminus (\{v\} \times V(D_1))$. Note that, since v is universal to $V(D_1)$ in H and is completely nonadjacent to $V(D_1)$ in G, it follows that $\{v\} \times V(D_1) \subseteq S$ and that S' is a proper subset of S. We claim that $H' = G + S'$ is also a trivially perfect graph, which contradicts the minimality of S. Indeed, consider a universal clique decomposition obtained from the universal clique decomposition of H by *(a)*, in the case $\ell = 2$, moving v from B to the root bag of D_2, or *(b)*, in the case $\ell > 2$, moving v from B to a new bag $B' = \{v\}$ attached below B, with all the root bags of $D_2, D_3, \ldots, D_\ell$ reattached from below B to below B'. It can easily be seen that this new universal clique decomposition is a universal clique decomposition of H', which proves that H' is trivially perfect.

(iii) For the sake of contradiction, suppose that $G[D_a]$ was disconnected. Let (D_{a_1}, D_{a_2}) be a partition of D_a such that there is no edge between D_{a_1} and D_{a_2} in G. Clearly, $H[D_{a_1}]$ and $H[D_{a_2}]$ are trivially perfect graphs as they are induced subgraphs of H, hence they admit some universal clique decompositions. Since $H[D_a]$ is connected, we infer that S contains some edges between D_{a_1} and D_{a_2}. Now, let $S' = S \setminus \{uv \mid u \in D_{a_1}, v \in D_{a_2}, uv \in S\}$; by the previous argument, we have that $S' \subsetneq S$. Modify now the given universal clique decomposition of H by removing the subtree below B that corresponds to D_a, and attaching instead two subtrees below B that are universal clique decompositions of $H[D_{a_1}]$ and $H[D_{a_2}]$. Observe that thus we obtain a universal clique decomposition of $G + S'$, which shows that $G + S'$ is trivially perfect. This contradicts the minimality of S.

(iv) It follows directly from *(i)* and *(ii)*: if $\ell > 0$, then $\ell > 1$ and every vertex of B has edges in G to all the different connected components of $D \setminus B$. $\square$

2.3. The Algorithm

As has been already mentioned, the following concept is crucial for our algorithm. Recall that, when Ω is a set of vertices in a graph G, by m_Ω we mean the number of edges in $G[\Omega]$.

Definition 2.8 (*Vital Potential Maximal Clique*). Let (G, k) be an input instance to TRIVIALLY PERFECT COMPLETION. A vertex set $\Omega \subseteq V(G)$ is a *trivially perfect potential maximal clique*, or simply *potential maximal clique*, if Ω is a maximal clique in some minimal trivially perfect completion of G. Moreover, if this trivially perfect completion contains at most k edges, then the potential maximal clique is called *vital*.

Observe that, given a yes instance (G, k) of TRIVIALLY PERFECT COMPLETION and a minimal completion S of size at most k, every maximal clique in $G + S$ is a vital potential maximal clique in G. Note also that, in particular, any vital potential maximal clique contains at most k nonedges. The following definition will be useful:

Definition 2.9 (*Fill Number*). Let $G = (V, E)$ be a graph, S a completion of G, and $H = G + S$. We define the *fill number* of a vertex v, denoted by $\mathrm{fn}_H^G(v)$, as the number of edges incident to v in S.

OBSERVATION 2.10. *If (G, k) is a yes instance of* TRIVIALLY PERFECT COMPLETION, *S a completion of G with $|S| \le k$, and $H = G + S$, then there are at most $2\sqrt{k}$ vertices v in G such that $\mathrm{fn}_H^G(v) > \sqrt{k}$.*

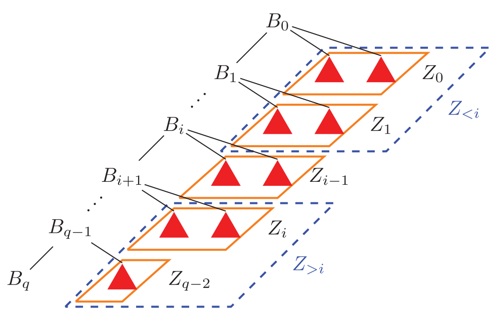
It follows that for such a graph $G = (V, E)$ and every set $U \subseteq V$ such that $|U| > 2\sqrt{k}$, there is a vertex $u \in U$ with $\mathrm{fn}_H^G(u) \le \sqrt{k}$. Any vertex u such that $\mathrm{fn}_H^G(u) \le \sqrt{k}$ will be referred to as a *cheap* vertex. If u is not cheap, we call it *expensive*.

We are now ready to begin the proof of Theorem 2.1. Our algorithm for TRIVIALLY PERFECT COMPLETION consists of three steps. We first compress, in polynomial time, the instance to an instance of size $\mathcal{O}(k^3)$. We then enumerate all the (subexponentially many) vital potential maximal cliques in this new instance. Finally, we do a dynamic programming procedure on these objects.

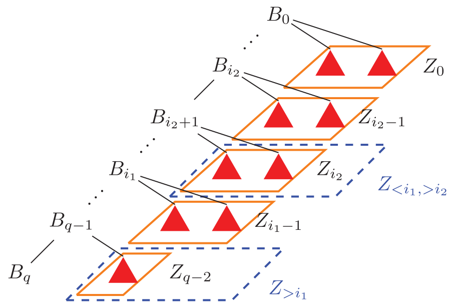
Step A. Kernelization. We start from one of the known polynomial kernelization algorithms for the problem. For a given input (G, k), we want to construct in polynomial time an equivalent instance (G', k'), where G' has $k^{\mathcal{O}(1)}$ vertices and $k' \le k$. Such a kernelization algorithm producing in time $\mathcal{O}(kn^4)$ an equivalent instance with graph G' on $\mathcal{O}(k^3)$ vertices was announced by Guo [2007]. The existence of a larger polynomial kernel with $\mathcal{O}(k^7)$ vertices also follows from the work on editing to trivially perfect graphs by Drange and Pilipczuk [2015, Theorem 3]. Let us note that, for our algorithm, any polynomial kernel will work fine. From now on, we assume that the input graph G has $\mathcal{O}(k^3)$ vertices. (Replacing $\mathcal{O}(k^3)$ by any other polynomial of k will not change our proof.) Without loss of generality, we will also assume that G is connected, since we can treat each connected component of G separately.

Step B. Enumeration. In this step, we describe an algorithm that in time $2^{\mathcal{O}(\sqrt{k}\log k)}$ outputs a family $\mathcal{C}$ of vertex subsets of G such that

—the size of $\mathcal{C}$ is $2^{\mathcal{O}(\sqrt{k}\log k)}$, and
—every vital potential maximal clique belongs to $\mathcal{C}$.



(a) Illustration of the vital potential maximal cliques of Type 3. Z_i is the collection of vertices corresponding to bags below B_i but not below B_{i+1}.



(b) Illustration of vital potential maximal cliques of Type 4 . Indices i_1, i_2 are the two largest indices such that Z_{i_1} and Z_{i_2} contain cheap vertices.

Fig. 4. Illustration of the different neighborhoods of the maximal clique that we use to find the maximal cliques of Types 3 and 4. A universal clique decomposition of a completed graph is shown, where $\Omega = B_0 \cup \ldots \cup B_q$ is a maximal clique. Observe that the leaf block is $L_q = (B_q, B_q)$ and that its tail is Ω.

We identify four different types of vital potential maximal cliques. For each type i, $1 \leq i \leq 4$, we list a family $\mathcal{C}_i$ of $2^{\mathcal{O}(\sqrt{k}\log k)}$ subsets containing all vital potential maximal cliques of that type. Finally, we set $\mathcal{C} = \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_4$. We show that every vital potential maximal clique of (G, k) is of at least one of these types and that all objects of each type can be enumerated in $2^{\mathcal{O}(\sqrt{k}\log k)}$ time.

Let Ω be a vital potential maximal clique of G. By the definition of Ω, there exists a minimal completion of G with at most k edges into a trivially perfect graph H such that Ω is a maximal clique in H. Let $(T = (V_T, E_T), \mathcal{B} = \{B_t\}_{t \in V_T})$ be the universal clique decomposition of H. Recall that, by Lemma 2.6, Ω corresponds to a path $P_{rt} = B_{t_0} B_{t_1} \cdots B_{t_q}$ in T from the root $r = t_0$ to a leaf $t = t_q$. Then, for the corresponding leaf block (B_t, D_t) with tail Q_t, we have that $\Omega = Q_t$. To simplify the notation, we use B_i for B_{t_i}.

Note that the algorithm knows neither the clique Ω nor the completed trivially perfect graph H. However, in the analysis, we may partition all the vital potential maximal cliques Ω with respect to structural properties of Ω and H, then provide simple enumeration rules ensuring that all vital potential maximal cliques of each type are indeed enumerated. We now proceed to the description of the types and enumeration rules and refer to Figure 4 for a visualization of these concepts. Henceforth, whenever we are referring to cheap or expensive vertices, we mean being cheap/expensive with respect to a fixed completion to H.

Type 1. Potential maximal cliques of the first type are such that $|V \setminus \Omega| \leq 2\sqrt{k}$. The family $\mathcal{C}_1$ consists of all sets $W \subseteq V$ such that $|V \setminus W| \leq 2\sqrt{k}$. There are at most $(2\sqrt{k}+1) \cdot \binom{\mathcal{O}(k^3)}{2\sqrt{k}}$ such sets, and using the fact that $\binom{a}{b} = a^{\mathcal{O}(b)} = 2^{\mathcal{O}(b \log a)}$, we enumerate all of them in time $2^{\mathcal{O}(\sqrt{k}\log k)}$ by trying all vertex subsets of size at least $|V| - 2\sqrt{k}$. Thus, every Type 1 vital potential maximal clique is in $\mathcal{C}_1$.

Type 2. By Lemma 2.6 *(i)*, we have that $\Omega = Q_t = N_H[v]$ for each vertex $v \in D_t = B_t$. Vital potential maximal cliques of the second type are those of the form $N_H[v]$ for some $v \in B_t$, where B_t is a leaf bag in the universal clique decomposition of H, and $|B_t| > 2\sqrt{k}$. We generate the family $\mathcal{C}_2$ as follows. Every set in $\mathcal{C}_2$ is of the form $W_1 \cup W_2$, where $W_1 = N_G[v]$ for some $v \in V$, and $|W_2| \leq \sqrt{k}$. There are at most $\binom{\mathcal{O}(k^3)}{\sqrt{k}} \cdot \mathcal{O}(k^3)$ such sets. They can be enumerated by computing for every vertex v the set $W_1 = N_G[v]$ and adding to each such set all possible subsets of size at most $\sqrt{k}$. Hence, every Type 2 vital potential maximal clique is in $\mathcal{C}_2$.

Thus, if Ω is not of Types 1 or 2, then $|V \setminus \Omega| > 2\sqrt{k}$; for the corresponding leaf block, we have $|B_t| \leq 2\sqrt{k}$. Since $|V \setminus \Omega| > 2\sqrt{k}$, it follows that $V \setminus \Omega$ contains at least one cheap vertex, that is, a vertex with fill number at most $\sqrt{k}$.

We partition the nodes of T that are not on the path $B_0, B_1, \ldots, B_q$ into q disjoint sets $Z_0, Z_1, \ldots, Z_{q-1}$ according to the nodes of the path P_{rt}. Node $x \notin V(P_{rt})$ belongs to Z_i, $i \in \{0, \ldots, q-1\}$, if i is the largest integer such that t_i is an ancestor of x in T. In other words, Z_i consists of bags of subtrees outside P_{rt} attached below t_i (see Figure 4(b)). For integers p_1, p_2, we shall denote $B_{p_1,p_2} = \bigcup_{j=p_1}^{p_2} B_j$.

For the remaining two types of vital potential maximal cliques, we distinguish cases depending on whether all cheap vertices in $V \setminus \Omega$ are located in exactly one set Z_i or not. Recall that all vital potential maximal cliques for which $V \setminus \Omega$ does not contain any cheap vertex are already contained in Type 1.

Type 3. Vital potential maximal cliques Ω of the third type are the ones that do not belong to Types 1 or 2, but for which there exists an index $i \in \{0, 1, \ldots, q-1\}$ such that all cheap vertices of $V \setminus \Omega$ belong to Z_i. Since Ω is not of Type 1, Z_i is nonempty. Also, since Ω is not of Type 2, we have that $|B_q| \leq 2\sqrt{k}$. Let us denote $Z_{<i} = \bigcup_{j=0}^{i-1} Z_j$ and $Z_{>i} = \bigcup_{j=i+1}^{q-1} Z_j$ (see Figure 4(a)). By our assumption, we have that $Z_{<i}$ and $Z_{>i}$ contain only expensive vertices, hence $|Z_{<i}|, |Z_{>i}| \leq 2\sqrt{k}$. Let u be any cheap vertex belonging to Z_i, and observe that the following equalities and inclusions are implied by Lemma 2.7 *(ii)*:

—$B_{0,i-1} = N_G(Z_{<i}) \subseteq \Omega$;
—$B_{i+1,q-1} \subseteq N_G(Z_{>i}) \subseteq \Omega$;
—$B_i \subseteq N_G(B_q \cup (N_G[Z_{>i}] \setminus N_H(u))) \subseteq \Omega$.

It follows that

$$\Omega = N_G(Z_{<i}) \cup N_G(Z_{>i}) \cup N_G\big(B_q \cup \big(N_G[Z_{>i}] \setminus N_H(u)\big)\big) \cup B_q. \tag{1}$$

Given (1), we may define family $\mathcal{C}_3$. Family $\mathcal{C}_3$ comprises all the sets that can be constructed as follows:

—Pick three disjoint sets $W_1, W_2, W_3 \subseteq V$ of size at most $2\sqrt{k}$ each. This corresponds to the choice of $Z_{<i}$, $Z_{>i}$, and B_q, respectively.
—Pick a vertex $v \in V$ and a set $A \subseteq V$ of size at most $\sqrt{k}$. This corresponds to the choice of u and fill-in edges adjacent to u. Let $N_v = N_G(v) \cup A$.
—Put the set $N_G(W_1) \cup N_G(W_2) \cup N_G(W_3 \cup (N_G[W_2] \setminus N_v)) \cup W_3$ into the family $\mathcal{C}_3$.

Observe that since $|V| = \mathcal{O}(k^3)$, and since we enumerate all possible collections of four sets of size $\mathcal{O}(\sqrt{k})$ and one vertex, the number of sets included in $\mathcal{C}_3$ is at most $\binom{\mathcal{O}(k^3)}{\mathcal{O}(\sqrt{k})}^4 \cdot \mathcal{O}(k^3) = 2^{\mathcal{O}(\sqrt{k}\log k)}$. Furthermore, this family can also be enumerated within the same asymptotic running time. From Equation (1), it follows immediately that each vital potential maximal clique of Type 3 is contained in $\mathcal{C}_3$.

Type 4. Vital potential maximal cliques Ω of the fourth type are the ones that do not belong to Types 1 or 2, but there exist at least two indices i_1 and i_2 such that Z_{i_1} and Z_{i_2} both contain a cheap vertex. Let i_1, i_2 be the two largest such indices, where $i_1 > i_2$. Let $Z_{<i_1,>i_2} = \bigcup_{j=i_2+1}^{i_1-1} Z_j$ and $Z_{>i_1} = \bigcup_{j=i_1+1}^{q-1} Z_j$ (see Figure 4(b)). By the maximality of i_1, i_2, we have that $Z_{<i_1,>i_2}$ and $Z_{>i_1}$ contain only expensive vertices, and hence that $|Z_{<i_1,>i_2}|, |Z_{>i_1}| \leq 2\sqrt{k}$. Again, since Ω is not of Type 2, we have that $|B_q| \leq 2\sqrt{k}$. Let

$u_1 \in Z_{i_1}$ and $u_2 \in Z_{i_2}$ be two cheap vertices. Observe that the following equalities and inclusions are implied by Lemma 2.7 *(ii)*:

—$B_{0,i_2} = N_H(u_1) \cap N_H(u_2)$;
—$B_{i_2+1,i_1-1} \subseteq N_G(Z_{<i_1,>i_2}) \subseteq \Omega$;
—$B_{i_1+1,q-1} \subseteq N_G(Z_{>i_1}) \subseteq \Omega$;
—$B_{i_1} \subseteq N_G(B_q \cup (N_G[Z_{>i_1}] \setminus N_H(u_1))) \subseteq \Omega$.

It follows that

$$\Omega = \big(N_H(u_1) \cap N_H(u_2)\big) \cup N_G(Z_{<i_1,>i_2}) \cup N_G(Z_{>i_1}) \\ \cup N_G\big(B_q \cup \big(N_G[Z_{>i_1}] \setminus N_H(u_1)\big)\big) \cup B_q \ . \tag{2}$$

Given Equation (2), we may define the family $\mathcal{C}_4$. This family comprises all the sets that can be constructed as follows:

—Pick three disjoint sets $W_1, W_2, W_3 \subseteq V$ of size at most $2\sqrt{k}$ each. This corresponds to the choice of $Z_{<i_1,>i_2}$, $Z_{>i_1}$, and B_q, respectively.
—Pick two vertices $v_1, v_2 \in V$ and two sets $A_1, A_2 \subseteq V$, each of size at most $\sqrt{k}$. This corresponds to the choice of u_1 and u_2, and of the neighbors in H adjacent to u_1 and u_2. Let $N_{v_i} = N_G(v_i) \cup A_i$, for $i = 1, 2$.
—Put the set $(N_{v_1} \cap N_{v_2}) \cup N_G(W_1) \cup N_G(W_2) \cup N_G(W_3 \cup (N_G[W_2] \setminus N_{v_1})) \cup W_3$ into the family $\mathcal{C}_4$.

Observe that, since $|V| = \mathcal{O}(k^3)$, and by the same analysis as for Type 3, the number of sets included in $\mathcal{C}_4$ is at most $2^{\mathcal{O}(\sqrt{k}\log k)}$, and that this family can be enumerated within the same asymptotic running time. From Equation (2), it follows immediately that each vital potential maximal clique of Type 4 is contained in $\mathcal{C}_4$.

Summarizing, every vital potential maximal clique of Types 1, 2, 3, and 4 is included in the family $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$, and $\mathcal{C}_4$, respectively. Since every vital potential maximal clique is of Types 1, 2, 3, or 4, by taking $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3 \cup \mathcal{C}_4$ we can infer the following lemma that formalizes the result of Step B.

LEMMA 2.11 (ENUMERATION LEMMA). *Let (G, k) be an instance of* TRIVIALLY PERFECT COMPLETION *such that $|V(G)| = \mathcal{O}(k^3)$. Then, in time $2^{\mathcal{O}(\sqrt{k}\log k)}$, we can construct a family $\mathcal{C}$ consisting of $2^{\mathcal{O}(\sqrt{k}\log k)}$ subsets of $V(G)$ such that every vital potential maximal clique of (G, k) is in $\mathcal{C}$.*

Step C. Dynamic programming. We first give an intuitive idea of the dynamic programming procedure. We start off by assuming that we have the family $\mathcal{C}$ containing all vital potential maximal cliques of (G, k). We start by generating in time $2^{\mathcal{O}(\sqrt{k}\log k)}$ a family $\mathcal{S}$ of pairs (X, Y), where $X, Y \subseteq V(G)$, such that for every minimal completion S of size at most k, and the corresponding universal clique decomposition $(T, \mathcal{B})$ of $H = G + S$, it holds that every block (B, D) is in $\mathcal{S}$, and the size of $\mathcal{S}$ is $2^{\mathcal{O}(\sqrt{k}\log k)}$. (See Definition 2.5 for the definition of a block.)

The construction of $\mathcal{S}$ is based on the following observations about blocks and vital potential maximal cliques: Let G be a graph, S a minimal completion, and $L = (B, D)$ a block of the universal clique decomposition of $H = G + S$, where H is not a complete graph, with Q being its tail. Then, the following hold, as we prove later:

—If L is a leaf block, then $B = \Omega_1 \setminus \Omega_2$ for some vital potential maximal cliques Ω_1 and Ω_2, and $D = B$.
—If L is the root block, then the tail of L is B, $B = \Omega_1 \cap \Omega_2$ for some vital potential maximal cliques Ω_1 and Ω_2, and $D = V$.

—If L is an internal block, then Q is the intersection of two vital potential maximal cliques Ω_1 and Ω_2 of G, $B = Q \setminus \Omega_3$ for some vital potential maximal clique Ω_3, and D is the connected component of $G - (Q \setminus B)$ containing B.

From this observation, we can conclude that by going through all triples $\Omega_1, \Omega_2, \Omega_3$ of elements of $\mathcal{C}$, we can compute the set $\mathcal{S}$ consisting of all blocks (B, D) of minimal completions. We now define the value $\text{dp}(B, D)$ as follows: $\text{dp}(B, D)$ is equal to the minimum number of edges needed to be added to $G[D]$ to make it a trivially perfect graph with B being the universal clique contained in the root of its universal clique decomposition, unless this minimum number is larger than k; in this case, we put $\text{dp}(B, D) = +\infty$. We later derive recurrence equations that enable us to compute all the relevant values of $\text{dp}(\cdot, \cdot)$ using dynamic programming. Finally, the minimum cost of completing G to a trivially perfect graph is equal to $\min_{(B,V(G))\in\mathcal{S}} \text{dp}(B, V(G))$. If this minimum is equal to $+\infty$, then no completion of size at most k exists and we can conclude that (G, k) is a no-instance.

We now proceed to a formal proof of the correctness of the dynamic programming procedure. Suppose that we have the family $\mathcal{C}$ containing all vital potential maximal cliques of (G, k). We start by generating in time $2^{\mathcal{O}(\sqrt{k}\log k)}$ a family $\mathcal{S}$ of pairs (X, Y), where $X, Y \subseteq V$, where $V = V(G)$, such that

—for every minimal completion H that adds at most k edges, every block (B, D) of the universal clique decomposition of H belongs to $\mathcal{S}$, and
—the size of $\mathcal{S}$ is $2^{\mathcal{O}(\sqrt{k}\log k)}$.

The construction of $\mathcal{S}$ is based on the following lemmata.

LEMMA 2.12. *Let G be a graph, S a minimal completion of G of size at most k, and (B, D) a nonleaf and nonroot block of the universal clique decomposition of $H = G + S$, with Q being its tail. Then*

(*i*) *Q is the intersection of two vital potential maximal cliques Ω_1 and Ω_2 of G,*
(*ii*) *$B = Q \setminus \Omega_3$ for some vital potential maximal clique Ω_3, and*
(*iii*) *D is the connected component of $G - (Q \setminus B)$ containing B.*

PROOF. *(i)* Consider two connected components D_1 and D_2 of $H[D \setminus B]$ and let Ω'_1 and Ω'_2 be maximal cliques in D_1 and D_2. Observe that $\Omega_1 = \Omega'_1 \cup Q$ and $\Omega_2 = \Omega'_2 \cup Q$ are maximal cliques in H. By definition, Ω_1 and Ω_2 are vital potential maximal cliques in G, and $\Omega_1 \cap \Omega_2 = Q$.

(ii) Let $\hat{L} = (\hat{B}, \hat{D})$ be the parent block of (B, D). Since $\hat{L}$ is not a leaf-block, $\hat{L}$ has at least two children; thus, there is a block $(B', D') \neq (B, D)$ that is also a child of $\hat{L}$. By the previous argument, $\hat{Q}$, the tail of $\hat{L}$ is exactly $\hat{Q} = \Omega_1 \cap \Omega_3$ for some vital potential maximal clique Ω_3. It follows that $B = Q \setminus \Omega_3$.

(iii) It follows from Lemma 2.7 that $G[D]$ is connected. It then follows immediately that D is the unique connected component of $G - (Q \setminus B)$ containing B. □

LEMMA 2.13. *Let G be a graph, S a minimal completion of G of size at most k, and $L = (B, D)$ a leaf block of the universal clique decomposition of $H = G + S$. If H is not a complete graph, then*

(*i*) *$B = \Omega_1 \setminus \Omega_2$ for some vital potential maximal cliques Ω_1 and Ω_2, and*
(*ii*) *$D = B$.*

PROOF. *(i)* Let $\hat{L} = (\hat{B}, \hat{D})$ be the parent block of L, which exists since L is not the root block. Let $L' = (B', D')$ be a child of $\hat{L}$, which is not L. If $L' = (B', D')$ is a leaf, then set $L'' = L'$, and if not, then let $L'' = (B'', D'')$ be a leaf that has L' as an ancestor. The blocks L' and L'' exist since $\hat{L}$ is not a leaf. Furthermore, let $\hat{Q}$ be the tail of $\hat{L}$, and let $\Omega_1 = N_H[B]$ and $\Omega_2 = N_H[B'']$; Ω_1 and Ω_2 are then two maximal cliques in H. From similar arguments as presented earlier, we get that $\hat{Q} = \Omega_1 \cap \Omega_2$, hence that $B = \Omega_1 \setminus \Omega_2$.

(ii) This follows immediately from Lemma 2.6. □

LEMMA 2.14. *Let G be a connected graph, S a minimal completion of G of size at most k, and $L = (B, D)$ the root block of the universal clique decomposition of $H = G + S$. If H is not a complete graph, then*

(*i*) *the tail of L is B,*
(*ii*) *$B = \Omega_1 \cap \Omega_2$ for some vital potential maximal cliques Ω_1 and Ω_2, and*
(*iii*) *$D = V$.*

PROOF. *(i)* By definition, the tail is the collection of vertices from B to the root. Since L is a root block, the tail is B itself.

(ii) This follows in the same manner as in the proof of Lemma 2.12 *(i)*, since B is the tail of block L.

(iii) From the definition of universal clique decompositions, we have that D is the connected component of $H[V \setminus (Q \setminus B)]$ containing B. But since $Q \setminus B = \emptyset$, we get that D is the connected component of H containing B. Now, since H is connected, the result follows. □

By making use of Lemmata 2.12–2.14, one can construct the required family $\mathcal{S}$ by going through all possible triples of elements of $\mathcal{C}$. The size of $\mathcal{S}$ is at most $|\mathcal{C}|^3 = 2^{\mathcal{O}(\sqrt{k}\log k)}$ and the running time of the construction of $\mathcal{S}$ is $2^{\mathcal{O}(\sqrt{k}\log k)}$. Note here that, by Lemma 2.7 *(iii)* and the fact that G is connected, we may discard from $\mathcal{S}$ every pair (B, D) where $G[D]$ is not connected.

For every pair $(X, Y) \in \mathcal{S}$, with $X \subseteq Y \subseteq V$, we define $\mathrm{dp}(X, Y)$ to be the minimum number of edges required to add to $G[Y]$ to obtain a trivially perfect graph where X is the maximal universal clique. If this minimum value exceeds k, we define $\mathrm{dp}(X, Y) = +\infty$. Thus, to compute an optimal solution, it is sufficient to go through the values $\mathrm{dp}(X, Y)$, where $(X, Y) \in \mathcal{S}$ with $Y = V$. In other words, to compute the size of a minimum completion, we find

$$\min_{(X,V)\in\mathcal{S}} \mathrm{dp}(X, V), \tag{3}$$

and if this value is $+\infty$, then the size of a minimum completion exceeds k.

In the following, for a subset of vertices A we write m_A to denote the number of edges inside A, that is, $m_A = |E(A)|$. We compute Equation (3) by making use of dynamic programming over elements of $\mathcal{S}$. For every pair $(X, Y) \in \mathcal{S}$ that can be a leaf block for some completion, that is, for all pairs with $X = Y$, we put

$$\mathrm{dp}(X, X) = \binom{|X|}{2} - m_X.$$

Of course, if the computed value exceeds k, then we put $\mathrm{dp}(X, X) = +\infty$.

For $(X, Y) \in \mathcal{S}$ with $X \subsetneq Y$, if (X, Y) is a block of some minimal completion H, then in H, we have that X is a maximal universal clique in $H[Y]$, every vertex of X is adjacent to all vertices of $Y \setminus X$, and the number of edges in $H[Y \setminus X]$ is the sum of the number of edges in the connected components of $H[Y \setminus X]$. By Lemma 2.7, the vertices of every connected component Y' of $H[Y \setminus X]$ induce a connected component in $G[Y \setminus X]$. Observe that each connected component Y' of $H[Y \setminus X]$ corresponds to a block (X', Y')

in the decomposition of H. Now, since $\mathcal{S}$ contains all blocks of minimal trivially perfect completions, it follows that $(X', Y') \in \mathcal{S}$.

Now, for $(X, Y) \in \mathcal{S}$, we use $m_{X,Y \setminus X} = |E(X, Y \setminus X)|$ to denote the number of edges between X and $Y \setminus X$ in G. Let C be the set of connected components of $G[Y \setminus X]$. Then we compute, in order of increasing size of Y,

$$\mathrm{dp}\,(X, Y) = \binom{|X|}{2} - m_X + |X| \cdot |Y \setminus X| - m_{X,Y \setminus X} + \sum_{G[Y'] \in C} \min_{(X',Y') \in \mathcal{S}} \mathrm{dp}(X', Y').$$

Again, if the value on the right-hand side exceeds k, then we set $\mathrm{dp}\,(X, Y) = +\infty$.

Since the cardinality of Y' is less than $|Y|$ and blocks are processed in increasing cardinality of Y, the value for $\mathrm{dp}\,(X', Y')$ has already been calculated when it is needed to compute $\mathrm{dp}\,(X, Y)$.

The running time required to compute $\mathrm{dp}\,(X, Y)$ is, up to a polynomial factor in k, proportional to the number of sets $(X', Y') \in \mathcal{S}$, which is $\mathcal{O}(|\mathcal{S}|)$. Thus, the total running time of the dynamic programming procedure is, up to a polynomial factor in k, proportional to $\mathcal{O}(|\mathcal{S}|^2)$; hence, Equation (3) can be computed in time $2^{\mathcal{O}(\sqrt{k}\log k)}$. This concludes Step C and the proof of Theorem 2.1.

3. COMPLETION TO THRESHOLD GRAPHS

In this section, we present an algorithm that solves THRESHOLD COMPLETION, which is $\mathcal{F}$-COMPLETION for the case when $\mathcal{F} = \{2K_2, C_4, P_4\}$, in subexponential parameterized time. More specifically, we show the following theorem:

THEOREM 3.1. THRESHOLD COMPLETION *is solvable in time* $2^{\mathcal{O}(\sqrt{k}\log k)} + \mathcal{O}(kn^4)$.

The proof of Theorem 3.1 is a combination of the following known techniques: the kernelization algorithm by Guo [2007]; the chromatic coding technique of Alon et al. [2009], also used in the subexponential algorithm of Ghosh et al. [2015] for split graphs; and the algorithm of Fomin and Villanger [2013] for chain completion. For the kernelization part, we use the following result from Guo [2007]. Guo stated and proved it for the complement problem THRESHOLD EDGE DELETION, but since the set of forbidden subgraphs $\mathcal{F} = \{2K_2, C_4, P_4\}$ is self-complementary, the deletion and completion problems are equivalent.

PROPOSITION 3.2 [GUO 2007, THEOREM 3]. THRESHOLD COMPLETION *admits a kernel with* $\mathcal{O}(k^3)$ *vertices. The running time of the kernelization algorithm is* $\mathcal{O}(kn^4)$.

Universal sets. We start by describing the *chromatic coding* technique by Alon et al. [2009]. Let f be a coloring (not necessarily proper) of the vertex set of a graph $G = (V, E)$ with t colors. We call an edge $e \in E$ *monochromatic* if its endpoints have the same color, and we call a set of edges $F \subseteq E$ *colorful* if no edge in F is monochromatic.

Definition 3.3. A *universal* (n, k, t)*-coloring family* is a family $\mathfrak{F}$ of functions from $[n]$ to $[t]$ such that for any graph G with vertex set $[n]$, and k edges, there is an $f \in \mathfrak{F}$ such that f is a proper coloring of G, that is, $E(G)$ is colorful.

PROPOSITION 3.4 [ALON ET AL. 2009]. *For any* $n > 10k^2$*, there exists an explicit universal* $(n, k, \mathcal{O}(\sqrt{k}))$*-coloring family* $\mathfrak{F}$ *of size* $|\mathfrak{F}| \leq 2^{\mathcal{O}(\sqrt{k}\log k)} \cdot \log n$.

Note that, by *explicit*, we mean here that the family $\mathfrak{F}$ not only exists, but can also be constructed in $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$ time.

3.1. Split, Threshold and Chain Graphs

Here,we give some known facts about split graphs, threshold graphs, and chain graphs that we will use to obtain the main result.

Definition 3.5. Given a graph $G = (V, E)$, a partition of its vertex set into sets C and I is called a *split partition* of G if C is a clique and I is an independent set.

We denote by (C, I) a split partition of a graph.

Definition 3.6 (*Split Graph*). A graph is a split graph if it admits a split partitioning.

PROPOSITION 3.7 (THEOREM 6.2, [GOLUMBIC 1980]). *A split graph on n vertices has at most $n + 1$ split partitions; these partitions can be enumerated in polynomial time.*

Definition 3.8. A *chain graph* is a bipartite graph $G = (A, B, E)$ where the neighborhoods of the vertices are nested, that is, there is an ordering of the vertices in A, $a_1, a_2, \ldots, a_{n_1}$, such that for each $i < n_1$, we have that $N(a_i) \subseteq N(a_{i+1})$, where $n_1 = |A|$.

We will use the following result, which is often used as an alternative definition of threshold graphs.

PROPOSITION 3.9 [MAHADEV AND PELED 1995, THEOREM 1.2.4]. *A graph G is a threshold graph if and only if G has a split partition (C, I) and the neighborhoods of the vertices of I are nested.*

Thus, the class of threshold graphs is a subclass of split graphs, and by Proposition 3.7, threshold graphs on n vertices have at most $n + 1$ split partitions.

3.2. The Algorithm

We now proceed to the details of the algorithm that solves THRESHOLD COMPLETION in the time stated in Theorem 3.1. Fomin and Villanger [2013] show that the following problem is solvable in subexponential time:

CHAIN COMPLETION	
Input:	A bipartite graph $G = (A, B, E)$ and integer k.
Parameter:	k
Question:	Is there a set of edges S of size at most k such that $(A, B, E \cup S)$ is a chain graph?

Note that in the CHAIN COMPLETION problem, the resulting chain graph must have the same bipartition as the input graph. Thus, despite the fact that chain graphs are exactly the $\{2K_2, C_3, C_4, P_4\}$-free graphs, formally CHAIN COMPLETION is not an $\mathcal{F}$-COMPLETION problem according to our definition.

PROPOSITION 3.10 [FOMIN AND VILLANGER 2013]. CHAIN COMPLETION *is solvable in $2^{\mathcal{O}(\sqrt{k}\log k)} + \mathcal{O}(k^2nm)$ time.*

We now have the results needed to give an algorithm for THRESHOLD COMPLETION, thus proving Theorem 3.1.

PROOF OF THEOREM 3.1. Before giving the full algorithm and proof of correctness, we start by giving an outline of the steps of the algorithm without proofs, the first of which is starting from Proposition 3.2, thereby obtaining a polynomial kernel with $\mathcal{O}(k^3)$ vertices in time $\mathcal{O}(kn^4)$. We will henceforth assume that the input graph G has $n = \mathcal{O}(k^3)$ vertices.

Suppose that (G, k) is a yes instance of Threshold Completion. Then there is an edge set S of size at most k such that $G + S$ is a threshold graph. Without loss of generality, we can assume that $n > 10k^2$. By Proposition 3.4, we can construct in $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)} = 2^{\mathcal{O}(\sqrt{k}\log k)}$ time an explicit universal $(n, k, \mathcal{O}(\sqrt{k}))$-coloring family $\mathfrak{F}$ of size $|\mathfrak{F}| = 2^{\mathcal{O}(\sqrt{k}\log k)} \cdot \log n = 2^{\mathcal{O}(\sqrt{k}\log k)}$. Since $|S| \leq k$, there is a vertex coloring $f \in \mathfrak{F}$ such that S is colorful.

We iterate through all the colorings $f \in \mathfrak{F}$. Let us examine one coloring $f \in \mathfrak{F}$, and let $V_1, V_2, \ldots, V_t$ be the partitioning of $V(G)$ according to f, where $t = \mathcal{O}(\sqrt{k})$. Then, since threshold graphs are hereditary and we assume S to be colorful, each V_i must induce a threshold graph—we cannot add edges within a color class. Denote by G_i^f the graph $G[V_i]$, where V_i is the ith partition according to f. We write G_i if f is clear from context.

By Proposition 3.7, $G+S$ has $\mathcal{O}(k^3)$ split partitions. Each such split partition of $G+S$ induces a split partition of G_i, $i \in \{1, \ldots, t\}$. Again, by Proposition 3.7, each G_i also has $\mathcal{O}(k^3)$ split partitions, which we can enumerate in polynomial time. We use brute force to generate the set of $\mathcal{O}((k^3)^t) = 2^{\mathcal{O}(\sqrt{k}\log k)}$ partitions of G, and the set of generated partitions contains all split partitions of $G + S$. This will be proved in Step C to follow. By Proposition 3.9, if (G, k) is a yes instance, then for at least one of the split partitions (C, I) of $G + S$, the neighborhoods of I are nested. To check if a split partition can be turned into a nested partition, we use Proposition 3.10.

We now describe the algorithm in detail:

Step A. Kernelization. Apply Proposition 3.2 to obtain in time $\mathcal{O}(kn^4)$ a kernel with $\mathcal{O}(k^3)$ vertices. From now on, we assume that the number of vertices n in G is $\mathcal{O}(k^3)$.

Step B. Generating universal families. If necessary, we add a set of isolated vertices to G to guarantee that $n > 10k^2$. We apply Proposition 3.4 to construct a universal $(n, k, \mathcal{O}(\sqrt{k}))$-coloring family $\mathfrak{F}$ of size $2^{\mathcal{O}(\sqrt{k}\log k)}$. For each generated coloring f and the corresponding vertex partition $V_1, V_2, \ldots, V_t$, $t = \mathcal{O}(\sqrt{k})$, we perform the following steps.

Step C. Generating split partitions. Let $f \in \mathfrak{F}$ be fixed. We generate all potential split partitions of G modulo f as follows, where $\mathcal{C}_i$ is the set of split partitions of G_i, the graph induced on color class i. Recall here that we may assume that G_i is a threshold graph, since S is assumed to be colorful, thus G_i has at most $n + 1 = O(k^3)$ split partitions. Construct $\mathcal{C}$ by taking every combination of split partitions for each G_i, hence a potential split partition $(C, I) \in \mathcal{C}$ has the property that for every $i \leq t$, $(C \cap V(G_i), I \cap V(G_i)) \in \mathcal{C}_i$. Since S is assumed to be colorful, and we collected the combination of every split partition of the color classes, $G + S$ has a split partition $(C, I) \in \mathcal{C}$.

The time required to generate all the elements of $\mathcal{C}$ is $\mathcal{O}((k^3)^t) = 2^{\mathcal{O}(\sqrt{k}\log k)}$. We also perform a sanity check by excluding from $\mathcal{C}$ all pairs (C, I), where I is not an independent set. It is important here to observe that there is a split partition such that when (G, k) is a yes instance, then there is a partitioning $\mathcal{C}_i$ for which there is a colorful solution S of size at most k. We perform the next step with each pair $(C, I) \in \mathcal{C}$.

Step D. Computing nested split partitions. For a pair $(C, I) \in \mathcal{C}$, such that I is an independent set in G, we first compute the number of edges c needed to turn C into a clique, that is, $c = \binom{|C|}{2} - m_C$. Finally, we use Proposition 3.10 to check if the neighborhood of I in C can be made nested by adding at most $k - c$ edges.

From the earlier discussions, if (G, k) is a yes instance of the problem, the solution will be found after completing the algorithm. Let S be such that $G + S$ is a threshold graph and $|S| \leq k$. Considering the graph $G - E + S$, that is, G but with edges only from S, we get that there is a proper coloring $f \in \mathcal{F}$ of G using $O(\sqrt{k})$ colors. Since we iterate through each $f \in \mathcal{F}$, in some iteration we will try with the correct one. Partition the vertices according to f. This partition has the property that S will not have an edge completely within one partition; hence, G_i must induce a split graph and has at most $|V(G_i)| + 1$ split partitions. Trying every single combination of split partitions of the different G_is gives every potential split partition of G. It then follows that running the chain completion on each of the potential split partitions will find any threshold completion of size at most k.

If no such completion is found, we conclude that (G, k) is a no instance. The running time to perform Step A is $\mathcal{O}(kn^4)$ and Step B is done in $2^{\mathcal{O}(\sqrt{k}\log k)}$ time. For every $f \in \mathfrak{F}$, in Step C we generate $2^{\mathcal{O}(\sqrt{k}\log k)}$ partitions. The total number of times that Step C is called is $|\mathfrak{F}|$, and the total number of partitions generated is $|\mathfrak{F}| \cdot 2^{\mathcal{O}(\sqrt{k}\log k)} = 2^{\mathcal{O}(\sqrt{k}\log k)}$. In Step D, we run the algorithm with running time $2^{\mathcal{O}(\sqrt{k}\log k)}$ on each of the $2^{\mathcal{O}(\sqrt{k}\log k)}$ partitions, resulting in a total running time of $2^{\mathcal{O}(\sqrt{k}\log k)} + \mathcal{O}(kn^4)$. □

4. COMPLETION TO PSEUDOSPLIT GRAPHS

In this section, we show that Pseudosplit Completion, or $\mathcal{F}$-Completion for $\mathcal{F} = \{2K_2, C_4\}$, can be solved by first applying a polynomial-time and parameter-preserving preprocessing routine, and then using the subexponential time algorithm of Ghosh et al. [2015] for Split Completion.

The crucial property of pseudosplit graphs that will be of use is the following characterization:

Proposition 4.1 [Maffray and Preissmann 1994]. *A graph $G = (V, E)$ is pseudosplit if and only if one of the following holds:*

—*G is a split graph, or*
—*V can be partitioned into a pseudosplit partition (C, I, X) such that $G[C \cup I]$ is a split graph with C being a clique and I being an independent set, $G[X] \cong C_5$; moreover, there is no edge between X and I and every edge is present between X and C.*

In other words, a pseudosplit graph is either a split graph or constructed from a split graph by adding a C_5 that is completely nonadjacent to the independent set of the split graph and completely adjacent to the clique set of the split graph. We call a graph that falls into the latter category a *proper pseudosplit graph*.

In order to ease the argumentation regarding minimal completions, we call a split partition (C, I) *I-maximal* if there is no vertex $v \in C$ such that $(C \setminus \{v\}, I \cup \{v\})$ is a split partition. Our algorithm uses the subexponential algorithm of Ghosh et al. [2015] for Split Completion as a subroutine. We therefore need the following result:

Proposition 4.2 [Ghosh et al. 2015]. Split Completion *is solvable in time $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$.*

Formally, in this section, we prove the following theorem:

Theorem 4.3. Pseudosplit Completion *is solvable in time $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$.*

The algorithm whose existence is asserted in Theorem 4.3 is given as Algorithm 1. We now proceed to prove that this algorithm is correct, and that its running time on input (G, k) is $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$. In the following, we adopt the notation from Algorithm 1.

As in the algorithm, we denote by X the set of five vertices that will be used as the set inducing a C_5 (we try all possible subsets; note that their number is bounded by $\mathcal{O}(n^5)$). Note here that since $G[X]$ admits a supergraph isomorphic to a C_5, it follows

ALGORITHM 1: Algorithm Solving PSEUDOSPLIT COMPLETION

```
(1) Use the algorithm from Proposition 4.2 to check in time $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$ if $(G, k)$ is a yes
    instance of SPLIT COMPLETION. If $(G, k)$ is a yes instance of SPLIT COMPLETION, then return
    that $(G, k)$ is a yes instance of PSEUDOSPLIT COMPLETION. Otherwise, we complete to a proper
    pseudosplit graph.
(2) For each $X = \{x_1, x_2, \ldots, x_5\} \subseteq V(G)$ such that there is a supergraph $G_X \supseteq G[X]$ and
    $G_X \cong C_5$, we construct an instance $(G', k')$ to SPLIT COMPLETION from $(G, k)$ as follows:
    (a) Let $k' = k + |E(G[X])| - 5$.
    (b) Add all possible edges between vertices of $X$, so that $X$ becomes a clique.
    (c) Add a set $A$ of $k + 2$ vertices to $G$.
    (d) Add every possible edge between $A$ and $N_G[X]$.
(3) Use Proposition 4.2 to check if $(G', k')$ is a yes instance of SPLIT COMPLETION. If $(G', k')$ is a
    yes instance of SPLIT COMPLETION, then return that $(G, k)$ is a yes instance of PSEUDOSPLIT
    COMPLETION.
(4) If for no set $X$ the answer yes was returned, then return no.
```

that $|E(G[X])| \leq 5$ and, consequently, $k' \leq k$. Similarly, by A we denote the set of $k + 2$ vertices we add that are adjacent only to $N_G[X]$. Intuitively, this set will be used to force that in any minimal split completion of size at most k it holds that $N_G[X] \subseteq C$. From now on, G' is the graph as in the algorithm, that is, G' is constructed from G by making X into a clique, adding vertices A and all possible edges between A and $N_G[X]$.

The following lemma will be crucial in the proof of the correctness of the algorithm.

LEMMA 4.4. *Assume that S is a minimal split completion of G' of size at most k', and let (C, I) be an I-maximal split partition of $G' + S$. Then:*

(i) $N_G[X] \subseteq C$,
(ii) $A \subseteq I$,
(iii) *no edge of S has an endpoint in A,*
(iv) *$C \setminus X$ is fully adjacent to X in $G' + S$, and*
(v) *$I \setminus A$ is fully nonadjacent to X in $G' + S$.*

PROOF. *(i)* Aiming towards a contradiction, suppose that some $v \in N_G[X]$ is in I. Since $A \subseteq N(v)$, we must then have that $A \subseteq C$. However, since A is independent in G', this demands adding at least $\binom{k+2}{2} > k'$ edges.

(ii) Aiming towards a contradiction, assume that $A \cap C \neq \emptyset$. Since $N_{G'}(A) \subseteq C$ and A is independent in G', it follows that $G' + S'$, where $S' = S \setminus (A \times A)$ is also a split graph with partition (C', I'), where $C' = C \setminus A$ and $I' = I \cup (A \cap C)$. Since $S' \subseteq S$ holds, we have that either $|S'| < |S|$, which contradicts the minimality of S, or that $S' = S$, which contradicts the assumption that partition (C, I) was I-maximal.

(iii) Suppose that there is an edge $e \in S$ incident to a vertex of A. Since $A \subseteq I$, we infer that $S \setminus \{e\}$ is still a split completion, which contradicts the minimality of S.

(iv) C is a clique in $G' + S$ and $X \subseteq C$, thus this holds trivially.

(v) Suppose for the sake of a contradiction that some $v^i \in I \setminus A$ is adjacent to some $v^x \in X$. Since $N_G[X] \subseteq C$, we have that $v^i v^x \in S$. But then $S \setminus \{v^i v^x\}$ is also a split completion, which contradicts the minimality of S. □

The correctness of the algorithm is implied by the following lemma:

LEMMA 4.5. *The instance (G, k) is a yes instance of* PSEUDOSPLIT COMPLETION *if and only if Algorithm 1 returns yes on input (G, k).*

PROOF. In the forward direction, let (G, k) be a yes instance for PSEUDOSPLIT COMPLETION. Observe that (G, k) is a yes instance for SPLIT COMPLETION if and only if our algorithm returns yes in the first test. We therefore assume that G has to be completed to a proper pseudosplit graph.

Let S_0 be a completion set with $|S_0| \leq k$ such that $G_0 = G + S_0$ is a proper pseudosplit graph. Let (C, I, X) be the pseudosplit partition of $G + S_0$; hence, $G_0[X]$ is isomorphic to a C_5. We claim that the algorithm will return yes when considering the set X in the second step; then, let G' be the graph constructed in the second step of the algorithm, starting with the set X. Let S be equal to S_0 with all the edges of $G_0[X]$ that were not present in $G[X]$ removed; note that $|S| = |S_0| + |E(G[X])| - 5 \leq k'$. We claim that $G' + S$ is a split graph with split partition $(C \cup X, I \cup A)$. Indeed, since $G'[X]$ is a clique, X is fully adjacent to C in $G_0 \subseteq G' + S$, and C is a clique in $G_0 \subseteq G' + S$, it follows that $C \cup X$ is a clique in $G' + S$. On the other hand, $I \cup A$ is independent in G' and all the edges of S have at least one endpoint belonging to $C \cup X$, thus $I \cup A$ remains independent in $G' + S$. As a result, $G' + S$ is a split graph, therefore, the algorithm will return yes after the application of Proposition 4.2 in the third step.

In the reverse direction, assume that Algorithm 1 returns yes on input (G, k). If it returned yes already after the first test, then G may be completed into a split graph by adding at most k edges; thus, in particular, (G, k) is a yes instance of Pseudosplit Completion. From now on, we assume that the algorithm returned yes in the third step. More precisely, for some set X, the application of Proposition 4.2 has found a minimal completion set S of size at most k' such that $G' + S$ is a split graph, with I-maximal split partition (C, I).

By Lemma 4.4, we have that *(i)* $N_G[X] \subseteq C$, *(ii)* $A \subseteq I$, *(iii)* no edge of S has an endpoint in A, *(iv)* $C \setminus X$ is fully adjacent to X in $G' + S$, and *(v)* $I \setminus A$ is fully nonadjacent to X in $G' + S$. By the choice of X, there exists a supergraph G_X of $G[X]$ such that $G_X \cong C_5$. Now, let S_0 be equal to S with all the edges of G_X that were not present in $G[X]$ included. Observe that $|S_0| \leq k$ and that by *(iii)* S_0 contains only edges incident to vertices of G. Consider now the partition $(C \setminus X, I \setminus A, X)$ of $V(G + S_0)$. Since (C, I) was a split partition of $G' + S$, it follows that $C \setminus X$ is a clique in $G + S_0$ and $I \setminus A$ is an independent set in $G + S_0$. Moreover, from *(iv)* and *(v)* it follows that X is fully adjacent to $C \setminus X$ in $G + S_0$ and fully nonadjacent to $I \setminus A$ in $G + S_0$. Finally, the graph induced by X in $G + S_0$ is $G_X \cong C_5$. By Proposition 4.1, we infer that $G + S_0$ is a pseudosplit graph; thus, the instance (G, k) is a yes instance of Pseudosplit Completion. □

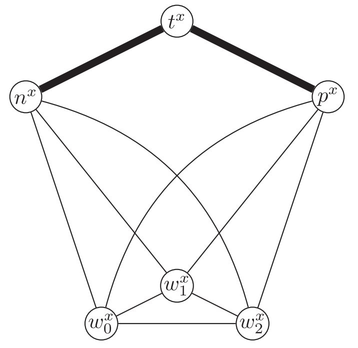
As for the time complexity of the algorithm, we try sets of five vertices for X, which is $\mathcal{O}(n^5)$ tries. For each such guess, we construct the graph G', which has $n+k+2$ vertices. Since $k' \leq k$, by Proposition 4.2 solving Split Completion requires time $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$, both in the first and the third steps of the algorithm. Therefore, the total running time of Algorithm 1 is $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$.

5. LOWER BOUNDS

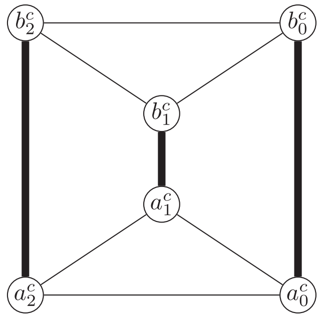
In this section, we will give the promised lower bounds described in Figure 2, that is, we will show that assuming ETH, $\mathcal{F}$-Completion is not solvable in subexponential time for $\mathcal{F}$ being one of $\{2K_2\}$, $\{C_4\}$, $\{P_4\}$, and $\{2K_2, P_4\}$.

Throughout this section, we will reduce to the problems presented earlier from 3Sat. We will assume that the input formula φ is in 3-CNF, that is, it is a conjunction of a number of clauses, each clause being a disjunction of at most three literals. By applying standard regularizing preprocessing for φ (e.g., see Fomin et al. [2014, Lemma 13]), we may also assume that each clause of φ contains *exactly* three literals, and that the variables appearing in these literals are pairwise different.

If φ is a 3Sat instance, we denote by $\mathcal{V}(\varphi)$ the variables in φ and by $\mathcal{C}(\varphi)$ the clauses. We assume that we have an ordering $c_1, c_2, \ldots, c_m$ for the clauses in $\mathcal{C}(\varphi)$ and the same



(a) The variable gadget G^x for a variable x with three occurrences of C_4. The edge $t^x p^x$ is the true edge and the edge $t^x n^x$ is the false edge. All C_4s of G^x can be eliminated by removing the true or the false edge.



(b) The clause gadget G^c for a clause c has three occurrences of C_4, which can be eliminated by removing two of the variable-edges, the thick edges in the figure.

Fig. 5. Variable (left) and clause (right) gadgets used in the reduction.

for the variables, $x_1, x_2, \ldots, x_n$. For simplicity, we also assume that the literals in each clause are ordered internally by the variable ordering.

We restate here the ETH, as that will be the crucial assumption for proving that the problems mentioned earlier do not admit subexponential time algorithms.

EXPONENTIAL TIME HYPOTHESIS *(ETH). There exists a positive real number s such that 3-CNF-SAT with n variables cannot be solved in time* 2^{sn}.

By the Sparsification Lemma of Impagliazzo et al. [2001], unless ETH fails, 3SAT cannot be solved in time $2^{o(n+m)}(n+m)^{\mathcal{O}(1)}$.

For each considered problem, we present a linear reduction from 3SAT, that is, a reduction that constructs an instance whose parameter is bounded linearly in the size of the input formula. Pipelining such a reduction with the assumed subexponential parameterized algorithm for the problem would give a subexponential algorithm for 3SAT, contradicting ETH.

5.1. $2K_2$-Free Completion Is Not Solvable in Subexponential Time

For $\mathcal{F} = \{2K_2\}$, we refer to $\mathcal{F}$-COMPLETION as to $2K_2$-FREE COMPLETION. We show the following theorem.

THEOREM 5.1. *The problem* $2K_2$-FREE COMPLETION *is NP-complete and not solvable in* $2^{o(k)}n^{\mathcal{O}(1)}$ *time unless the Exponential Time Hypothesis (ETH) fails.*

For the proof, however, instead of working directly on this problem, we find it more convenient to show the hardness of the (polynomially) equivalent problem C_4-FREE EDGE DELETION. We will throughout this section write $G - S$ for the graph $(V(G), E(G) \setminus S)$ when $S \subseteq E(G)$.

Construction. We reduce from 3SAT using the gadgets in Figure 5. Let φ be an instance of 3SAT. We construct the instance (G_φ, k_φ) for C_4-FREE EDGE DELETION and we begin by defining the graph G_φ. For every variable $x \in \mathcal{V}(\varphi)$, we construct a variable gadget graph G^x. The graph G^x consists of six vertices: w_0^x, w_1^x, w_2^x, n^x (for *negative*), p^x (for *positive*), and t^x. The three vertices w_0^x, w_1^x, and w_2^x will induce a triangle, whereas n^x and p^x are adjacent to the vertices in the triangle and to t^x. We can observe that the

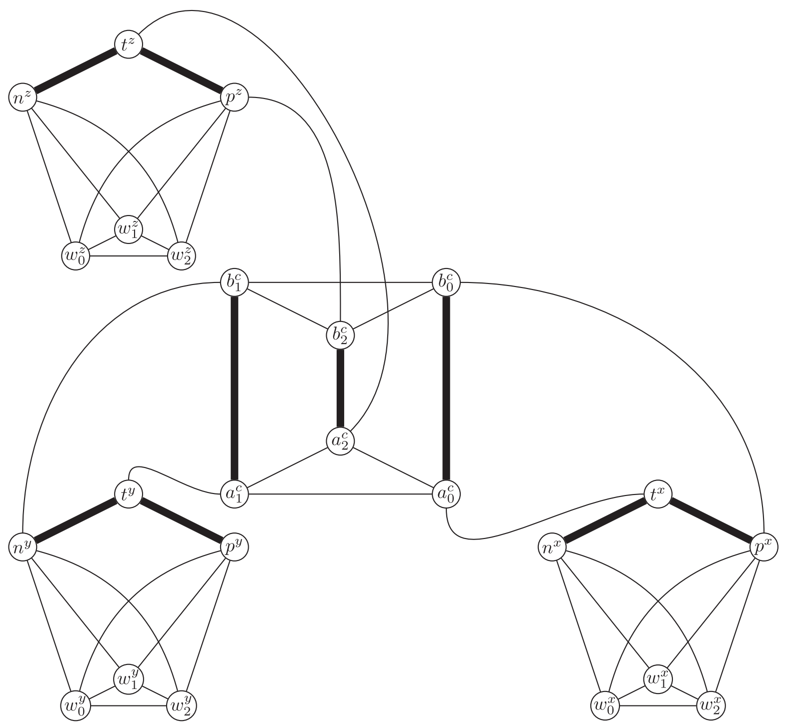


Fig. 6. The connections for a clause $c = x \vee \neg y \vee z$. For the negated variable, $\neg y$, we connect the clause gadget to n^y and t^y, whereas for the variables in the nonnegated form, we have the clause connected to the t and p vertices. Observe that a budget of five is sufficient and necessary for eliminating all occurrences of C_4 in the depicted subgraph.

four vertices n^x, t^x, p^x, and w_i^x induce a C_4 for $i = 0, 1, 2$, and that no other induced C_4 occurs in the gadget (see Figure 5(a)). It can also be observed that by removing either one of the edges $n^x t^x$ and $p^x t^x$, the gadget becomes C_4-free. We will refer to the edge $t^x p^x$ as the *true edge* and to $t^x n^x$ as the *false edge*. These edges are the thick edges in Figure 5(a). This concludes the variable gadget construction.

For every clause $c \in \mathcal{C}(\varphi)$, we construct a clause gadget graph G^c as follows. The graph G^c consists of two triangles, a_0^c, a_1^c, a_2^c and b_0^c, b_1^c, b_2^c. We also add the edges $a_0^c b_0^c$, $a_1^c b_1^c$, and $a_2^c b_2^c$. These three latter edges will correspond to the variables contained in c; we refer to them as *variable-edges* (the thick edges in Figure 5(b)). No more edges are added. The clause gadget can be seen in Figure 5(b). Observe that there are exactly three induced C_4s in G^c, all of the form $a_i^c, a_{i+1}^c, b_{i+1}^c, b_i^c$ for $i = 0, 1, 2$, where the indices behave cyclically are taken modulo 3. Moreover, removing any two edges of the form $a_i^c b_i^c$ for $i = 0, 1, 2$ eliminates all the induced C_4s contained in G^c.

To conclude the construction, we present the connections between variable gadgets and clause gadgets that encode literals in the clauses (see Figure 6). If a variable x appears in a nonnegated form as the ith (for $i = 0, 1, 2$) variable in a clause c, we add the edges $t^x a_i^c$ and $p^x b_i^c$. If it appears in a negated form, we add the edges $t^x a_i^c$ and $n^x b_i^c$. The connections can be seen in Figure 6. Observe that we get exactly one extra induced C_4 in the connection, and that this can be eliminated by removing either one of the thick edges.

This concludes the construction. We have now obtained a graph G_φ constructed from an instance φ of 3SAT. We let $k_\varphi = |\mathcal{V}(\varphi)| + 2|\mathcal{C}(\varphi)|$ be the allowed (and necessary) budget; the instance of C_4-FREE EDGE DELETION is then (G_φ, k_φ). This exposition, with the tightness of the budget in mind, can be summarized in two observations.

OBSERVATION 5.2. *In any solution S of an instance (G_φ, k_φ),*

—*for each variable gadget G^x, exactly one of $n^x t^x$ and $t^x p^x$ is contained in S, and*
—*for each clause gadget G^c, exactly two of the three edges $\{a_0^c b_0^c, a_1^c b_1^c, a_2^c b_2^c\}$ are in S.*

We now proceed to prove the following lemma, which will give the result.

LEMMA 5.3. *A given* 3SAT *instance φ has a satisfying assignment if and only if (G_φ, k_φ) is a yes instance of C_4-*FREE EDGE DELETION.

PROOF. Let φ be satisfiable and G_φ and k_φ be as before. We show that (G_φ, k_φ) is a yes instance for C_4-FREE EDGE DELETION. Let $\alpha : \mathcal{V}(\varphi) \to \{\texttt{true}, \texttt{false}\}$ be a satisfying assignment for φ. For every variable $x \in \mathcal{V}(\varphi)$, if $\alpha(x) = \texttt{true}$, we remove from the variable gadget G^x the edge corresponding to true, that is, the edge $t^x p^x$; otherwise, we remove the edge corresponding to false, that is, the edge $t^x n^x$. Every clause $c \in \mathcal{C}$ is satisfied by α; we pick an arbitrary variable x whose literal satisfies c and remove two edges corresponding to the two other literals. If a clause is satisfied by more than one literal, we pick any of the corresponding variables.

For every clause, we deleted exactly two edges; for every variable, we deleted exactly one edge. Thus, the total number of edges removed is $2|\mathcal{C}(\varphi)| + |\mathcal{V}(\varphi)| = k_\varphi$. We argue now that the remaining graph G'_φ is C_4-free. Since variables appearing in clauses are pairwise different, it can be easily observed that every induced cycle of length four in G_φ is either

—entirely contained in some clause gadget,
—entirely contained in some variable gadget, or
—is of form $t^x \gamma^x b_i^c a_i^c$, where x is the ith variable of clause c, and $\gamma \in \{n, p\}$ denotes whether the literal in c that corresponds to x is negated or nonnegated.

By the construction of G'_φ, we destroyed all induced 4-cycles of the first two types. Consider a 4-cycle $t^x p^x b_i^c a_i^c$ of the third type, where x appears positively in clause c. In the case when the literal of variable x was not chosen to satisfy c, we have deleted the edge $a_i^c b_i^c$, therefore this 4-cycle is removed. Otherwise, we have that $\alpha(x) = \texttt{true}$, and we have deleted the edge $t^x p^x$, thus also removing the considered 4-cycle. The case of a 4-cycle of the form $t^x n^x b_i^c a_i^c$ is symmetric.

We therefore get that all the induced 4-cycles that were contained in G_φ have been removed in G'_φ. Since vertex pairs (a_i^c, b_i^c) and (γ^x, t^x) for $\gamma \in \{n, p\}$ do not have common neighbors, it follows that no new C_4 could be created when obtaining G'_φ from G_φ by removing edges as described earlier. We infer that G'_φ is indeed C_4-free.

We now prove the opposite direction. Let S be an edge set of G_φ of size at most k_φ such that $G - S$ is C_4-free. By the definition of the budget k_φ and from Observation 5.2, that every variable gadget needs at least one edge to be in S and every clause gadget needs at least two edges to be in S (note here that the edge sets of clause and variable gadgets are pairwise disjoint), we have that S contains *exactly* one edge from each variable gadget, *exactly* two edges from each clause gadget, and no other edges.

We construct an assignment $\alpha : \mathcal{V}(\varphi) \to \{\texttt{true}, \texttt{false}\}$ for the formula φ as follows. For a variable $x \in \mathcal{V}(\varphi)$, we let $\alpha(x) = \texttt{true}$ if the true edge $t^x p^x$ of G^x is in S, and $\alpha(x) = \texttt{false}$ otherwise. By Observation 5.2, it follows that if $\alpha(x) = \texttt{false}$, then the false edge, $t^x n^x$ of G^x, is in S. We claim that the assignment α satisfies φ.

Suppose for the sake of contradiction that a clause $c \in \mathcal{C}$ is not satisfied. Since exactly two edges in the clause gadget G^c are in S, there is a variable x in c such that the corresponding variable edge of G^c is not in S. If $\alpha(x) = \texttt{true}$, then because c is not satisfied, we have that $\neg x \in c$. By the definition of α, we have that the false edge of G^x does not belong to S. Then, in G_φ, the false edge of G^x and the variable edge of G^c corresponding to x form part of an induced C_4 that is not destroyed by S, a contradiction. The case $\alpha(x) = \texttt{false}$ is symmetric. This concludes the proof of the lemma. □

Finally, the proof of Theorem 5.1 follows from Lemma 5.3: Combining the presented reduction with an algorithm solving C_4-Free Edge Deletion in $2^{o(k)}n^{\mathcal{O}(1)}$ time would yield an algorithm that solves 3Sat in $2^{o(n+m)} \cdot (n+m)^{\mathcal{O}(1)}$ time, which contradicts ETH by the results of Impagliazzo et al. [2001].

5.2. C_4-Free Completion Is Not Solvable in Subexponential Time

For every $\mathcal{F}$-Completion problem that so far turned out to be solvable in subexponential time, we had the graph C_4 in $\mathcal{F}$ together with some other graphs: trivially perfect graphs are the class excluding C_4 and P_4; threshold graphs are the class excluding $2K_2$, P_4 and C_4; and pseudosplit graphs are the class excluding $2K_2$ and C_4. Previous known subexponentiality results in the area of graph modifications are: completing to chordal graphs and chain graphs [Fomin and Villanger 2013]; completing to split graphs [Ghosh et al. 2015]; and, recently, completing to interval graphs [Bliznets et al. 2014a] and proper interval graphs [Bliznets et al. 2014b]. All these graph classes have C_4 as a forbidden induced subgraph.

It is therefore natural to ask whether the C_4 is the “reason” for the existence of subexponential algorithms. However, in this section, we show that excluding C_4 alone is not sufficient for achieving a subexponential time algorithm. For $\mathcal{F} = \{C_4\}$, we refer to $\mathcal{F}$-Completion as C_4-Free Completion.

Theorem 5.4. *The problem C_4-Free Completion is NP-complete and not solvable in $2^{o(k)}n^{\mathcal{O}(1)}$ time unless the Exponential Time Hypothesis (ETH) fails.*

To prove the theorem, we reduce from 3Sat, and similarly as before, we start with a formula in which each clause contains exactly three literals corresponding to pairwise different variables. By duplicating clauses if necessary, we also assume that each variable appears in at least two clauses.

We again need two types of gadgets: one gadget to emulate variables in the formula and one gadget to emulate clauses. Let φ be the 3Sat instance and denote by $\mathcal{V}(\varphi)$ the variables in φ and by $\mathcal{C}(\varphi)$ the clauses. We construct the graph G_φ as follows.

For each variable $x \in \mathcal{V}(\varphi)$, we construct a variable gadget graph G^x as depicted in Figure 7. Let p_x be the number of clauses in which x occurs; by our assumption, we have that $p_x \geq 2$. The graph G^x consists of a “tape” of $4p_x$ squares arranged in a cycle, with additional vertices attached to the sides of the tape. The intuition is that every fourth square in G^x is “reserved” for a clause in which x occurs. Formally, the vertex set of G^x is

$$V(G^x) = \bigcup_{0 \leq i < 4p_x} \{u_i^x, t_i^x, b_i^x, d_i^x\},$$

and its edge set is

$$E(G^x) = \bigcup_{0 \leq i < 4p_x} \{u_i^x t_i^x, u_i^x t_{i+1}^x, t_i^x u_{i+1}^x, t_i^x t_{i+1}^x,$$
$$t_i^x b_i^x, b_i^x b_{i+1}^x, b_i^x d_{i+1}^x, b_i^x d_i^x, d_i^x b_{i+1}^x\},$$

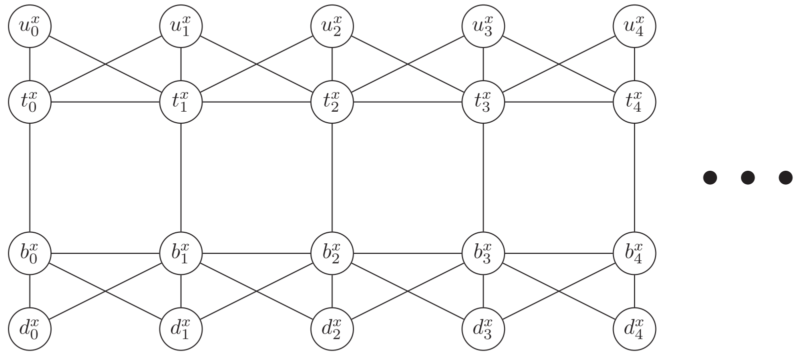
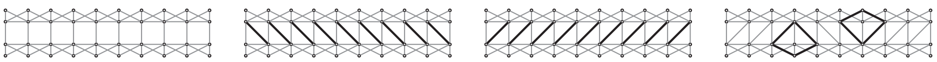


Fig. 7. Variable gadget G_x.



(a) Before completion (b) True (c) False (d) Differing orientations

Fig. 8. The variable gadget G_x before completion, its two completions corresponding to assignments, and a completion with differing orientations.

where the indices behave cyclically modulo $4p_x$. The letters for the vertices are chosen to correspond with top and bottom (t^x and b^x) of tape, and up and down (u^x and d^x). The construction is visualized in Figures 7 and 8(a).

Claim 5.5. *The minimum number of edges required to add to G^x to make it C_4-free is $4p_x$. Moreover, there are exactly two ways of eliminating all C_4s with $4p_x$ edges, namely, adding an edge on the diagonal for each square. Furthermore, if we add one edge to eliminate some cycle, the rest must have the same orientation, that is, either all added edges are of the form $t_i^x b_{i+1}^x$ or of the form $t_{i+1}^x b_i^x$. See Figure 8.*

Proof of Claim. A gadget G^x contains $4p_x$ induced C_4s, and no two of them can be eliminated by adding just one edge. Hence, to eliminate all C_4s in G^x, we need at least $4p_x$ edges. On the other hand, it is easy to verify that after adding $4p_x$ diagonals to C_4s of the same orientation, the resulting graph does not contain any induced C_4 (see Figure 8 for examples). Whenever we have two consecutive cycles with completion edges of different orientation, we create a new C_4 consisting of the two completion edges, and (depending on their orientation) either two edges incident to vertex u_i^x above their common vertex, or two edges incident to vertex d_i^x below. See Figure 8(d). □

Corollary 5.6. *The minimum number of edges required to eliminate all C_4s appearing inside all the variable gadgets is $12|\mathcal{C}(\varphi)|$.*

Proof. Since each clause of $\mathcal{C}(\varphi)$ contains exactly three occurrences of variables, it follows that $\sum_{x\in\mathcal{V}(\varphi)} p_x = 3|\mathcal{C}(\varphi)|$. The constructed variable gadgets are pairwise disjoint; thus, by Claim 5.5, we infer that the minimum number of edges required in all the variable gadgets is equal to $\sum_{x\in\mathcal{V}(\varphi)} 4p_x = 3\cdot 4|\mathcal{C}(\varphi)| = 12|\mathcal{C}(\varphi)|$. □

We now proceed to create the clause gadgets. For each clause $c \in \mathcal{C}(\varphi)$, we create the graph G^c as depicted in Figure 9. It consists of an induced 4-cycle $v_1^c v_4^c v_2^c v_3^c$ and induced paths $v_2^c u_1^c u_2^c v_1^c$ and $v_3^c u_4^c u_3^c v_4^c$. We also attach a gadget consisting of k_φ internally disjoint-induced paths of four vertices with endpoints in v_4^c and u_4^c, where k_φ is the budget to

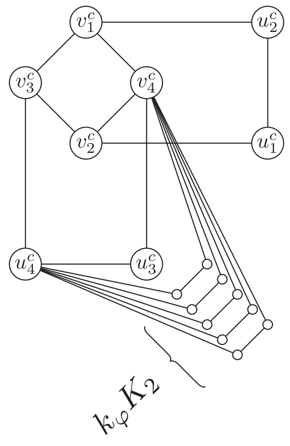


Fig. 9. The clause gadget G^c. It contains one C_4, and if we add either edge $v_1^c v_2^c$ or edge $v_3^c v_4^c$, we get a new C_4 that must be destroyed by adding one more edge. The $k_\varphi K_2$ gadget makes sure that we cannot add edge $v_4^c u_4^c$.

be specified later. This makes it impossible to add an edge between v_4^c and u_4^c in any C_4-free completion with at most k_φ edges.

By the i-th *square*, we mean a quadruple $(t_i^x, b_i^x, t_{i+1}^x, b_{i+1}^x)$. If a clause c is the ℓ-th clause the variable x appears in, we will use the vertices of the $4(\ell-1)$st square for connections to the gadget corresponding to c. For ease of notation, let $j = 4(\ell-1)$. We also use pairs $\{v_1^c, u_1^c\}$, $\{v_2^c, u_2^c\}$, and $\{v_3^c, u_3^c\}$ of G^c for connecting to the corresponding variable gadgets. If a variable x appears in a nonnegated form as the ith (for $i = 1, 2, 3$) literal of a clause c, then we add the edges $t_{j+1}^x v_i^c$ and $b_j^x u_i^c$. If it appears in a negated form, we add the edges $t_j^x v_i^c$ and $b_{j+1}^x u_i^c$ (see Figure 10). This concludes the construction of G_φ. Finally, we set the budget for the instance to be $k_\varphi = 14|\mathcal{C}(\varphi)|$.

Claim 5.7. *For each clause gadget G^c for a clause $c \in \mathcal{C}(\varphi)$, we need to add at least two edges between vertices of G^c to eliminate all induced C_4s in G^c. Moreover, there are exactly three ways of adding exactly two edges to G^c so that the resulting graph does not contain any induced C_4: by adding $\{v_1^c v_2^c, v_1^c u_1^c\}$, $\{v_1^c v_2^c, v_2^c u_2^c\}$, or $\{v_3^c v_4^c, v_3^c u_3^c\}$.*

Proof of Claim. There is a four-cycle $v_1^c v_4^c v_2^c v_3^c$ that needs to be eliminated, either by adding the edge $v_1^c v_2^c$ (Figure 11(b)) or $v_3^c v_4^c$ (Figure 11(c)). In any case, we create a new C_4, $v_1^c u_2^c u_1^c v_2^c$ in the former case and $v_4^c u_3^c u_4^c v_3^c$ in the latter case. In the former case, we can eliminate the created C_4 by adding $v_1^c u_1^c$ or $v_2^c u_2^c$; in the latter case, we can eliminate it by adding $v_3^c u_3^c$. Note that, in the latter case, we cannot add $v_4^c u_4^c$, since then we would create k_φ new induced four-cycles. A direct check shows that all three aforementioned completion sets lead to a C_4-free graph. □

Lemma 5.8. *Given a* 3Sat *instance φ, we have that (G_φ, k_φ) is a yes instance for C_4-*Free Completion *for $k_\varphi = 14|\mathcal{C}(\varphi)|$ if and only if φ is satisfiable.*

Proof. In the backwards direction, suppose φ is satisfiable. Let $\alpha : \mathcal{V}(\varphi) \to \{\texttt{true}, \texttt{false}\}$ be a satisfying assignment for φ. For every variable $x \in \mathcal{V}(\varphi)$, if $\alpha(x) = \texttt{true}$, we add the edges $t_i^x b_{i+1}^x$ to the completion set S for $i \in \{0, \ldots, 4p_x - 1\}$ and if $\alpha(x) = \texttt{false}$, we add the edges $t_{i+1}^x b_i^x$ to S for $i \in \{0, \ldots, 4p_x - 1\}$.

For a clause c in $\mathcal{C}(\varphi)$, if the first literal satisfies the clause, we add the edges $v_1^c v_2^c$ and $v_1^c u_1^c$ to S. If the second literal satisfies the clause, we add $v_1^c v_2^c$ and $v_2^c u_2^c$ to S. If it is the third literal, we add $v_3^c v_4^c$ and $v_3^c u_3^c$ to S. If more than one literal satisfies the clause,

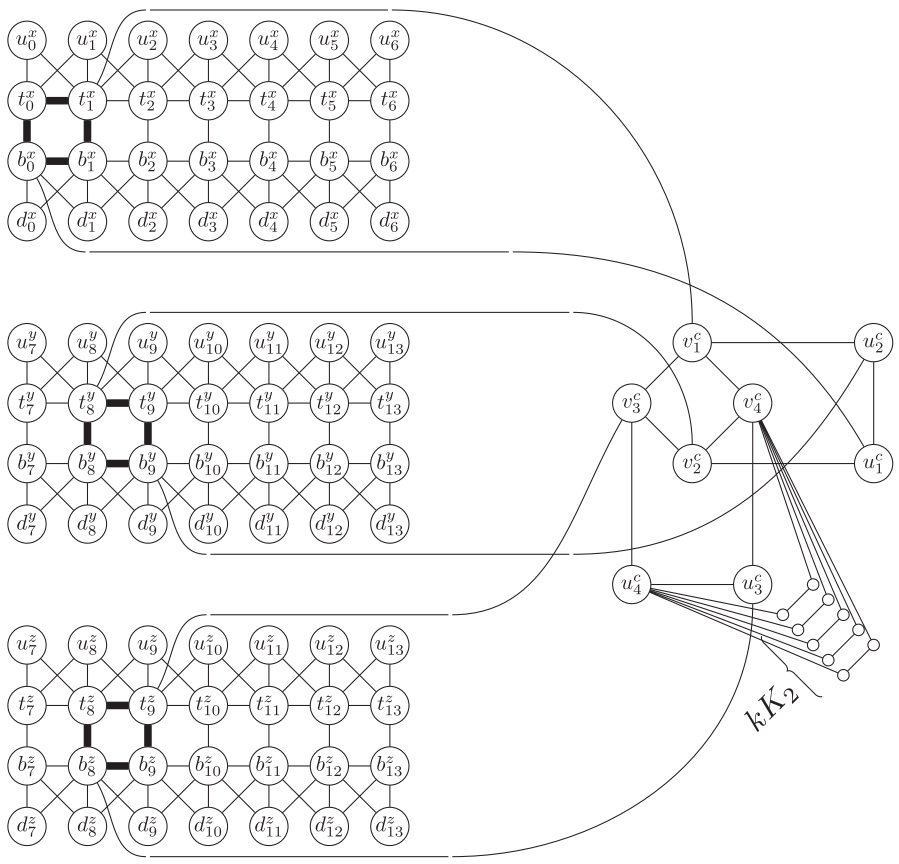
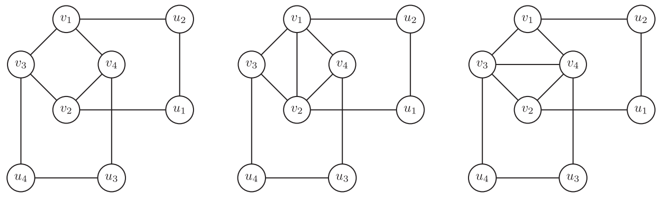

u_0^x u_1^x u_2^x u_3^x u_4^x u_5^x u_6^x
t_0^x t_1^x t_2^x t_3^x t_4^x t_5^x t_6^x
b_0^x b_1^x b_2^x b_3^x b_4^x b_5^x b_6^x
d_0^x d_1^x d_2^x d_3^x d_4^x d_5^x d_6^x
u_7^y u_8^y u_9^y u_{10}^y u_{11}^y u_{12}^y u_{13}^y
t_7^y t_8^y t_9^y t_{10}^y t_{11}^y t_{12}^y t_{13}^y
b_7^y b_8^y b_9^y b_{10}^y b_{11}^y b_{12}^y b_{13}^y
d_7^y d_8^y d_9^y d_{10}^y d_{11}^y d_{12}^y d_{13}^y
u_7^z u_8^z u_9^z u_{10}^z u_{11}^z u_{12}^z u_{13}^z
t_7^z t_8^z t_9^z t_{10}^z t_{11}^z t_{12}^z t_{13}^z
b_7^z b_8^z b_9^z b_{10}^z b_{11}^z b_{12}^z b_{13}^z
d_7^z d_8^z d_9^z d_{10}^z d_{11}^z d_{12}^z d_{13}^z
v_1^c u_2^c v_3^c v_4^c v_2^c u_1^c u_4^c u_3^c
kK_2

Fig. 10. The connections for a clause $c = x \vee \neg y \vee z$. In this example, c is the first clause of appearance for x, thus x is connected to G^c via the 0th square. For y and z, we assume that c is the third clause in which they appear, thus y and z use the 8th square.


v_1 u_2 v_3 v_4 v_2 u_1 u_4 u_3
v_1 u_2 v_3 v_4 v_2 u_1 u_4 u_3
v_1 u_2 v_3 v_4 v_2 u_1 u_4 u_3

(a) Before completion

(b) Variable 1 or 2 in the clause must be satisfied

(c) Variable 3 in the clause must be satisfied

Fig. 11. The clause gadget.

we pick any one. In total, this makes $12|\mathcal{C}(\varphi)|$ edges added to the variable gadgets and $2|\mathcal{C}(\varphi)|$ edges added to the clause gadgets.

Suppose now, for a contradiction, that $G_\varphi + S$ contains a cycle L of length four. From Claims 5.5 and 5.7, we know that L is not completely contained in a variable or clause gadget. Each vertex has at most one incident edge ending outside the gadget of the vertex and such edges are there only between variable and clause gadgets. Thus L consists of one edge from a variable gadget and one from a clause gadget and two edges between. We can observe that L then must contain either $v_1^c u_1^c$, $v_2^c u_2^c$ or $v_3^c u_3^c$ of the clause gadget (see Figure 10). Let us assume without loss of generality that L contains the edge $v_1^c u_1^c$. By the construction of the set S, this implies that the literal of the first variable x of c satisfies c. If x is nonnegated in c, then we have that $\alpha(x) = \texttt{true}$ and that $v_1^c t_{j+1}^x$ and $u_1^c b_j^x$ are edges of L. To complete the cycle, $t_{j+1}^x b_j^x$ must be an edge of L; however, by the definition of S, we have added the edge $t_j^x b_{j+1}^x$ to S instead of $t_{j+1}^x b_j^x$, and we obtain a contradiction. The case in which x is negated is symmetric.

In the forward direction, suppose (G_φ, k_φ) is a yes instance for $k_\varphi = 14|\mathcal{C}(\varphi)|$ and let S be such that $G_\varphi + S$ is C_4-free with $|S| \leq k_\varphi$. By Corollary 5.6 and Claim 5.7, we know that we need to use at least $12|\mathcal{C}(\varphi)|$ edges to fix the variable gadgets and we need to use at least $2|\mathcal{C}(\varphi)|$ edges for the clause gadgets. Since $|S| \leq k_\varphi$, we infer that $|S| = k_\varphi$, that we use exactly $4p_x$ edges to fix each variable gadget G^x (and that the orientation of the added edges must be the same within each gadget), that we use exactly two edges for each clause gadget G^c, and that S contains no edges other than those mentioned earlier.

We now define an assignment α for $\mathcal{V}(\varphi)$ and prove that it is indeed a satisfying assignment. If S contains the edge $t_0^x b_1^x$, we let $\alpha(x) = \texttt{true}$, and if S contains the edge $t_1^x b_0^x$, we let $\alpha(x) = \texttt{false}$. Let $c \in \mathcal{C}(\varphi)$ be a clause and suppose that c is not satisfied by the assignment α. We know by Claim 5.7 that the gadget for c contains $\{v_1^c v_2^c, v_1^c u_1^c\}$, $\{v_1^c v_2^c, v_2^c u_2^c\}$, or $\{v_3^c v_4^c, v_3^c u_3^c\}$.

Without loss of generality, assume that G^c contains $\{v_1^c v_2^c, v_1^c u_1^c\}$ and that x is the first variable in c, and that it appears nonnegated. Since x does not satisfy c, we infer that $\alpha(x) = \texttt{false}$. This means that $t_1^x b_0^x \in S$, and since the orientation of the added edges in the gadget G^x is the same, then also $t_{i+1}^x b_i^x \in S$. As a result, both edges $t_{i+1}^x b_i^x$ and $v_1^c u_1^c$ are present in $G_\varphi + S$. But then we have an induced four-cycle $v_1^c u_1^c b_i^x t_{i+1}^x v_1^c$, contradicting the assumption that $G_\varphi + S$ was C_4-free. The cases for y, z and negative literals are symmetric. This concludes the proof. □

Similarly, as before, the proof of Theorem 5.4 can be completed as follows: combining the presented reduction with an algorithm that solves C_4-Free Completion in $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ time would give an algorithm that solves 3Sat in $2^{o(n+m)} \cdot (n+m)^{\mathcal{O}(1)}$ time, which contradicts ETH by the results of Impagliazzo et al. [2001].

5.3. *P*$_4$-Free Completion Is Not Solvable in Subexponential Time

In this section, we show that there is no subexponential algorithm for $\mathcal{F}$-Completion for $\mathcal{F} = \{P_4\}$ unless the ETH fails. Let us recall that since $\overline{P_4} = P_4$, the problems P_4-Free Edge Deletion and P_4-Free Completion are polynomial time equivalent, and that this graph class more commonly goes under the name *cographs*. In other words, we aim to convince the reader of the following.

Theorem 5.9. *The problem P_4-Free Completion is NP-complete and not solvable in $2^{o(k)} n^{\mathcal{O}(1)}$ time unless the Exponential Time Hypothesis (ETH) fails.*

We reduce from 3Sat to the complement problem P_4-Free Edge Deletion. Let φ be the input 3Sat formula, for which we again assume that every clause of φ contains exactly three literals corresponding to pairwise different variables. For a variable $x \in \mathcal{V}(\varphi)$, we denote by p_x the number of clauses in φ containing x. Note that, since each clause contains exactly three variables, we have that $\sum_{x \in \mathcal{V}(\varphi)} p_x = 3|\mathcal{C}(\varphi)|$. We construct a graph G_φ such that for $k_\varphi = 4|\mathcal{C}(\varphi)| + \sum_{x \in \mathcal{V}(\varphi)} 4p_x = 16|\mathcal{C}(\varphi)|$, φ is satisfiable if and only if (G_φ, k_φ) is a yes instance of P_4-Free Edge Deletion. Since the complement of P_4 is P_4, this will prove the theorem.

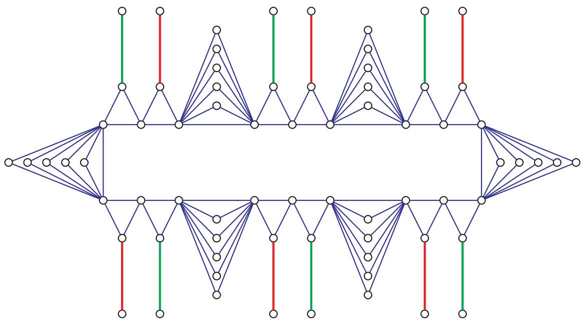


Fig. 12. Variable gadget G_x for a variable appearing in six clauses in φ, that is, $p_x = 6$. Deleting the leftmost edge in each tower pair corresponds to setting x to false; deleting the rightmost edge in each tower pair corresponds to setting x to true.

Variable gadget. For each variable $x \in \mathcal{V}(\varphi)$, we create a gadget G^x that looks like the one given in Figure 12. Before defining the construction formally, let us first describe it informally. We call a triangle with a pendant vertex a *tower*: the triangle will be referred to as the *base* of the tower, and the pendant vertex the *spike* of the tower. In the construction, the towers will always come in pairs, and they are joined in one of the vertices in the bases (two vertices are identified; see Figure 12). Pairs of towers will be separated by k' (defined later) triangles sharing an edge. The vertices not shared between the k' triangles will be called the *stack*, whereas the edge shared among the triangles will be called the *shortcut*.

The gadget G^x for a variable x consists of p_x pairs of towers arranged in a cycle, one for each clause in which x appears; pairs of consecutive towers are separated by a shortcut edge and a stack of vertices. The stack is chosen to be big enough ($k' = k_\varphi + 3$ vertices) so that we will never delete the edge that connects the two towers on each side of the stack, nor any edge incident to a vertex from the stack. We will refer to the two towers in the pairs as Tower 1 (the one with lower index) and Tower 2.

Formally, let φ be an instance of 3SAT. The budget for the output instance will be $k_\varphi = 4|\mathcal{C}(\varphi)| + \sum_{x \in \mathcal{V}(\varphi)} 4p_x = 16|\mathcal{C}(\varphi)|$. Let $k' = k_\varphi + 3$. For a variable x that appears in p_x clauses, we create vertices $s^x_{i,j}$ for $i \in \{1, \ldots, p_x\}$ and $j \in \{1, \ldots, k'\}$. These will be the vertices for the stacks. For the spikes of the towers, we add vertices $t^x_{i,1}$ and $t^x_{i,2}$ for $i \in \{1, \ldots, p_x\}$. For the bases of the towers, we add vertices $b^x_{i,j}$ for $j \in \{1, \ldots, 5\}$ and $i \in \{1, \ldots, p_x\}$. These are all the vertices of the gadget G^x for $x \in \mathcal{V}(\varphi)$.

The vertices denoted by t are the two spikes in the tower, that is, $t^x_{i,1}$ is the spike of Tower 1 of the ith pair for variable x. The vertices denoted by b are for the bases (there are five vertices in the bases of a pair of towers).

Now, we add edges to G^x (see Figure 13):

—For the stack, we add edges $s^x_{i,j} b^x_{i,1}$ for all $i \in \{1, \ldots, p_x\}$ and $j \in \{1, \ldots, k'\}$ (right side of the stack) and edges $b^x_{i,5} s^x_{i+1,j}$ for $i \in \{1, \ldots, p_x\}$ and $j \in \{1, \ldots, k'\}$ (left side of the next stack), where the indices behave cyclically modulo p_x.

—For the bases, we add the edges $b^x_{i,1} b^x_{i,2}$, $b^x_{i,1} b^x_{i,3}$, $b^x_{i,2} b^x_{i,3}$, $b^x_{i,3} b^x_{i,4}$, $b^x_{i,3} b^x_{i,5}$, and $b^x_{i,4} b^x_{i,5}$ for $i \in \{1, \ldots, p_x\}$. To attach the towers, we add the edges $b^x_{i,2} t^x_{i,1}$ and $b^x_{i,4} t^x_{i,2}$. The set of these eight edges will be denoted by R^x_i.

—The last edges to add are the shortcut edges $b^x_{i,5} b^x_{i+1,1}$ for $i \in \{1, \ldots, p_x\}$, where again the indices behave cyclically modulo p_x.

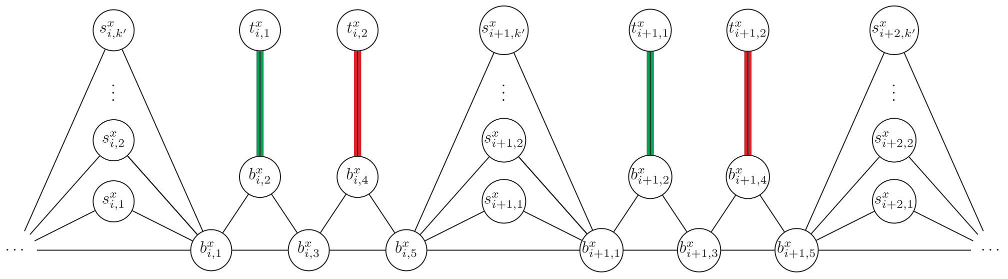


Fig. 13. Variable gadget G^x. The counter i ranges from 1 to p_x, the number of clauses x appears in. This figure does not illustrate that the gadget is a cycle; see Figure 12 for a zoomed-out version.

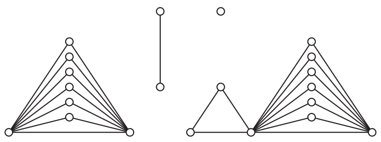
Elimination from variable gadgets. We will now show that there are exactly two ways of eliminating all P_4s occurring in a variable gadget using at most $4p_x$ edges. To state this claim formally, we need to control how the variable gadget is situated in a larger construction of the whole output instance that will be defined later. We say that a variable gadget G^x is *properly embedded* in the output instance G_φ if G^x is an induced subgraph of G_φ. Moreover, the only vertices of G^x that are incident to edges outside G^x are the spikes of the towers, that is, vertices $t^x_{i,1}$ and $t^x_{i,2}$ for $i \in \{1, 2, \ldots, p_x\}$. This property will be satisfied for gadgets G_x for all $x \in \mathcal{V}(\varphi)$ in the next steps of the construction. Using this notion, we can infer properties of the variable gadget irrespective of what the whole output instance G_φ constructed later looks like.

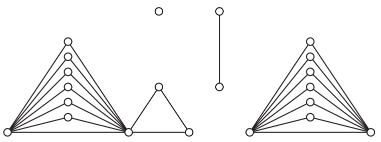
We first show that an inclusion minimal deletion set S that has size at most k_φ cannot touch the stacks or the shortcut edges.

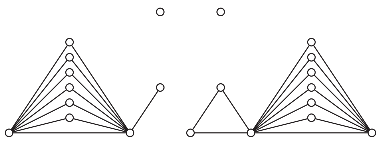
Claim 5.10. *Assume that gadget G^x is embedded properly in the output graph G_φ, and that S is an inclusion minimal P_4-free edge deletion set in G_φ of size at most k_φ. Then S does not contain any edge of type $b^x_{i,5}b^x_{i+1,1}$ (a shortcut edge), or any edge incident to a vertex of the form $s^x_{i,j}$.*

Proof of Claim. Suppose first that a shortcut edge $b^x_{i,5}b^x_{i+1,1}$ belongs to S. (See Figure 13 for indices.) Let $S' = S \setminus \{b^x_{i,5}b^x_{i+1,1}\}$. Since S was inclusion minimal, the graph $G_\varphi - S'$ must contain an induced P_4 that contains the edge $b^x_{i,5}b^x_{i+1,1}$; denote such a P_4 by L. By the assumption that G^x is properly embedded in G_φ we have that L is entirely contained in G^x. Since the stack between pairs of towers i and $i+1$ has height $k' = k_\varphi + 3$, we know that there are at least three vertices of the form $s^x_{i+1,j}$ for some $j \leq k'$ that are not incident to an edge in S. Since L passes through 2 vertices apart from $b^x_{i,5}$ and $b^x_{i+1,1}$, we infer that at least one of these three vertices, say s^x_{i+1,j_0}, is not incident to any edge of S, nor does it lie on L. Create L' by replacing the edge $b^x_{i,5}b^x_{i+1,1}$ in L with the path $b^x_{i,5} - s^x_{i+1,j_0} - b^x_{i+1,1}$. We infer that L' is an induced P_5 in $G_\varphi - S$, which in particular contains an induced P_4. This is a contradiction to the definition of S.

Second, without loss of generality, suppose now that the edge $b^x_{i,5}s^x_{i+1,j}$ belongs to S for some $j \in \{1, 2, \ldots, k'\}$. Let $S' = S \setminus \{b^x_{i,5}s^x_{i+1,j}, b^x_{i+1,1}s^x_{i+1,j}\}$; note here that the edge $b^x_{i+1,1}s^x_{i+1,j}$ might have not belonged to S, but if it had, then we remove it when constructing S'. Since S was inclusion minimal, the graph $G_\varphi - S'$ must contain an induced P_4 that contains the vertex $s^x_{i+1,j}$, thus also one of the vertices $b^x_{i,5}$ or $b^x_{i+1,1}$; denote such a P_4 by L. Again, by the definition of proper embedding, we have that L is entirely contained in G^x. By the same argumentation as before, we infer that there exists a vertex s^x_{i+1,j_0} such that s^x_{i+1,j_0} is not traversed by L and is not incident to an edge of S.


(a) Elimination A (false)


(b) Elimination B (true)


(c) Elimination C

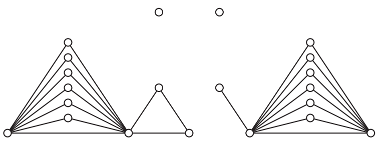

(d) Elimination D

Fig. 14. The four different ways of eliminating a tower pair in a variable gadget. Only Eliminations A and B yield optimum deletion sets in an entire variable gadget. They all use exactly four edges per pair of towers, as is evident in the figure.

Since vertices s^x_{i+1,j_0} and $s^x_{i+1,j}$ are twins in $G_\varphi - S'$, it follows that the path L' constructed from L by substituting $s^x_{i+1,j}$ with s^x_{i+1,j_0} is an induced P_4 in $G_\varphi - S$. This is a contradiction to the definition of S. $\square$

Now, we show that every minimal deletion set S must use at least 4 edges from each pair of towers, and that if it uses exactly 4 edges, then there are exactly 4 ways that the intersection of S with this pair of towers can look like.

CLAIM 5.11. *Assume that the gadget G^x is embedded properly in the output graph G_φ, and that S is an inclusion minimal P_4-free edge deletion set in G_φ of size at most k_φ. Then, for each $i \in \{1, 2, \ldots, p_x\}$, it holds that $|R^x_i \cap S| \geq 4$, and if $|R^x_i \cap S| = 4$, then either:*

Elimination A: $R^x_i \cap S$ consists of the edges of the base of Tower 1 and the spike of Tower 2, or
Elimination B: $R^x_i \cap S$ consists of the edges of the base of Tower 2 and the spike of Tower 1, or
Elimination C: $R^x_i \cap S$ consists of the edges of both spikes and of the base of Tower 1 apart from the edge $b^x_{i,1}b^x_{i,2}$, or
Elimination D: $R^x_i \cap S$ consists of the edges of both spikes and of the base of Tower 2 apart from the edge $b^x_{i,4}b^x_{i,5}$.

We refer to Figure 14 for visualization of all four types of elimination. We will say that $R^x_i \cap S$ *realizes Elimination X* for X being A, B, C, or D if $R^x_i \cap S$ is as described in the statement of Claim 5.11. Similarly, we say that the ith pair of towers *realizes Elimination X* if $R^x_i \cap S$ does.

PROOF OF CLAIM 5.11. By Claim 5.10, we infer that S does not contain any edge incident to stacks i and $i+1$, or any of the shortcut edges incident to the considered pair of towers. We consider four cases, depending on what the set $S \cap \{t^x_{i,1}b^x_{i,2}, t^x_{i,2}b^x_{i,4}\}$ looks like. In each case, we prove that $|R^x_i \cap S| \geq 4$, and that $|R^x_i \cap S| = 4$ implies that one of four listed elimination types is used.

First, assume that $S \cap \{t^x_{i,1}b^x_{i,2}, t^x_{i,2}b^x_{i,4}\} = \emptyset$ and observe that $s^x_{i,1} - b^x_{i,1} - b^x_{i,2} - t^x_{i,1}$ and $s^x_{i+1,1} - b^x_{i,5} - b^x_{i,4} - t^x_{i,2}$ are induced P_4s in G_φ. Since on each of these P_4s there is only one edge that is assumed to be *not* in S, it follows that both $b^x_{i,1}b^x_{i,2}$ and $b^x_{i,4}b^x_{i,5}$ must belong to S. Suppose that $b^x_{i,1}b^x_{i,3} \notin S$. Then, we infer that $b^x_{i,2}b^x_{i,3}, b^x_{i,3}b^x_{i,4}, b^x_{i,3}b^x_{i,5} \in S$, since otherwise any of these edges would form an induced P_4 in $G_\varphi - S$ together with edges $b^x_{i,1}b^x_{i,3}$ and $b^x_{i,1}s^x_{i,1}$. We infer that in this case $|R^x_i \cap S| \geq 5$, and a symmetric conclusion can be drawn when $b^x_{i,3}b^x_{i,5} \notin S$. We are left with the case when $b^x_{i,1}b^x_{i,3}, b^x_{i,3}b^x_{i,5} \in S$. But then, S must include also one of the edges $b^x_{i,2}b^x_{i,3}$ or $b^x_{i,3}b^x_{i,4}$ so that the induced P_4 $t^x_{i,1} - b^x_{i,2} - b^x_{i,3} - b^x_{i,4}$ is destroyed. Hence, in all the considered cases, we conclude that $|R^x_i \cap S| \geq 5$.

Second, assume that $S \cap \{t^x_{i,1}b^x_{i,2}, t^x_{i,2}b^x_{i,4}\} = \{t^x_{i,2}b^x_{i,4}\}$. The same reasoning as in the previous paragraph shows that $b^x_{i,1}b^x_{i,2}$ must belong to S. Again, if $b^x_{i,1}b^x_{i,3} \notin S$, then all the edges $b^x_{i,2}b^x_{i,3}, b^x_{i,3}b^x_{i,4}, b^x_{i,3}b^x_{i,5}$ must belong to S, thus $|R^x_i \cap S| \geq 5$. Assume, then, that $b^x_{i,1}b^x_{i,3} \in S$. Note now that we have two induced P_4s: $t^x_{i,1} - b^x_{i,2} - b^x_{i,3} - b^x_{i,4}$ and $t^x_{i,1} - b^x_{i,2} - b^x_{i,3} - b^x_{i,5}$ that share the edge $t^x_{i,1}b^x_{i,2}$, about which we assumed that it does not belong to S, and the edge $b^x_{i,2}b^x_{i,3}$. To remove both these P_4s, we either remove at least two more edges, which results in the conclusion that $|R^x_i \cap S| \geq 5$, or remove the edge $b^x_{i,2}b^x_{i,3}$, which results in Elimination A.

The third case when $S \cap \{t^x_{i,1}b^x_{i,2}, t^x_{i,2}b^x_{i,4}\} = \{t^x_{i,1}b^x_{i,2}\}$ is symmetric to the second case, and leads to a conclusion that either $|R^x_i \cap S| \geq 5$ or $R^x_i \cap S$ realizes Elimination B.

Finally, assume that $t^x_{i,1}b^x_{i,2}, t^x_{i,2}b^x_{i,4} \in S$. Observe that we have an induced P_4 $s^x_{i,1} - b^x_{i,1} - b^x_{i,3} - b^x_{i,5}$ in G_φ, thus one of the edges $b^x_{i,1}b^x_{i,3}$ or $b^x_{i,3}b^x_{i,5}$ must be included in S. Assume first that $b^x_{i,1}b^x_{i,3} \in S$. Consider now P_4s $s^x_{i,1} - b^x_{i,1} - b^x_{i,2} - b^x_{i,3}$ and $b^x_{i,2} - b^x_{i,3} - b^x_{i,5} - s^x_{i+1,1}$. Both these P_4s need to be destroyed by S since after removing $b^x_{i,1}b^x_{i,3}$, the first P_4 becomes induced, while the second is induced already in G_φ. Moreover, these P_4s share only the edge $b^x_{i,2}b^x_{i,3}$, which means that either $|R^x_i \cap S| \geq 5$ or $b^x_{i,2}b^x_{i,3} \in S$ and $R^x_i \cap S$ realizes Elimination C. The case when $b^x_{i,3}b^x_{i,5} \in S$ is symmetric and leads to a conclusion that either $|R^x_i \cap S| \geq 5$ or $R^x_i \cap S$ realizes Elimination D. □

Finally, we prove that the variable gadget G^x requires at least $4p_x$ edge deletions, and that there are only two ways of destroying all P_4s by using exactly $4p_x$ edge deletions: either by applying Elimination A or Elimination B to all the pairs of towers.

CLAIM 5.12. *Suppose a gadget G^x is embedded properly in the output graph G_φ, and that S is an inclusion minimal P_4-free edge deletion set in G_φ of size at most k_φ. Then $|E(G^x) \cap S| \geq 4p_x$, and if $|E(G^x) \cap S| = 4p_x$, then either $R^x_i \cap S$ realizes Elimination A for all $i \in \{1, 2, \ldots, p_x\}$ or $R^x_i \cap S$ realizes Elimination B for all $i \in \{1, 2, \ldots, p_x\}$.*

PROOF OF CLAIM. By Claims 5.10 and 5.11, we have that S does not contain any shortcut edge or an edge incident to a stack vertex, and moreover that $|R^x_i \cap S| \geq 4$ for all $i \in \{1, 2, \ldots, p_x\}$. Since sets R^x_i are pairwise disjoint, it follows that $|E(G^x) \cap S| \geq 4p_x$. Moreover, if $|E(G^x) \cap S| = 4p_x$, then $|R^x_i \cap S| = 4$ for all $i \in \{1, 2, \ldots, p_x\}$ and, by Claim 5.11, for all $i \in \{1, 2, \ldots, p_x\}$ the set $R^x_i \cap S$ must realize Elimination A, B, C, or D.

We say that one pair of towers is *followed* by another if the former has index i and the latter has index $i + 1$ (of course, modulo p_x). To obtain the conclusion that either all the sets $R^x_i \cap S$ realize Elimination A or all realize Elimination B, we observe that when some pair of towers realize Elimination A, C, or D, then the following pair must realize Elimination A. Indeed, otherwise the graph $G_\varphi - S$ would contain an induced P_4 of the form $b^x_{i,4} - b^x_{i,5} - b^x_{i+1,1} - b^x_{i+1,2}$, where the ith pair of towers is the considered pair that realizes Elimination A, C, or D. Now, observe that since the pairs of towers are arranged on a cycle, then either all pairs of towers realize Elimination B or at least one realizes Elimination A, C, or D, which means that the following pair realizes Elimination A, thus all pairs must realize Elimination A. □

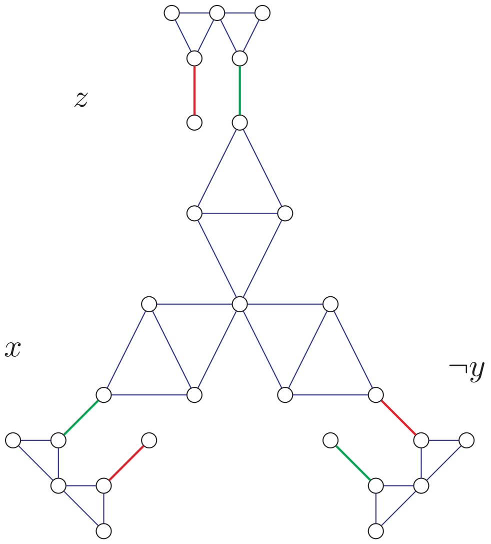


Fig. 15. For a clause $c = x \vee \neg y \vee z$, we obtain the connection in this figure. For negated variables, the rightmost spike is attached to the gadgets; otherwise, the leftmost spike is attached. If x is given a value satisfying c, the edge spike between G^x and G^c is deleted.

Clause gadget. We now move on to the construction of the clause gadget G^c for a clause $c \in \mathcal{C}(\varphi)$. Assume that $c = \ell_x \vee \ell_y \vee \ell_z$, where ℓ_r is a literal of variable r for $r \in \{x, y, z\}$. We create seven vertices: one vertex u^c and vertices u^r_2 and u^r_3 for $r = x, y, z$. We also add the edges $u^c u^r_2$, $u^c u^r_3$ and $u^r_2 u^r_3$. Now, for nonnegated $r \in \{x, y, z\}$ in c, where c is the ith clause r appears in, we add edges $u^r_2 t^r_{i,1}$ and $u^r_3 t^r_{i,1}$ (recall that $t^r_{i,1}$ is the spike of Tower 1 in tower pair i). If r appears negated, we add the edges $u^r_2 t^r_{i,2}$ and $u^r_3 t^r_{i,2}$ instead (see Figure 15). Let M^c be the set comprising all 15 created edges, including the ones incident to the spikes of the towers. By $M^{c,r}$ for $r \in \{x, y, z\}$, we denote the subset of M^c containing 5 edges that are incident to vertex u^r_2 or u^r_3.

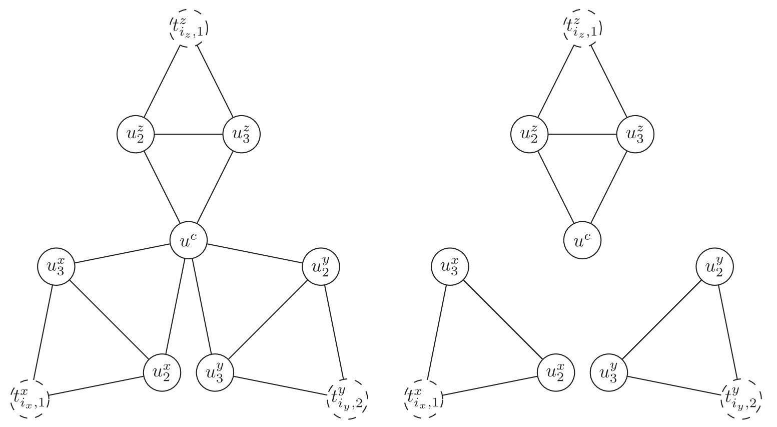
This concludes the construction of the graph G_φ; note that all the variable gadgets are properly embedded in G_φ. Before showing the correctness of the reduction, we prove the following claims about the number of edges needed for clause gadgets:

Claim 5.13. *Assume that S is a P_4-free deletion set of graph G_φ. Let c be a clause of φ, and assume that x, y, z are the variables appearing in c. Then $|S \cap M^c| \geq 4$, and if $|S \cap M^c| = 4$, then $S \cap M^{c,r} = \emptyset$ for some $r \in \{x, y, z\}$ (see Figure 16(b) for an example where $S \cap M^{c,z} = \emptyset$).*

Proof of Claim. To simplify the notation, let t^x, t^y, t^z be the corresponding vertices of the variable gadgets that are incident to edges of M^c.

If $|S \cap M^{c,r}| \geq 2$ for all $r \in \{x, y, z\}$, then $|S \cap M^c| \geq 6$, and we are done. Assume, then, without loss of generality that $|S \cap M^{c,x}| \leq 1$. Hence, at least one of the paths $t^x - u^x_2 - u^c$ and $t^x - u^x_3 - u^c$ does not contain an edge of S. Assume without loss of generality that it is $t^x - u^x_2 - u^c$. Now, observe that in G_φ we have 4 induced P_4s created by prolonging this P_3 by vertex u^y_2, u^y_3, u^z_2, or u^z_3. Since $t^x - u^x_2 - u^c$ is disjoint with S, it follows that all the four edges connecting these vertices with u^c must belong to S. Hence $|S \cap M^c| \geq 4$, and if $|S \cap M^c| = 4$, then $M^{c,x}$ must actually be disjoint with S. $\square$

We are finally ready to prove the following lemma, which implies the correctness of the reduction.



(a) Clause gadget for a clause $c = x \vee \neg y \vee z$. The dashed vertices are the connection points in the variable gadgets. Observe that since y appears negated, we attach it to Tower 2 in its pair. The clause c is the i_ℓth clause ℓ appears in.

(b) Clause gadget elimination when c is satisfied by variable z.

Fig. 16. Clause gadget G^c for a clause $c = x \vee \neg y \vee z$. On the left it is before elimination, and on the right after an optimal elimination when satisfied by z.

Lemma 5.14. *Given an input instance φ to* 3Sat*, φ is satisfiable if and only if the constructed graph G_φ has a P_4 deletion set of size $k_\varphi = 16|\mathcal{C}(\varphi)|$.*

Proof. In the forward direction, suppose that φ is satisfiable by an assignment α, and let G_φ and k_φ be as earlier. If a variable x is assigned false in α, we delete as in Figure 14(a), that is, we apply Elimination A to all the pairs of towers in the variable gadget G^x. Otherwise, we delete as in Figure 14(b), that is, we apply Elimination B to all pairs of towers in the variable gadget G^x. In other words, if x is assigned false (Elimination A), we delete the edges $t^x_{i,2}b^x_{i,4}$, $b^x_{i,1}b^x_{i,2}$, $b^x_{i,1}b^x_{i,3}$, and $b^x_{i,2}b^x_{i,3}$; otherwise, when x is assigned true (Elimination B), we delete the edges $t^x_{i,1}b^x_{i,2}$, $b^x_{i,3}b^x_{i,4}$, $b^x_{i,3}b^x_{i,5}$, and $b^x_{i,4}b^x_{i,5}$, for all $i \in \{1, \dots, p_x\}$.

Furthermore, for every clause $c = \ell_x \vee \ell_y \vee \ell_z$, we choose an arbitrary variable whose literal satisfies c, say r. We remove the edges $u^{r'}_2 u^c$ and $u^{r'}_3 u^c$ for $r' \neq r$. We have thus used exactly four edge removals per clause, $4|\mathcal{C}(\varphi)|$ in total, and for each $x \in \mathcal{V}(\varphi)$ we have removed $4p_x$ edges. This sums up exactly to $4|\mathcal{C}(\varphi)| + \sum_{x\in\mathcal{V}(\varphi)} 4p_x = 4|\mathcal{C}(\varphi)| + 4\sum_{x\in\mathcal{V}(\varphi)} p_x = 4|\mathcal{C}(\varphi)| + 4 \cdot 3|\mathcal{C}(\varphi)| = 16|\mathcal{C}(\varphi)| = k_\varphi$ edge removals.

We now claim that G_φ is P_4-free. A direct check shows that there is no induced P_4 left inside any variable gadget, nor inside any clause gadget. Therefore, any induced P_4 left must necessarily contain a vertex of the form $t^x_{i,q}$ for some $x \in \mathcal{V}(\varphi)$, $i \in \{1, 2, \dots, p_x\}$, and $q \in \{1, 2\}$, together with the edge of the spike incident to this vertex and one of the edges of gadget G^c incident to this vertex, where c is the ith clause x appears in. Assume without loss of generality that $q = 1$, thus x appears in c positively. Since we did not delete the spike edge $t^x_{i,1}b^x_{i,2}$, we infer that $\alpha(x) = \texttt{false}$. Therefore, x does not satisfy c, thus we must have deleted edges $u^c u^x_2$ and $u^c u^x_3$. Therefore, in the remaining graph $G_\varphi - S$, the connected component of the vertex $t^x_{i,q}$ is a triangle with a pendant edge, which is P_4-free. We conclude that $G_\varphi - S$ is indeed P_4-free.

In the reverse direction, suppose now that G_φ is the graph constructed from a fixed φ and that for k_φ as before, we have that (G_φ, k_φ) is a yes instance of P_4-Free Edge Deletion. Let S be a P_4 deletion set of size at most k_φ, and without loss of generality assume that S is inclusion minimal. By Claims 5.12 and 5.13, set S must contain at least $4p_x$ edges from each set $E(G^x)$, and at least four edges from each set M^c. Since $4|\mathcal{C}(\varphi)| + \sum_{x\in\mathcal{V}(\varphi)} 4p_x = k_\varphi$, we infer that S contains exactly $4p_x$ edges from each set $E(G^x)$ and exactly four edges in each set M^c. By Claim 5.12, we infer that for each variable x, all pairs of towers in G^x realize Elimination A, or all realize Elimination B. Let $\alpha : \mathcal{V}(\varphi) \to \{\texttt{true}, \texttt{false}\}$ be an assignment that assigns value `false` if Elimination A is used throughout the corresponding gadget, and value `true` otherwise. We claim that α satisfies φ.

Consider a clause $c \in \mathcal{C}(\varphi)$ and assume that x, y, z are variables appearing in c. By Claim 5.13, we infer that there exists $r \in \{x, y, z\}$ such that $S \cap M^{c,r} = \emptyset$. Assume without loss of generality that $r = x$, and that x appears positively in c. Moreover, assume that c is the i_xth clause x appears in. We claim that $\alpha(x) = \texttt{true}$, and thus c is satisfied by x. Indeed, otherwise the edge $t^x_{i_x,1}b^x_{i_x,2}$ would not be deleted, thus $b^x_{i_x,2} - t^x_{i_x,1} - u^x_2 - u^c$ would be an induced P_4 in $G_\varphi - S$. This is a contradiction to the definition of S. $\square$

Again, the proof of Theorem 5.9 follows: combining the presented reduction with an algorithm that solves P_4-Free Edge Deletion in $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ time would give an algorithm that solves 3Sat in $2^{o(n+m)} \cdot (n+m)^{\mathcal{O}(1)}$ time, which contradicts ETH by the results of Impagliazzo et al. [2001].

It is easy to verify that in the presented reduction, both the graph G_φ and $G_\varphi - S$ for S being the deletion set constructed for a satisfying assignment for φ are actually C_4-free. Thus, the same reduction also shows that $\{C_4, P_4\}$-free Deletion is not solvable in $2^{o(k)}n^{\mathcal{O}(1)}$ time unless ETH fails. Since $\overline{P_4} = P_4$ and $\overline{C_4} = 2K_2$, it follows that $\{2K_2, P_4\}$-Free Completion is hard under ETH as well. In other words, we derive the following result: Co-Trivially Perfect Completion is not solvable in subexponential time unless ETH fails.

Theorem 5.15. *The problem* $\{2K_2, P_4\}$-Free Completion *is NP-complete and not solvable in* $2^{o(k)}n^{\mathcal{O}(1)}$ *time unless ETH fails.*

6. CONCLUSION AND FUTURE WORK

In this article, we provided several upper and lower subexponential parameterized bounds for $\mathcal{F}$-Completion. The most natural open question would be to ask for a dichotomy characterizing for which sets $\mathcal{F}$, $\mathcal{F}$-Completion problems are in P, in SUBEPT, and not in SUBEPT (under ETH). Keeping in mind the lack of such characterization concerning classes P and NP, an answer to this question can be very nontrivial. Even a more modest task—deriving general arguments explaining what causes a completion problem to be in SUBEPT—is an important open question.

Similarly, from an algorithmic perspective, obtaining generic subexponential algorithms for completion problems would be a big step forwards. With current knowledge, for different cases of $\mathcal{F}$, the algorithms are built on different ideas such as chromatic coding, potential maximal cliques, and k-cuts, and each new case requires special treatment.

Another interesting property is that all graph classes for which subexponential algorithms for completion problems are known are tightly connected to chordal graphs. Indeed, all known algorithms exploit existence of a chordal-like decomposition of the target completed graph. Are there natural NP-hard graph modification problems admitting subexponential time algorithms where the graph class target is *not* related to chordal graphs?

Finally, in this article, we have presented SUBEPT lower bounds (under ETH) for $\mathcal{F}$-Completion for several different cases of $\mathcal{F}$, but we lack a method for proving tight lower bounds on the running time for problems that actually are in SUBEPT. For instance, it may be the case that Trivially Perfect Completion or Chordal Completion can be solved in time $2^{\mathcal{O}(k^{1/4})}n^{\mathcal{O}(1)}$. As Fomin and Villanger [2013] observed, in the case of Chordal Completion, known NP-hardness reductions provide lower bounds much weaker than the current upper bound of $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$. However, we feel that a $2^{o(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ running time should be impossible to achieve, since such an algorithm would immediately imply the existence of an exact algorithm with running time $2^{o(n)}$. Is it possible to prove $2^{o(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ lower bounds under ETH for Trivially Perfect Completion, Chordal Completion, and other completion problems to subclasses of chordal graphs known to be contained in SUBEPT? These are intriguing questions to pursue in future work.

REFERENCES

Noga Alon, Daniel Lokshtanov, and Saket Saurabh. 2009. Fast fast. In *Proceedings of the 36th Colloquium of Automata, Languages and Programming (ICALP)*. Lecture Notes in Computer Science, Vol. 5555. Springer, Berlin, 49–58.

Ivan Bliznets, Fedor V. Fomin, Marcin Pilipczuk, and Michał Pilipczuk. 2014a. A subexponential parameterized algorithm for interval completion. *CoRR* abs/1402.3473.

Ivan Bliznets, Fedor V. Fomin, Marcin Pilipczuk, and Michał Pilipczuk. 2014b. A subexponential parameterized algorithm for proper interval completion. In *Proceedings of the European Symposium on Algorithms (ESA)*. Lecture Notes in Computer Science, Vol. 8737. Springer, Berlin, 173–184.

Hans L. Bodlaender. 1998. A partial *k*-Arboretum of graphs with bounded treewidth. *Theoretical Computer Science* 209, 1–2, 1–45.

Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. 1999. *Graph Classes. A Survey*. SIAM, Philadelphia, PA.

Pablo Burzyn, Flavia Bonomo, and Guillermo Durán. 2006. NP-Completeness results for edge modification problems. *Discrete Applied Mathematics* 154, 13, 1824–1844.

Leizhen Cai. 1996. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters* 58, 4, 171–176.

Leizhen Cai and Yufei Cai. 2015. Incompressibility of H-Free edge modification problems. *Algorithmica* 71, 3, 731–757.

Erik D. Demaine, Fedor V. Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M. Thilikos. 2005. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *Journal of the ACM* 52, 6, 866–893.

Pål Grønås Drange, Fedor V. Fomin, Michał Pilipczuk, and Yngve Villanger. 2014. Exploring subexponential parameterized complexity of completion problems. In *Proceedings of the 31st International Symposium on Theoretical Aspects of Computer Science (STACS) (LIPIcs)*, Vol. 25. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 288–299.

Pål Grønås Drange and Michał Pilipczuk. 2015. A polynomial kernel for trivially perfect editing. In *European Symposium on Algorithms (ESA)*, N. Bansal and I. Finocchi (Eds.). LNCS 9294, Springer, Heidelberg, 424–436.

Jörg Flum and Martin Grohe. 2006. *Parameterized Complexity Theory*. Springer-Verlag, New York.

Fedor V. Fomin, Stefan Kratsch, Marcin Pilipczuk, Michał Pilipczuk, and Yngve Villanger. 2014. Tight bounds for parameterized complexity of cluster editing with a small number of clusters. *Journal of Computer and System Sciences* 80, 7, 1430–1447.

Fedor V. Fomin and Yngve Villanger. 2013. Subexponential parameterized algorithm for minimum fill-in. *SIAM Journal on Computing* 42, 6, 2197–2216.

Esha Ghosh, Sudeshna Kolay, Mrinal Kumar, Pranabendu Misra, Fahad Panolan, Ashutosh Rai, and M. S. Ramanujan. 2015. Faster parameterized algorithms for deletion to split graphs. *Algorithmica* 71, 4, 989–1006.

Martin Charles Golumbic. 1980. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, NY.

Sylvain Guillemot, Frédéric Havet, Christophe Paul, and Anthony Perez. 2013. On the (non-)existence of polynomial kernels for P_l-free edge modification problems. *Algorithmica* 65, 4, 900–926.

Jiong Guo. 2007. Problem kernels for NP-complete edge deletion problems: Split and related graphs. In *Algorithms and Computation, 18th International Symposium (ISAAC).* Lecture Notes in Computer Science, Vol. 4835. Springer, Berlin, 915–926.

Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. 2001. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences* 63, 4, 512–530.

Yan Jing-Ho, Chen Jer-Jeong, and Gerard J. Chang. 1996. Quasi-threshold graphs. *Discrete Applied Mathematics* 69, 3, 247–255.

Haim Kaplan and Ron Shamir. 1996. Pathwidth, bandwidth, and completion problems to proper interval graphs with small cliques. *SIAM Journal on Computing* 25, 3, 540–561.

Haim Kaplan, Ron Shamir, and Robert E. Tarjan. 1999. Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. *SIAM Journal on Computing* 28, 5, 1906–1922.

Christian Komusiewicz and Johannes Uhlmann. 2012. Cluster editing with locally bounded modifications. *Discrete Applied Mathematics* 160, 15, 2259–2270.

Stefan Kratsch and Magnus Wahlström. 2013. Two edge modification problems without polynomial kernels. *Discrete Optimization* 10, 3, 193–199.

Frédéric Maffray and Myriam Preissmann. 1994. Linear recognition of pseudo-split graphs. *Discrete Applied Mathematics* 52, 3, 307–312.

Nadimpalli V. R. Mahadev and Uri N. Peled. 1995. *Threshold Graphs and Related Topics*. Annals of Discrete Mathematics, Vol. 56. Elsevier.

Federico Mancini. 2008. *Graph Modification Problems Related to Graph Classes*. Ph.D. Dissertation. University of Bergen, Bergen, Norway.

Assaf Natanzon, Ron Shamir, and Roded Sharan. 2000. A polynomial approximation algorithm for the minimum fill-in problem. *SIAM Journal on Computing* 30, 1067–1079. Issue 4.

Jaroslav Nešetřil and Patrice Ossona de Mendez. 2012. *Sparsity – Graphs, Structures, and Algorithms*. Algorithms and combinatorics, Vol. 28. Springer.

Yngve Villanger, Pinar Heggernes, Christophe Paul, and Jan Arne Telle. 2009. Interval completion is fixed parameter tractable. *SIAM Journal on Computing* 38, 5, 2007–2020.

Mihalis Yannakakis. 1981a. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic Discrete Methods* 2, 1, 77–79.

Mihalis Yannakakis. 1981b. Edge-deletion problems. *SIAM Journal on Computing* 10, 2, 297–309.

Received April 2014; revised June 2015; accepted June 2015