On the Threshold of Intractability

Pål Grønås Drange

University of Bergen

Markus Fanebust Dregi

University of Bergen

Daniel Lokshtanov

University of California, Santa Barbara

Blair D. Sullivan

University of Utah

Abstract

The computational complexity of the graph modification problems Threshold Editing and Chain Editing has been an important open question in computational graph theory for more than 15 years. These problems consist of adding and deleting as few edges as possible to transform a graph into a threshold or chain graph. We show that both problems are NP-hard, resolving a conjecture by Natanzon, Shamir, and Sharan from 2001.

On the positive side, we show both problems admit quadratic vertex kernels, and give a subexponential time parameterized algorithm solving Threshold Editing in $2^{O(\sqrt{k}\log k)} + \text{poly}(n)$ time, making it one of relatively few natural problems in this complexity class on general graphs. Here n is the size of the input graph and k the prescribed budget.

Finally, we show that all our positive results extend to the related problem of Chain Editing, as well as the completion and deletion variants of both problems. These are the only known graph classes for which all three versions—completion, deletion, and editing—are NP-complete as well as solvable in subexponential parameterized time.

Keywords: Edge editing, threshold graphs, parameterized complexity
2000 MSC: 68W40, 68Q17, 68Q25, 68Q27, 68R10, 05C82, 91D30

Email address: `Pal.Drange@uib.no` (Pål Grønås Drange)
A preliminary version appeared in ESA 2015 [10].

1. Introduction

In this paper we study the computational complexity of two edge modification problems that have arisen in the study of complex networks—namely editing to threshold and chain graphs. Graph modification problems ask whether a given graph G can be transformed to have a certain property using a small number of edits (such as deleting/adding vertices or edges), and have been the subject of significant previous theoretical work [31, 7, 8, 9, 26].

Graph editing problems are well-motivated by problems arising in the applied sciences, where we often have a predicted model from domain knowledge, but observed data fails to fit this model exactly. In this setting, edge modification corresponds to correcting false positives (and/or false negatives) to obtain data that is consistent with the model. A specific problem of recent interest in the social sciences is THRESHOLD EDITING. Threshold graphs have been studied as a theoretical basis for complex networks exhibiting "scale–free" [25] and core–periphery [29] structures. The editing problem is also considered in work of Brandes et al. using distance to threshold graphs in an axiomatization of centrality measures [2, 30]. More generally, editing to threshold graphs and their close relatives *chain graphs* arises in the study of sparse matrix multiplications [33].

In the THRESHOLD EDITING problem, we are given as input an n-vertex graph $G = (V, E)$ and a non-negative integer k. The objective is to find a set F of at most k pairs of vertices such that G minus any edges in F plus all non-edges in F is a threshold graph. A graph is a *threshold graph* if it can be constructed from the empty graph by repeatedly adding either an isolated vertex or a universal vertex [3].

THRESHOLD EDITING	
Input:	A graph $G = (V, E)$ and a non-negative integer k
Question:	Is there a set $F \subseteq [V]^2$ of size at most k such that $G \oplus F$ is a threshold graph.

The computational complexity of THRESHOLD EDITING has repeatedly been stated as open, starting from Natanzon et al. [28], and then more recently by Burzyn et al. [4], and again very recently by Liu, Wang, Guo and Chen [21]. We resolve this by showing that the problem is indeed NP-hard. Natanzon et al. showed that THRESHOLD EDITING can be solved in polynomial time on bounded degree input graphs [28].

Theorem 1. THRESHOLD EDITING *is* NP*-complete, even on split graphs.*

Chain graphs are the bipartite analogue of threshold graphs (see Definition 2.6), and here we also establish hardness of CHAIN EDITING.

Theorem 2. CHAIN EDITING *is* NP*-complete, even on bipartite graphs.*

Our final complexity result is for CHORDAL EDITING—a problem whose NP-hardness is well-known and widely used. This result also follows from our techniques, and as the authors were unable to find a proof in the literature, we include this argument for the sake of completeness.

Having settled the complexity of these problems, we turn to studying ways of dealing with their intractability. Cai's theorem [5] shows that Threshold Editing and Chain Editing are *fixed parameter tractable*, i.e., solvable in $f(k) \cdot \text{poly}(n)$ time where k is the edit distance from the desired model (graph class); However, the lower bounds we prove when showing NP-hardness are on the order of $2^{o(\sqrt{k})}$ under ETH, and thus leave a gap. We show that it is in fact the lower bound which is tight (up to logarithmic factors in the exponent) by giving a subexponential time algorithm for both problems.

Theorem 3. Threshold Editing *and* Chain Editing *admit* $2^{O(\sqrt{k}\log k)}+\text{poly}(n)$ *subexponential time algorithms.*

Since our results also hold for the *completion* and *deletion* variants of both problems (when F is restricted to be a set of non-edges or edges, respectively), this also answers a question of Liu et al. [22] by giving a subexponential time algorithm for Chain Edge Deletion.

A crucial first step in our algorithms is to preprocess the instance, reducing to a kernel of size polynomial in the parameter. We give quadratic kernels for all three variants (of both Threshold Editing and Chain Editing).

Theorem 4. Threshold Editing, Threshold Completion, *and* Threshold Deletion *admit polynomial kernels with* $O(k^2)$ *vertices.*

This answers (affirmatively) a recent question of Liu, Wang and Guo [20]—whether the previously known kernel, with $O(k^3)$ vertices, for Threshold Completion (equivalently Threshold Deletion) can be improved.

2. Preliminaries

Graphs. We will consider only undirected simple finite graphs. For a graph G, let $V(G)$ and $E(G)$ denote the vertex set and the edge set of G, respectively. For a vertex $v \in V(G)$, by $N_G(v)$ we denote the open neighborhood of v, i.e. $N_G(v) = \{u \in V(G) \mid uv \in E(G)\}$. The closed neighborhood of v, denoted by $N_G[v]$, is defined as $N_G(v) \cup \{v\}$. These notions are extended to subsets of vertices as follows: $N_G[X] = \bigcup_{v \in X} N_G[v]$ and $N_G(X) = N_G[X] \setminus X$. We omit the subscript whenever G is clear from context.

When $U \subseteq V(G)$ is a subset of vertices of G, we write $G[U]$ to denote the *induced subgraph* of G, i.e., the graph $G' = (U, E_U)$ where E_U is $E(G)$ restricted to U. The degree of a vertex $v \in V(G)$, denoted $\deg_G(v)$, is the number of vertices it is adjacent to, i.e., $\deg_G(v) = |N_G(v)|$. We denote by $\Delta(G)$ the maximum degree in the graph, more specifically $\Delta(G) = \max_{v \in V(G)} \deg(v)$.

For a set A, we write $[A]^2$ to denote the set of unordered pairs of elements of A; thus $E(G) \subseteq [V(G)]^2$. By $\overline{G}$ we denote the *complement* of graph G, i.e., $V(\overline{G}) = V(G)$ and $E(\overline{G}) = [V(G)]^2 \setminus E(G)$. For two sets A and B we define the *symmetric difference* of A and B, denoted $A \oplus B$ as the set $(A \setminus B) \cup (B \setminus A)$. For a graph $G = (V, E)$ and $F \subseteq [V]^2$ we define $G \oplus F$ as the graph $(V, E \oplus F)$.

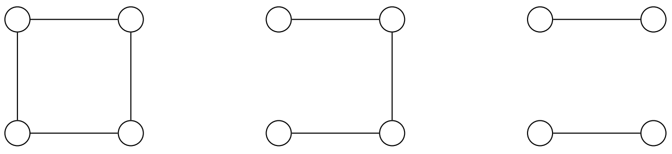

Figure 1: *Threshold graphs* are $\{C_4, P_4, 2K_2\}$-free. *Chain graphs* are bipartite graphs that are $2K_2$-free.

For a graph G and a vertex v we define the *true twin class* of v, denoted $\mathrm{ttc}(v)$ as the set of vertices $\{u \in V(G) \mid N[u] = N[v]\}$. Similarly, we define the *false twin class* of v, denoted $\mathrm{ftc}(v)$ as the set $\{u \in V(G) \mid N(u) = N(v)\}$. Observe that at least one of $\mathrm{ttc}(v)$ and $\mathrm{ftc}(v)$ is $\{v\}$. From this we define the *twin class* of v, denoted $\mathrm{tc}(v)$ as $\mathrm{ttc}(v)$ if $|\mathrm{ttc}(v)| > |\mathrm{ftc}(v)|$ and $\mathrm{ftc}(v)$ otherwise.

Split and threshold graphs. A split graph is a graph $G = (V, E)$ whose vertex set can be partitioned into two sets C and I such that $G[C]$ is a complete graph and $G[I]$ is edgeless, i.e., C is a clique and I an independent set [3]. For a split graph G we say that a partition (C, I) of $V(G)$ forms a *split partition* of G if $G[C]$ induces a clique and $G[I]$ an independent set. A split partition (C, I) is called a *complete* split partition if for every vertex $v \in I$, $N(v) = C$. If G admits a complete split partition, we say that G is a complete split graph.

The original definition of a *threshold graph* is the following: A graph is a threshold graph if it is possible to assign real weights to the vertices $w \colon V \to [0, 1]$ such that two vertices u and v are adjacent if and only if $w(u) + w(v) \geq 1$. We now give two characterizations of threshold graphs that will be more useful for us:

Proposition 2.1 ([23])**.** *A graph G is a* threshold graph *if and only if G has a split partition (C, I) such that the neighborhoods of the vertices in I are nested, i.e., for every pair of vertices v and u in I, either $N(v) \subseteq N[u]$ or $N(u) \subseteq N[v]$.*

Proposition 2.2 ([3])**.** *A graph G is a threshold graph if and only if G does not have a C_4, P_4 nor a $2K_2$ as an induced subgraph. Thus, the threshold graphs are exactly the* $\{C_4, P_4, 2K_2\}$-free graphs *(see Figure 1).*

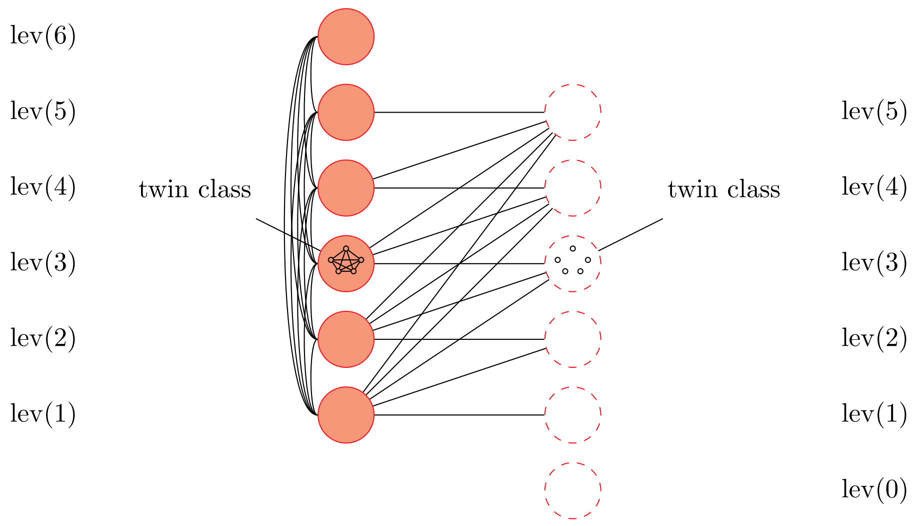

Figure 2: A threshold partition—the left hand side is the clique and the right hand side is an independent set, each bag contains a twin class. All bags are non-empty, otherwise two twin classes on the opposite side would collapse into one, except possibly the two extremal bags.

Definition 2.3. We say that $(\mathcal{C}, \mathcal{I}) = (\langle C_1, \dots, C_t \rangle, \langle I_1, \dots, I_t \rangle)$ forms a *threshold partition* of G if the following holds (see Figure 2 for an illustration):

- (C, I) is a split partition of G, where $C = \bigcup_{i \leq t} C_i$ and $I = \bigcup_{i \leq t} I_i$,
- C_i and I_i are (non-empty) twin classes in G for every i
- $N[C_j] \subset N[C_i]$ and $N(I_i) \subset N(I_j)$ for every $i < j$.
- Finally, we demand that for every $i \leq t$, $(C_i, I_{\geq i})$ form a complete split partition of the graph induced by $C_i \cup I_{\geq i}$, where $I_{\geq i} = \bigcup_{j \geq i} I_j$

We furthermore define, for every vertex v in G, $\text{lev}(v)$ as the number i such that $v \in C_i \cup I_i$ and we denote each level $L_i = C_i \cup I_i$.

In a threshold decomposition we will refer to C_i for every i as a *clique fragment* and I_i as an *independent fragment*. Furthermore, we will refer to a vertex in $\cup\mathcal{C}$ as a *clique vertex* and a vertex in $\cup\mathcal{I}$ as an *independent vertex*.

Proposition 2.4 (Threshold decomposition)**.** *A graph G is a threshold graph if and only if G admits a threshold partition.*

Proof. Suppose that G is a threshold graph and therefore admits a nested ordering of the neighborhoods of vertices of each of the two sets C and I (Proposition 2.1). We show that partitioning the graph into sets depending only on their degree yields the levels of a threshold partition. The clique side is naturally defined as the maximal set of highest degree vertices that form a clique. Suppose now for contradiction that this did not constitute a threshold partition. From Proposition 2.1, since the neighborhoods are nested, and we partition into sets of vertices with the same degree, every level consists of twin classes, and also, for two twin classes I_i and I_j, since their neighborhoods are nested in the threshold graph, their neighborhoods are nested in the threshold partition as well. So what is left to verify is that $(C_i, I_{\geq i})$ is a complete split partition of $G[C_i \cup I_{\geq i}]$. But that follows directly from the assumption that G admitted a nested ordering and C_i is a true twin class.

For the reverse direction, suppose G admits a threshold partition $(\mathcal{C}, \mathcal{I})$. Recall that threshold graphs exactly the $\{2K_2, C_4, P_4\}$-free graphs. Since it admits a threshold partition, it is a split graph, and hence already $\{2K_2, C_4\}$-free. What remains is to show that G does not have an induced P_4. Suppose towards a contradiction that there are four vertices a, b, c, d that induce a P_4 in G. Again, since G is a split graph, the vertices b and c must be in the clique set and a and d in the independent set. But b and c are not twins, nor is $N(b) \subset N(c)$ nor $N(c) \subset N(b)$, which contradicts the fact that $N[C_j] \subset N[C_i]$ for every $i < j$. Hence, the graph cannot contain an induced P_4, and G is thus a threshold graph. $\square$

Lemma 2.5. *For every instance (G, k) of* Threshold Editing *or* Threshold Completion *it holds that there exists an optimal solution F such that for every pair of vertices $u, v \in V(G)$, if $N_G(u) \subseteq N_G[v]$ then $N_{G \oplus F}(u) \subseteq N_{G \oplus F}[v]$.*

Proof. Let us define, for any editing set F and two vertices u and v, the set

$$F_{v \leftrightarrow u} = \{e \mid e' \in F \text{ and } e \text{ is } e' \text{ with } u \text{ and } v \text{ switched}\}.$$

Suppose F is an optimal solution for which the above statement does not hold. Then $N_G(u) \subseteq N_G[v]$ and $N_{G \oplus F}(v) \subseteq N_{G \oplus F}[u]$ (see Proposition 2.1). But then it is easy to see that we can flip edges in an ordering such that at some point, say after flipping F^0, u and v are twins in this intermediate graph $G \oplus F^0$. Let $F^1 = F \setminus F^0$. It is clear that for $G' = G \oplus (F^0 \cup F^1_{v \leftrightarrow u})$, $N_{G'}(u) \subseteq N_{G'}[v]$. Since $|F| \geq |F^0 \cup F^1_{v \leftrightarrow u}|$, it follows that $F^0 \cup F^1_{v \leftrightarrow u}$ is also optimal and therefore the claim holds. $\square$

Chain graphs. Chain graphs are the bipartite graphs whose neighborhoods of the vertices on one of the sides form an inclusion chain. It follows that the neighborhoods on the opposite side form an inclusion chain as well. If this is the case, we say that the neighborhoods are *nested*. The relation to threshold graphs is obvious, see Figure 3 for a comparison. The problem of completing edges to obtain a chain graph was introduced by Golumbic [17] and later studied by Yannakakis [33], Feder, Mannila and Terzi [13] and finally by Fomin and Villanger [15] who showed that Chain Completion when given a bipartite graph whose bipartition must be respected is solvable in subexponential time.

Definition 2.6 (Chain graph)**.** A bipartite graph $G = (A, B, E)$ is a chain graph if there is an ordering of the vertices of A, $a_1, a_2, \ldots, a_{|A|}$ such that $N(a_1) \subseteq N(a_2) \subseteq \cdots \subseteq N(a_{|A|})$.

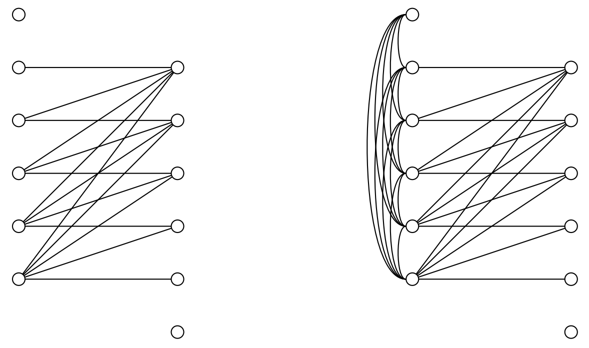
(a) A chain graph (b) A threshold graph

Figure 3: Illustration of the similarities between chain and threshold graphs. Note that the nodes drawn can be replaced by twin classes of any size, even empty. However, if on one side of a level there is an empty class, the other two levels on the opposite side will collapse to a twin class. See Proposition 2.4.

From the following proposition, it follows that chain graphs are characterized by a finite set of forbidden induced subgraphs and hence are subject to Cai's theorem [5].

Proposition 2.7 ([3]). *Let G be a graph. The following are equivalent:*

- *G is a chain graph.*
- *G is bipartite and $2K_2$-free.*
- *G is $\{2K_2, C_3, C_5\}$-free.*
- *G can be constructed from a threshold graph by removing all the edges in the clique partition.*

Since they have the same structure as threshold graphs, it is natural to talk about a *chain decomposition*, $(\mathcal{A}, \mathcal{B})$ of a bipartite graph G with bipartition (A, B). We say that $(\mathcal{A}, \mathcal{B})$ is a chain decomposition for a chain graph G if and only if $(\mathcal{A}, \mathcal{B})$ is a threshold decomposition for the corresponding threshold graph G' where A is made into a clique.

Parameterized complexity. The running time of an algorithm in classical complexity analysis is described as a function of the length of the input. To refine the analysis of computationally hard problems, especially NP-hard problems, parameterized complexity introduced the notion of an extra "parameter"—an additional part of a problem instance used to measure the problem complexity when the parameter is taken into consideration. To simplify the notation, here we consider inputs to problems to be of the form (G, k)—a pair consisting of a graph G and a nonnegative integer k. We will say that a problem is *fixed parameter tractable* whenever there is an algorithm solving the problem in time $f(k) \cdot \text{poly}(|G|)$, where f is any function, and poly: $\mathbb{N} \to \mathbb{N}$ any polynomial function. In the case when $f(k) = 2^{o(k)}$ we say that the algorithm is a subexponential parameterized algorithm. When a problem

$\Pi \subseteq \mathcal{G} \times \mathbb{N}$ is fixed-parameter tractable, where $\mathcal{G}$ is the class of all graphs, we say that Π belongs to the complexity class FPT. For a more rigorous introduction to parameterized complexity we refer to the book of Flum and Grohe [14].

Given a parameterized problem Π, we say two instances (G, k) and (G', k') are *equivalent* if $(G, k) \in \Pi$ if and only if $(G', k') \in \Pi$. A *kernelization algorithm* (or *kernel*) is a polynomial-time algorithm for a parameterized problem Π that takes as input a problem instance (G, k) and returns an equivalent instance (G', k'), where both $|G'|$ and k' are bounded by $f(k)$ for some function f. We then say that f is the *size of the kernel.* When $k' \leq k$, we say that the kernel is a *proper kernel.* Specifically, a proper polynomial kernelization algorithm for Π is a polynomial time algorithm which takes as input an instance (G, k) and returns an equivalent instance (G', k') with $k' \leq k$ and $|G'| \leq p(k)$ for some polynomial function p.

Laminar set system. We call a set system *laminar* if its member sets are pairwise either nested or disjoint (consider subtrees of a rooted tree). One crucial property that we will use of laminar set systems is that their sizes are bounded linearly by the ground set, as the following lemma attests. But first, a formal definition:

Definition 2.8 (Laminar set system)**.** A set system $\mathcal{F}$ over a finite ground set U is *laminar* if for every two sets $X_1 \in \mathcal{F}$ and $X_2 \in \mathcal{F}$, either $X_1 \subseteq X_2$, $X_2 \subseteq X_1$, or $X_1 \cap X_2 = \emptyset$.

Lemma 2.9 (Folklore)**.** *Let $\mathcal{F}$ be a laminar set system over a finite ground set U. Then the cardinality of $\mathcal{F}$ is at most $2|U|$.*

Proof sketch. By associating the elements of U with the leaves of a rooted tree where each internal non-root node has degree at least three, a non-empty element of F corresponds exactly to a rooted induced subtree. Since this tree can have at most $2|U| - 1$ nodes, by adding the possibility of the empty set, we obtain the result. □

3. Hardness

In this section we show that Threshold Editing is NP-complete. Recalling (see Figure 3) that chain graphs are bipartite graphs with structure very similar to that of threshold graphs, it should not be surprising that we obtain as a corollary that Chain Editing is NP-complete as well.

We will also conclude the section by giving a proof for the fact that Chordal Editing is NP-complete; Although this has been known for a long time (Natanzon [27], Natanzon et al. [28], Sharan [32]), the authors were unable to find a proof in the literature for the NP-completeness of Chordal Editing and therefore include the observation. The problem was recently shown to be FPT by Cao and Marx [6], however we would like to point out that the more general problem studied there is indeed well-known to be NP-complete as it is a generalized version of Chordal Vertex Deletion.

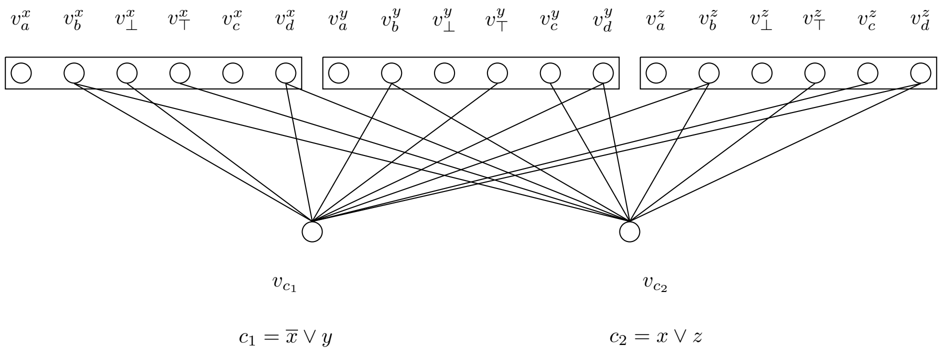
v_a^x v_b^x $v_\perp^x$ $v_\top^x$ v_c^x v_d^x v_a^y v_b^y $v_\perp^y$ $v_\top^y$ v_c^y v_d^y v_a^z v_b^z $v_\perp^z$ $v_\top^z$ v_c^z v_d^z
v_{c_1}
v_{c_2}
$c_1 = \overline{x} \vee y$
$c_2 = x \vee z$

Figure 4: The connections of a clause and a variable. All the vertices on the top (the variable vertices) belong to the clique, while the vertices on the bottom (the clause vertices) belong to the independent set. The vertices in the left part of the clique has higher degree than the vertices of the right part of the clique, whereas all the clause vertices (in the independent set) will all have the same degree, namely $3 \cdot |\mathcal{V}_\varphi|$. To ensure the ordering of the vertices, we add k^2 extra vertices in the independent set; Those are not illustrated here.

3.1. NP-completeness of Threshold Editing

Recall that a boolean formula φ is in 3-CNF-SAT if it is in conjunctive normal form and each clause has at most three variables. Our hardness reduction is from the problem 3Sat, where we are given a 3-CNF-SAT formula φ and asked to decide whether φ admits a satisfying assignment. We will denote by $\mathcal{C}_\varphi$ the set of clauses, and by $\mathcal{V}_\varphi$ the set of variables in a given 3-CNF-SAT formula φ. An *assignment* for a formula φ is a function $\alpha\colon \mathcal{V}_\varphi \to \{\texttt{true}, \texttt{false}\}$. Furthermore, we assume we have some natural lexicographical ordering $<_{\text{lex}}$ of the clauses $c_1, \dots, c_{|\mathcal{C}_\varphi|}$ and the same for the variables $v_1, \dots, v_{|\mathcal{V}_\varphi|}$, hence we may write, for some variables x and y, that $x <_{\text{lex}} y$. To immediately get an impression of the reduction we aim for, the construction is depicted in Figure 4.

3.1.1. Construction

Recall that we want to form a graph G_φ and pick an integer k_φ so that (G_φ, k_φ) is a yes-instance of Threshold Editing if and only if φ is satisfiable. We will design G_φ to be a split graph, so that the split partition is forced to be maintained in any threshold graph within distance k_φ of G_φ, where $k_\varphi = |\mathcal{C}_\varphi| \cdot (3|\mathcal{V}_\varphi| - 1)$.

Given φ, we first create a clique of size $6|\mathcal{V}_\varphi|$; To each variable $x \in \mathcal{V}_\varphi$, we associate six vertices of this clique, and order them in the following manner

$$v_a^x, v_b^x, v_\perp^x, v_\top^x, v_c^x, v_d^x.$$

We will throughout the reduction refer to this ordering as π_φ: π_φ is a partial order which has

$$v_a^x <_{\pi_\varphi} v_b^x <_{\pi_\varphi} v_\top^x, v_\perp^x <_{\pi_\varphi} v_c^x <_{\pi_\varphi} v_d^x,$$

and for every two vertices $v_\star^x$ and $v_\star^y$ with $x <_{\text{lex}} y$, we have $v_\star^x <_{\pi_\varphi} v_\star^y$. Observe that we do not specify which comes first of $v_\top^x$ and $v_\bot^x$—this is the choice that will result in the assignment α for φ.

We enforce this ordering by adding $O(k_\varphi^2)$ vertices in the independent set; Enforcing that v_1 comes before v_2 in the ordering is done by adding $k_\varphi + 1$ vertices in the independent set incident to all the vertices coming before v_1, including v_1. Since swapping the position of v_1 and v_2 would demand at least $k_\varphi + 1$ edge modifications and k_φ is the intended budget, in any yes-instance, v_1 ends up before v_2 in the ordering of the clique.

We proceed adding the clause gadgets: For every clause $c \in \mathcal{C}_\varphi$, we add one vertex v_c to the independent set. Hence, the size of the independent set is $O(|\mathcal{C}_\varphi| + k_\varphi^2)$. For a variable x occurring in c, we add an edge between v_c and $v_\bot^x$ if it occurs negatively, and between v_c and $v_\top^x$ otherwise. In addition, we make v_c incident to v_b^x and v_d^x.

For a variable z which does not occur in a clause c, we make v_c adjacent to v_b^z, v_c^z, and v_d^z. To complete the reduction, we add $4(k_\varphi+1)$ isolated vertices; $k_\varphi+1$ vertices to the left in the independent set, $k_\varphi + 1$ vertices to the right in the independent set, and $k_\varphi + 1$ to the left and $k_\varphi + 1$ to the right in the clique. This ensures that no vertex will move from the clique to the independent set partition, and vice versa.

3.1.2. Properties of the Constructed Instance

Before proving Lemma 3.4, which directly implies Theorem 1, we can observe the following, which may serve as an intuition for the idea of the reduction. When we consider a fixed permutation of the variable gadget vertices (the clique side), the only thing we need to determine for a clause vertex v_c, is the *cut-off point*: the point in π_φ at which the vertex v_c will no longer have any neighbors; If the permutation of the clique side is fixed, a vertex in the independent set will be adjacent to a prefix of the cliques. The cut-off point is defined as the size of this prefix. Observing that no vertex v_i^x swaps places with any other v_j^x for $i, j \in \{a, b, c, d\}$, and that no $v_\star^x$ changes with $v_\star^y$ for $x, y \in \mathcal{V}_\varphi$, consider a fixed permutation of the variable vertices. We charge the clause vertices with the edits incident to the clause vertex. Since the budget is $k_\varphi = |\mathcal{C}| \cdot (3|\mathcal{V}_\varphi| - 1)$, and every clause needs at least $3|\mathcal{V}_\varphi| - 1$, to obtain a solution (upcoming Lemma 3.2) we need to charge every clause vertex with exactly $3|\mathcal{V}_\varphi| - 1$ edits. Figure 5 illustrates the charged cost of a clause vertex.

Observation 3.1. *The graph G_φ resulting from the above procedure is a split graph and when $k_\varphi = |\mathcal{C}| \cdot (3|\mathcal{V}_\varphi| - 1)$, if H is a threshold graph within distance k_φ of G_φ, H must have the same clique-maximizing split partition as G_φ.*

Lemma 3.2. *Let (G_φ, k_φ) be a yes-instance to* Threshold Editing *constructed from a 3-CNF-SAT formula φ, and let F be a solution of size at most k_φ for the instance. For any clause vertex v_c, at least $3|\mathcal{V}_\varphi| - 1$ edges in F are incident to v_c.*

Proof. By the properties of π_φ, we know that the only vertices we may change the order of are those corresponding to $v_\top^\star$ and $v_\bot^\star$. Pick any index in π_φ for which we know that v_c is adjacent to all vertices on the left hand side and non-adjacent to all vertices on the right hand side. Let L_c be the set of variables whose vertices are

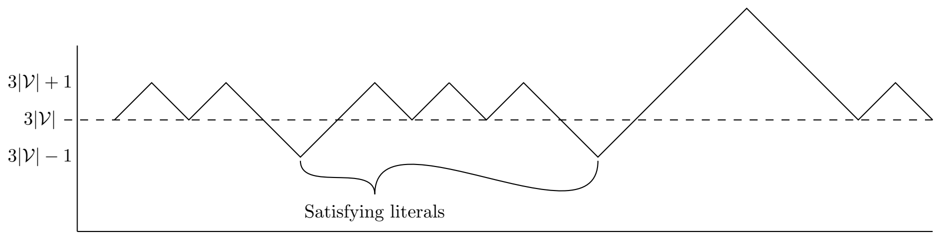

Figure 5: The cost with which we charge a clause vertex depends on the cut-off point; The x-axis denotes the point in the lexicographic ordering which separates the vertices adjacent to the clause vertex from the vertices not adjacent to the clause vertex.

completely adjacent to v_c and R_c the corresponding set completely non-adjacent to v_c. By construction, v_c has exactly three neighbors in each variable and thus these variable gadgets contribute $3(|L_c|+|R_c|)$ to the budget. If $L_c \cup R_c = \mathcal{V}_\varphi$, we are done, as v_c needs at least $3|\mathcal{V}_\varphi|$ edits here.

Suppose therefore that in $G_\varphi \oplus F$ there is a variable x whose vertex v_a^x is adjacent to v_c and v_d^x is non-adjacent to v_c. But then we have already deleted the existing edge $v_c v_d^x$ and added the non-existing edge $v_c v_a^x$. This immediately gives a lower bound on $3(|\mathcal{V}_\varphi|-1)+2 = 3|\mathcal{V}_\varphi|-1$ edits. □

3.1.3. Proof of Correctness

Lemma 3.3. *If there is an editing set F of size at most k_φ for an instance (G_φ, k_φ) constructed from a 3-CNF-SAT formula φ, and $|F(v_c)| = 3|\mathcal{V}_\varphi|-1$, then the $<_{\text{lex}}$-highest vertex connected to v_c corresponds to a variable satisfying the clause c.*

Proof. From the proof of Lemma 3.2, we observed that for a clause c to be within budget, we must choose a cut-off point within a variable gadget, meaning that there is a variable x for which, in $G_\varphi \oplus F$, v_c is adjacent to v_a^x and non-adjacent to v_d^x.

We now distinguish two cases, *(i)* x is a variable occurring (w.l.o.g. positively) in c and *(ii)* x does not occur in c. For *(i)*, v_c was in G_φ adjacent to v_b^x, $v_\top^x$, and v_d^x. By assumption, F adds the edge $v_c v_a^x$ and deletes the edge $v_c v_d^x$. But then we have already spent the entire budget, hence the only way this is a legal editing, $v_\top^x$ must come before $v_\bot^x$, and hence satisfies v_c. See Figure 6.

For *(ii)* we have that v_c was in G_φ adjacent to v_b^x, v_c^x, and v_d^x. Here F, again by assumption, adds the edge $v_c v_a^x$ and deletes the edge $v_c v_d^x$. This alone costs two edits, so we are done. But observe that these two edits alone are not enough, hence if we want to achieve the goal of $3|\mathcal{V}_\varphi|-1$ edited edges, the cut-off index must be inside a variable gadget corresponding to a variable occurring in c, i.e. *(i)* must be the case. □

Lemma 3.4. *A 3-CNF-SAT formula φ is satisfiable if and only if (G_φ, k_φ) is a yes-instance to* Threshold Editing.

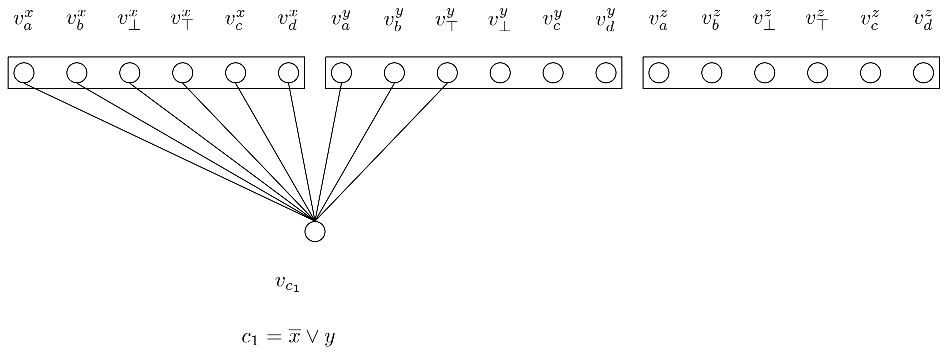
v_a^x v_b^x $v_\bot^x$ $v_\top^x$ v_c^x v_d^x v_a^y v_b^y $v_\top^y$ $v_\bot^y$ v_c^y v_d^y v_a^z v_b^z $v_\bot^z$ $v_\top^z$ v_c^z v_d^z
v_{c_1}
$c_1 = \overline{x} \vee y$

Figure 6: The edited version when y satisfies c_1. We have added three edges to the gadget x and deleted three edges to the gadget z, and added the edge to v_a^y and deleted the edge to v_d^y, that is, we have edited exactly $3 \cdot 2 + 2 = 3(|\mathcal{V}| - 1) + 2 = 3|\mathcal{V}| - 1$ edges incident to c_1. Notice that if $v_\bot^y$ was coming before $v_\top^y$, we would have to choose a different variable to satisfy c_1.

Proof of Lemma 3.4. For the forwards direction, let φ be a satisfiable 3-CNF-SAT formula where $\alpha\colon \mathcal{V}_\varphi \to \{\texttt{true}, \texttt{false}\}$ is any satisfying assignment, and (G_φ, k_φ) the Threshold Editing instance as described above. Furthermore, let π be any permutation of the vertices of the clique side with the following properties

- for every $x <_{\text{lex}} y \in \mathcal{V}_\varphi$, we have $v_\star^x <_\pi v_\star^y$,
- for every $x \in \mathcal{V}_\varphi$, we have $v_a^x <_\pi v_b^x <_\pi v_\top^x < v_c^x <_\pi v_d^x$ and $v_a^x <_\pi v_b^x <_\pi v_\bot^x < v_c^x <_\pi v_d^x$, and finally
- for every $x \in \mathcal{V}_\varphi$, we have $v_\bot^x <_\pi v_\top^x$ if and only if $\alpha(x) = \texttt{false}$.

We now show how to construct the threshold graph H_φ^π from the constructed graph G_φ by editing exactly $k_\varphi = |\mathcal{C}| \cdot (3|\mathcal{V}_\varphi| - 1)$ edges. For a clause c, let x be any variable satisfying c. If x appears positively, add every non-existing edge from v_c to every vertex $v \leq_\pi v_\top^x$ and delete all the rest. If x appears negated, use $v_\bot^x$ instead. We break the remainder of the proof in the forward direction into two claims:

Claim 3.5. *H_φ^π is a threshold graph.*

Proof of Claim 3.5. Let G_φ and π be given, both adhering to the above construction. Since G_φ was a split graph, π a total ordering of the elements in the independent set part and every vertex of the clique part of H_φ^π sees a prefix of the vertices of the independent set, their neighborhoods are naturally nested. Hence H_φ^π is a threshold graph by Proposition 2.1. □

Claim 3.6. $\left|E(G_\varphi) \oplus E(H_\varphi^\pi)\right| = k_\varphi$.

Proof of Claim 3.6. Since we did not edit any of the edges within the clique part nor the independent set part, we only need to count the number of edits going between a clause vertex and the variable vertices. Let c be any clause and x the lexicographically smallest variable satisfying c. Suppose furthermore, without loss of generality, that x appears positively in c and has thus $\alpha(x) = \texttt{true}$. We now show that $|F(v_c)| = 3|\mathcal{V}_\varphi| - 1$, and since c was arbitrary, this concludes the proof of the claim. Since in G_φ, we have that v_c is adjacent to exactly three vertices per variable, and non-adjacent to exactly three vertices per variable, we added all the edges to the vertices appearing before x and removed all the edges to the vertices appearing after x. This cost exactly $3(|\mathcal{V}_\varphi| - 1) = 3|\mathcal{V}_\varphi| - 3$, hence we have two edges left in our budget for c. Moreover, the edge $v_c v_a^x$ was added and the edge $v_c v_d^x$ was deleted. Now, c is adjacent to every vertex to the before, and including, x, and non-adjacent to all the vertices after x. The budget used was $3(|\mathcal{V}_\varphi| - 1) + 2 = 3|\mathcal{V}_\varphi| - 1$. Hence, the total number of edges edited to obtain H_φ^π is $\sum_{c \in \mathcal{C}} 3|\mathcal{V}_\varphi| - 1 = |\mathcal{C}| \cdot (3|\mathcal{V}_\varphi| - 1) = k_\varphi$. □

This shows that if φ is satisfiable, then (G_φ, k_φ) is a yes-instance of Threshold Editing.

In the reverse direction, let (G_φ, k_φ) be a constructed instance from a given 3-CNF-SAT formula φ and let F be a minimal editing set such that $G_\varphi \oplus F$ is a threshold graph and $|F| \leq k_\varphi$. We aim to construct a satisfying assignment $\alpha \colon \mathcal{V}_\varphi \to \{\texttt{true}, \texttt{false}\}$ from $G_\varphi \oplus F$. By Observation 3.1, $H = G_\varphi \oplus F$ has the same split partition as G_φ. By construction, we have enforced the ordering, π_φ, of each of the vertices corresponding to the variables. Thus, we know exactly how H looks, with the exception of the internal ordering of each literal and its negation. Construct the assignment α as described above, i.e., $\alpha(x) = \texttt{false}$ if and only if $v_\perp^x <_\pi v_\top^x$.

By Lemmata 3.2 and 3.3, it follows directly that α is a satisfying assignment for φ which concludes the proof of the main lemma. □

The above lemma shows that there is a polynomial time many-one (Karp) reduction from 3Sat to Threshold Editing so we may wrap up the main theorem of this section. Lemma 3.4 implies Theorem 1, that Threshold Editing is NP-complete, even on split graphs.

For the sake of the next section, devoted to the proof of Theorem 2, we define the following annotated version of editing to threshold graphs. In this problem, we are given a split graph and we are asked to edit the graph to a threshold graph while respecting the split partition.

Split Threshold Editing	
Input:	A split graph $G = (V, E)$ with split partition (C, I), and an integer k.
Question:	Is there an editing set $F \subseteq C \times I$ of size at most k such that $G \oplus F$ is a threshold graph?

Corollary 3.7. Split Threshold Editing *is* NP*-complete.*

Proof. Split Threshold Editing is clearly in NP and that the problem is NP-complete follows immediately from combining Lemma 3.4 with Observation 3.1. □

Corollary 3.8. *Assuming ETH, neither* Threshold Editing *nor* Split Threshold Editing *are solvable in* $2^{o(\sqrt{k})} \cdot \mathrm{poly}(n)$ *time.*

3.2. NP-hardness of Chain Editing and Chordal Editing

3.2.1. Chain Graphs: Proof of Theorem 2

A bipartite graph $G = (A, B, E)$ is a *chain graph* if the neighborhoods of A are nested (which necessarily implies the neighborhoods of B are nested as well). Recalling Proposition 2.7, chain graphs are closely related to threshold graphs; Given a bipartite graph $G = (A, B, E)$, if one replaces A (or B) by a clique, the resulting graph is a threshold graph if and only if G was a chain graph.

It immediately follows from the above exposition that the following problem is NP-complete. This problem has also been referred to as Chain Editing in the literature (for instance in the work by Guo [18]).

Bipartite Chain Editing	
Input:	A bipartite graph $G = (A, B, E)$ and an integer k
Question:	Does there exist a set $F \subseteq A \times B$ of size at most k such that $G \oplus F$ is a chain graph?

Observe that we in this problem are given a bipartite graph together with a bipartition, and we are asked to respect the bipartition in the editing set.

Corollary 3.9. *The problem* Bipartite Chain Editing *is* NP*-complete.*

Proof. We reduce from Split Threshold Editing. Recall that to this problem, we are given a split graph $G = (V, E)$ with split partition (C, I), and an integer k, and asked whether there is an editing set $F \subseteq C \times I$ of size at most k such that $G \oplus F$ is a threshold graph. Since a chain graph is a threshold graph with the edges in the clique partition removed (Proposition 2.7), it follows that $G \oplus F$ with all the edges in the clique partition removed is a chain graph.

Let (G, k) be the input to Split Threshold Editing and let (C, I) be the split partition. Remove all the edges in C to obtain a bipartite graph $G' = (A, B, E')$. Now it follows directly from Proposition 2.7 that (G, k) is a yes-instance to Split Threshold Editing if and only if (G', k) is a yes-instance to Bipartite Chain Editing. □

Chain Editing	
Input:	A graph $G = (V, E)$ and a non-negative integer k
Question:	Is there a set F of size at most k such that $G \oplus F$ is a chain graph?

We now aim to prove Theorem 2, that Chain Editing is NP-complete.

Proof of Theorem 2. Reduction from Bipartite Chain Editing. Let $G = (A, B, E)$ be a bipartite graph and consider the input instance (G, k) to Bipartite Chain

Editing. We now show that adding $2(k+1)$ new vertices to G to obtain a graph $G' = (V', E')$, gives us that (G', k) is a yes-instance for Chain Editing if and only if (G, k) is a yes-instance for Bipartite Chain Editing.

Let $G = (A, B, E)$ be a bipartite graph and k a positive integer. Add $k+1$ new vertices $a_1, \cdots a_{k+1}$ to A and make them universal to B, and add $k+1$ new vertices $b_1, \cdots b_{k+1}$ to B and make them universal to A. Call the resulting graph $G' = (V', E')$.

The following claim follows immediately from the construction.

Claim 3.10. *If $G' \oplus F$ is a chain graph with $|F| \leq k$, then $G' \oplus F$ has bipartition $(A \cup \{a_1, \dots, a_{k+1}\}, B \cup \{b_1, \dots, b_{k+1}\})$.*

It follows that for any instance (G, k) to Bipartite Chain Editing, the instance (G', k) as constructed above is a yes-instance for Chain Editing if and only if (G, k) is a yes-instance for Bipartite Chain Editing. □

Corollary 3.11. *Assuming ETH, there is no algorithm solving neither* Chain Editing *nor* Bipartite Chain Editing *in time* $2^{o(\sqrt{k})} \cdot \text{poly}(n)$.

Proof. In both these cases we reduced from Split Threshold Editing without changing the parameter k. Hence this follows immediately from the above exposition and from Corollary 3.8. □

3.2.2. Chordal Graphs

We will now combine our previous result on Chain Editing with the following observation of Yannakakis to prove Theorem 5. Yannakakis showed [33], while proving the NP-completeness of Chordal Completion (more often known as Minimum Fill-In [15]), that a bipartite graph can be transformed into a chain graph by adding at most k edges if and only if the cobipartite graph formed by completing the two sides can be transformed into a chordal graph by adding at most k edges.

Theorem 5. Chordal Editing *is* NP*-hard.*

To prove the theorem, we will first give an intermediate problem that makes the proof simpler. Let $G = (A, B, E)$ be a cobipartite graph. Define the problem Cobipartite Chordal Editing to be the problem which on input (G, k) asks if we can edit at most k edges between A and B, i.e., does there exist an editing set $F \subseteq A \times B$ of size at most k, such that $G \oplus F$ is a chordal graph. That is, Cobipartite Chordal Editing asks for the bipartition A, B to be respected.

Cobipartite Chordal Editing	
Input:	A cobipartite graph $G = (A, B, E)$ and an integer k
Question:	Does there exist a set $F \subseteq A \times B$ of size at most k such that $G \oplus F$ is a chordal graph?

We will use the following observation to prove the above theorem:

Lemma 3.12. *If $G = (A, B, E)$ is a bipartite graph, and $G' = (A, B, E')$ is the cobipartite graph constructed from G by completing A and B, then F is an optimal edge editing set for* Bipartite Chain Editing *on input (G, k) if and only if F is an optimal edge editing set for* Cobipartite Chordal Editing *on input (G', k).*

Proof. Let F be an optimal editing set for Bipartite Chain Editing on input (G, k) and suppose that $G' \oplus F$ has an induced cycle of length at least four. Since G' is cobipartite, it has a cycle of length exactly four. Let $a_1b_1b_2a_2a_1$ be this cycle. But then it is clear that a_1b_1, a_2b_2 forms an induced $2K_2$ in $G \oplus F$, contradicting the assumption that F was an editing set (see Proposition 2.7).

For the reverse direction, suppose F is an optimal edge editing set for Cobipartite Chordal Editing on input (G', k) only editing edges between A and B. Suppose for the sake of a contradiction that $G \oplus F$ was not a chain graph. Since $F \subseteq A \times B$, $G \oplus F$ is bipartite and hence by the assumption must have an induced $2K_2$ (recall again Proposition 2.7). This obstruction must be on the form a_1b_1, a_2b_2, but then $a_1b_1b_2a_2a_1$ is an induced C_4 in $G' \oplus F$ which is a contradiction to the assumption that $G' \oplus F$ was chordal. Hence $G \oplus F$ is a chain graph. ☐

Corollary 3.13. Cobipartite Chordal Editing *is* NP*-complete.*

We are now ready to prove Theorem 5.

Proof of Theorem 5. Let $(G = (A, B, E), k)$ be a cobipartite graph as input to Cobipartite Chordal Editing. Our reduction is as follows. Create $G' = (A' \cup B', E')$ as follows:

- $A' = A \cup \{a_1, a_2, \dots, a_{k+1}\}$,
- $B' = B \cup \{b_1, b_2, \dots, b_{k+1}\}$,
- $E' = E \cup \bigcup_{i \le k+1, b \in B'} \{a_i b\} \cup \bigcup_{a \in A', i \le k+1} \{a b_i\} \cup \bigcup_{i,j \le k+1} \{a_i a_j, b_i b_j\}$

Finally, we create G'' as follows. For every edge a_ia_j in A' create $k+1$ new vertices adjacent to only a_i and a_j. Do the same thing for every edge b_ib_j in B'. This forces none of the edges in A' to be removed and none of the edges in B' to be removed.

Claim 3.14. *The instance of* Chordal Editing *(G'', k) is equivalent to the instance (G, k) to* Cobipartite Chordal Editing.

Proof of claim. The proof of the above claim is straight-forward. If we delete an edge within A (resp. B), we create at least $\binom{k+1}{2}$ cycles of length 4 namely $a_ix_1a_jx_2a_i$ where a_ia_j is the removed edge and x_1 and x_2 are two of the newly introduced vertices for the given edge. This subgraph is isomorphic to $K_{k+1,2}$, and unless we add a_ia_j back, we must edit at least k edges to make it chordal. Furthermore, any chordal graph remains chordal when adding a simplicial vertex, which is exactly what the $k+1$ new vertices are. ☐

From the claim it follows that (G'', k) is a yes-instance to Chordal Editing if and only if (G, k) is a yes-instance to Cobipartite Chordal Editing. The theorem follows immediately from Corollary 3.13. ☐

Corollary 3.15. *Assuming ETH, there is no algorithm solving* Chordal Editing *in time* $2^{o(\sqrt{k})} \cdot \mathrm{poly}(n)$.

4. Kernels for Modifications into Threshold and Chain Graphs

First we give kernels with quadratically many vertices for the following three problems: Threshold Completion, Threshold Deletion, and Threshold Editing. By this we answer a recent question of Liu, Wang and Guo [20]. Then we continue by providing kernels with quadratically many vertices for Chain Completion, Chain Deletion, and Chain Editing. Our kernelization algorithms uses techniques similar to the previous result that Trivially Perfect Editing admits a polynomial kernel [12]. Observe that the class of threshold graphs is closed under taking complements. It follows that for every instance (G, k) of Threshold Completion, $(\bar{G}, k)$ is an equivalent instance of Threshold Deletion (and vice versa). Almost the same trick applies to Chain Deletion. Due to this, we restrict our attention to the completion and editing variants for the remainder of the section. Motivated by the characterization of threshold graphs in Propositions 2.2 and 2.7, we define obstructions (also see Figure 1).

Definition 4.1 ($\mathcal{H}$, Obstruction)**.** A graph H is a *threshold obstruction* if it is isomorphic to a member of the set $\{C_4, P_4, 2K_2\}$ and a *chain obstruction* if it is isomorphic to a member of the set $\{C_3, 2K_2, C_5\}$. If it is clear from the context, we will often use the term obstruction for both threshold and chain obstructions and denote the set of obstructions by $\mathcal{H}$. Furthermore, if an obstruction H is an induced subgraph of a graph G we call H an *obstruction in* G.

Definition 4.2 (Realizing)**.** For a graph G and a set of vertices $X \subseteq V(G)$ we say that a vertex $v \in V(G) \setminus X$ is *realizing* $Y \subseteq X$ if $N_X(v) = Y$. Furthermore, we say that a set $Y \subseteq X$ is *being realized* if there is a vertex $v \in V(G) \setminus X$ such that v is realizing Y.

Before proceeding, we observe that our kernelization algorithms does not modify any edges, and only changes the budget in the case that we discover that we have a no-instance (in which case we return $(H, 0)$, where H is an obstruction in G). The only modification of the instance is to delete vertices, hence the kernelized instance is an induced subgraph of the original graph. Since the parameter is never increased, we obtain *proper kernels.*

4.1. Modifications into Threshold Graphs

We now focus on modifications to threshold graphs and obtaining kernels for these operations.

4.1.1. Outline of the Kernelization Algorithm

The kernelization algorithm consists of a twin reduction rule and an irrelevant vertex rule. The twin reduction rule is based on the observation that any obstruction containing vertices from a large enough twin class will have to be handled by edges not incident to the twin class. From this observation, we may conclude that for any twin class, we may keep only a certain amount without affecting the solutions.

A key concept of the irrelevant vertex rule is what will be referred to as a *threshold-modulator*. A threshold-modulator is a set of vertices X in G of linear size in k, such that for every obstruction H in G one can add and remove edges in $[X]^2$ to turn H into a non-obstruction. First, we prove that we can in polynomial time either obtain such a set X or conclude correctly that the instance is a no-instance. The observation that $G - X$ is a threshold graph will be exploited heavily and we now fix a threshold decomposition $(\mathcal{C}, \mathcal{I})$ of $G - X$. We then prove that the idea of Proposition 2.1 can be extended to vertices in $G - X$ when considering their neighborhoods in G. In other words, the neighborhoods of the vertices in $G - X$ are nested also when considering G. This immediately yields that the number of subsets of X that are being realized is bounded linearly in the size of X and hence also in k.

We now either conclude that the graph is small or we identify a sequence of levels in the threshold decomposition containing many vertices, such that all the clique vertices and all the independent set vertices in the sequence have identical neighborhoods in X, respectively. The crux is that in the middle of such a sequence there will be a vertex that is replaceable by other vertices in every obstruction and hence is irrelevant. Such a sequence is obtained by discarding all levels in the decomposition that are extremal with respect to a subset Y of X, meaning that there either are no levels above or underneath that contain vertices realizing Y. One can prove that in this process, only a quadratic number of vertices are discarded and from this we obtain a kernel.

4.1.2. The Twin Reduction Rule

First, we introduce the twin reduction rule as described above. For the remainder of the section we will assume this rule to be applied exhaustively and hence we can assume all twin classes to be small.

Rule 1 (Twin reduction rule)**.** *Let (G, k) be an instance of* Threshold Completion *or* Threshold Editing *and v a vertex in G such that $|\operatorname{tc}(v)| > 2k + 2$. We then reduce the instance to $(G - v, k)$.*

Lemma 4.3. *Let G be a graph and v a vertex in G such that $|\operatorname{tc}(v)| > 2k + 2$. Then for every k we have that (G, k) is a yes-instance of* Threshold Completion *(or* Threshold Editing*) if and only if $(G - v, k)$ is a yes-instance of* Threshold Completion *(resp.* Threshold Editing*).*

Proof. For readability we only consider Threshold Completion, however the exact same proof works for Threshold Editing. Let $G' = G - v$. It trivially holds that if (G, k) is a yes-instance, then also (G', k) is a yes-instance. This is due to the fact that removing a vertex never will create new obstructions.

Now, let (G', k) be a yes-instance and assume for a contradiction that (G, k) is a no-instance. Let F be an optimal solution of (G', k) and W an obstruction in $(G \oplus F, k)$. Since W is not an obstruction in G' it follows immediately that v is in W. Furthermore, since $|F| \leq k$ it follows that there are two vertices $a, b \in$

$\text{tc}(v) \setminus \{v\}$ that F is not incident to. Also, one can observe that no obstruction contains more than two vertices from a twin class and hence we can assume without loss of generality that b is not in W. It follows that $N_{G \oplus F}(v) \cap (W - v) = N_G(v) \cap (W - v) = N_G(b) \cap (W - v) = N_{G'}(b) \cap (W - v)$ and hence the graph induced on $V(W) \oplus \{b, v\}$ is an obstruction in $G' \oplus F$, contradicting that F is a solution. □

4.1.3. The Modulator

To obtain an $O(k^2)$ kernel we aim at an irrelevant vertex rule. However, this requires some tools. The first one is the concept of a threshold-modulator, as defined below.

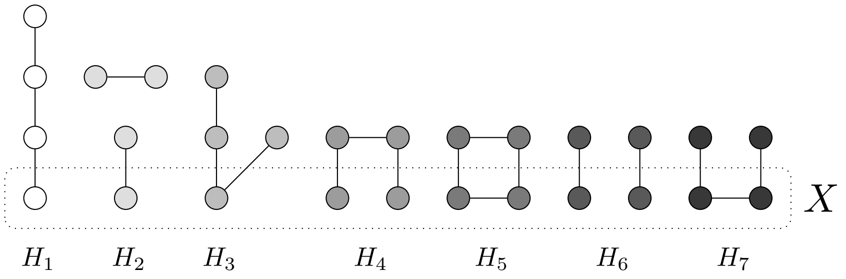

Figure 7: Some of the intersections of an obstruction with a threshold-modulator X that will not occur by definition. More specifically the ones necessary for the proof of the kernel.

Definition 4.4 (Threshold modulator)**.** Let G be a graph and $X \subseteq V(G)$ a set of vertices. We say that X is a *threshold-modulator* of G if for every obstruction W in G it holds that there is a set of edges F in $[X]^2$ such that $W \oplus F$ is not an obstruction.

Less formally, a set X is a threshold-modulator of a graph G if for every obstruction W in G you can edit edges between vertices in X to turn W into a non-obstruction. Our kernelization algorithm will heavily depend on finding a small threshold-modulator X and the fact that $G - X$ is a threshold graph.

Lemma 4.5. *There is a polynomial time algorithm that given a graph G and an integer k either*

- *outputs a threshold-modulator X of G such that $|X| \leq 4k$ or*
- *correctly concludes that (G, k) is a no-instance of both* Threshold Completion *and* Threshold Editing.

Proof. Let X_1 be the empty set and $\mathcal{W} = \{W_1, \ldots, W_t\}$ the set of all obstructions in G. We execute the following procedure for every W_i in $\mathcal{W}$: If $W_i \oplus F$ is an obstruction for every $F \subseteq [X_i \cap V(W_i)]^2$ we let $X_{i+1} = X_i \cup V(W_i)$, otherwise we let $X_{i+1} = X_i$. After we have considered all obstructions we let $X = X_{t+1}$. If $|X| > 4k$ we conclude that (G, k) is a no-instance, otherwise we output X.

Since all obstructions are finite the algorithm described clearly runs in polynomial time. We now argue that X is a threshold-modulator of G. If $V(W_i)$ was added to X_{i+1}, we let F be all the non-edges of W_i. Since $W_i \oplus F$ is isomorphic to K_4 it follows immediately that $W_i \oplus F$ is not an obstruction. If W_i was not added to X_{i+1}, let F the set found in $[X_i \cap V(W_i)]^2$ such that $W_i \oplus F$ is not an obstruction. Observe that in both cases above, $F \subseteq [X]^2$ and hence X is a threshold-modulator.

It remains to prove that if $|X| > 4k$ then (G, k) is a no-instance of Threshold Editing. Observe that it will follow immediately that (G, k) is a no-instance of Threshold Completion. Since every obstruction consists of four vertices there was at least $k+1$ obstructions added during the procedure. We can assume without loss of generality that $W_1, \dots, W_{k+1}$ was added. Observe that by construction, a solution must contain an edge in $[X_{i+1} - X_i]^2$ for every $i \in [k+1]$ and hence contains at least $k+1$ edges. □

4.1.4. Obtaining Structure

We now exploit the threshold-modulator and its interaction with the remaining graph to obtain structure. First, we prove that the neighborhoods of the vertices in $V(G) \setminus X$ are nested in G and that the number of realized sets in X are bounded linearly in k.

Lemma 4.6. *Let G be a graph and X a threshold-modulator. For every pair of vertices u and v in $G - X$ it holds that either $N_G(u) \subset N_G[v]$ or $N_G(v) \subset N_G[u]$.*

Proof. Assume otherwise for a contradiction and let u' be a vertex in $N_G(u) \setminus N_G[v]$ and v' a vertex in $N_G(v) \setminus N_G[u]$, i.e., u' is a "private neighbor" of u and v' is a "private neighbor" of v, which means that $\{u, v, u', v'\}$ will induce either $2K_2$, P_4, or C_4 in G. Since X is a threshold-modulator, $\{u', v'\} \subseteq X$. Regardless of whether there is an edge between u and v, and regardless of whether there is an edge between u' and v', these four vertices form one of the illegal interactions defined by the threshold-modulator. Specifically, $\{u, v, u', v'\}$ will induce one of H_4, H_5, H_6, or H_7 in Figure 7. This contradicts X being a threshold-modulator. □

Lemma 4.7. *Let G be a graph and X a corresponding threshold-modulator, then*

$$|\{N_X(v) \text{ for } v \in V(G) \setminus X\}| \leq |X| + 1.$$

Or in other words, there are at most $|X| + 1$ sets of X that are being realized.

Proof. Let u and v be two vertices in $G - X$. It follows directly from Lemma 4.6 that either $N_X(v) \subseteq N_X(u)$ or $N_X(v) \supseteq N_X(u)$. The result follows immediately. □

With the definition of the modulator and the basic properties above, we are now ready to extract more vertices from the instance, aiming at many consecutive levels that have the same neighborhood in X for the clique, and independent set vertices, respectively. This will lead up to our irrelevant vertex rule.

Let G be a graph, X a threshold-modulator and $(\mathcal{C}, \mathcal{I})$ a threshold partition of $G - X$. Recall Definition 4.2 on realizing subsets. Letting P denote either C or I, we say that a subset $Y \subseteq X$ has its *upper extreme* in P_i if P_i realizes Y and for every $j > i$ it holds that P_j does not realize Y. Similarly, a subset $Y \subseteq X$ has its *lower extreme* in P_i if P_i realizes Y and for every $j < i$ it holds that P_j does not realize Y. We say that $Y \subseteq X$ is *extremal* in P_i if Y has its upper or lower extreme in P_i. Observe that every $Y \subseteq X$ is extremal in at most two clique fragments and two independent set fragments. We continue having P denote either C or I.

Lemma 4.8. *Let G be a graph, X a threshold-modulator and $(\mathcal{C}, \mathcal{I})$ a threshold partition of $G - X$. Let P be either C or I. For every $Y \subseteq X$ it holds that if Y has its lower extreme in P_ℓ and upper extreme in P_u, then for every vertex $v \in P_i$ with $i \in [\ell + 1, u - 1]$ it holds that $N_X(v) = Y$.*

Proof. Let Y be a subset of X with P_ℓ and P_u being its lower and upper extremes. By definition there is a vertex $u \in P_\ell$ and a vertex $w \in P_u$ such that $N_X(u) = N_X(w) = Y$. Let i be an integer in $[\ell + 1, u - 1]$ and v a vertex in P_i. By the definition of a threshold partition it holds that $N_{G-X}(w) \subset N_{G-X}(v) \subset N_{G-X}(u)$. It follows from Lemma 4.6 that $N(w) \subset N[v]$ and that $N(v) \subset N[u]$. Hence,

$$Y = N_X(w) \subseteq N_X(v) \subseteq N_X(u) = Y$$

and we conclude that $N_X(v) = Y$. Since i and v was arbitrary, the proof is complete. □

Definition 4.9 (Important, Outlying, and Regular)**.** We say that P_i in the partition is *important* if there is a $Y \subseteq X$ such that Y has its extreme in P_i. Furthermore, a level L_i is important if C_i or I_i is important. Let f be the smallest number such that $|\bigcup_{i \leq f} C_i| \geq 2k + 2$ and r the largest number such that $|\bigcup_{i \geq r} I_i| \geq 2k + 2$. A level L_i is *outlying* if $i \leq f$ or $i \geq r$. All other levels of the decomposition are *regular* and a vertex is regular, outlying or important depending on the type of the level it is contained in.

Lemma 4.10. *Let G be a graph and X a threshold-modulator of G of size at most $4k$. Then every threshold partition of $G - X$ has at most $16k + 4$ important levels.*

Proof. The result follows from the definition of important levels and Lemma 4.7. □

Lemma 4.11. *Let G be a graph, X a threshold-modulator of G, and $(\mathcal{C}, \mathcal{I})$ a threshold partition of $G - X$, then for every set $Y \subseteq X$ there are at most two important clique fragments (independent fragments) realizing Y.*

Proof. We first prove the statement for clique fragments. Let Y be a subset of X and $i < j < k$ three integers. Assume for a contradiction that C_i, C_j, and C_k are important clique fragments all realizing Y. By definition there are vertices $u \in C_i$, $v \in C_j$ and $w \in C_k$ such that $N_X(u) = N_X(v) = N_X(w) = Y$. Furthermore, there is a vertex $v' \in C_j$ such that $N_X(v') \neq Y$ since C_j is important and Y does not have an extreme in C_j. By the definition of threshold partitions, we have that $N_{G-X}(w) \subset N_{G-X}(v') \subset N_{G-X}(u)$. Lemma 4.6 immediately implies that $N(w) \subset N[v']$ and $N(v') \subset N[u]$ and since $\{u, v', w\} \subseteq \cup\mathcal{C}$ it holds that

$N[u] \subseteq N[v'] \subseteq N[w]$. Since $N_X(v') \neq Y$, we have $N_X(w) \subset N_X(v') \subset N_X(u)$, which contradicts the definition of w and u since $N_X(u) = N_X(w)$. By a symmetric argument, the statement also holds for independent fragments. $\square$

Lemma 4.12. *Let G be a graph, X a threshold-modulator of G of size at most $4k$ and $(\mathcal{C}, \mathcal{I})$ a threshold partition of $G - X$. Then there are at most $64k^2 + 80k + 16$ important vertices in $G - X$.*

Proof. Let Y be the set of all vertices contained in an important clique or independent fragment and let Z be the set of all important vertices. Observe that $Y \subseteq Z$ and that every clique (resp. independent) fragment contained in $Z \setminus Y$ is a twin class in G by definition of important vertices and levels (Definitions 4.9 and 2.3). By Lemma 4.10 there are at most $16k + 4$ important levels and since the twin-rule has been applied exhaustively it holds that $|Z \setminus Y| \leq (16k+4)(2k+2) = 32k^2 + 40k + 8$.

Let A be a subset of X and B the vertices in Y such that their neighborhood in X is exactly A. Let P be a C_i or I_i contained in Y and observe that $P \cap B$ is a twin class in G and hence $|P \cap B| \leq 2k + 2$. And hence it follows from Lemma 4.11 that $|B| \leq 8k + 8$. Furthermore, we know from Lemma 4.7 that there are at most $4k + 1$ sets realized in X and hence $|Y| \leq (8k + 8)(4k + 1) = 32k^2 + 40k + 8$. It follows immediately that $|Z| \leq 64k^2 + 80k + 16$, completing the proof. $\square$

Lemma 4.13. *Let G be a graph, X a threshold-modulator of G of size at most $4k$ and $(\mathcal{C}, \mathcal{I})$ a threshold partition of $G - X$. Then there are at most $80k^2 + 112k + 32$ important and outlying vertices in total in $G - X$.*

Proof. By Lemma 4.12 it follows that there are at most $64k^2 + 80k + 16$ vertices that are important and possibly outlying. It follows from Lemma 4.8 that if a level is not important its vertices are covered by at most two twin classes in G and hence the level contains at most $4k + 4$ vertices. By definition there are at most $4k + 4$ outlying levels and hence at most $(4k + 4)(4k + 4) = 16k^2 + 32k + 16$ vertices which are outlying, but not important. The result follows immediately. $\square$

Lemma 4.14. *Let G be a graph, X a threshold-modulator of G, v a regular vertex in some threshold partition $(\mathcal{C}, \mathcal{I})$ of $G - X$, $C = \cup\mathcal{C}$ and $I = \cup\mathcal{I}$. Then for every $F \subseteq [V(G)]^2$ such that $G \oplus F$ is a threshold graph, $|F| \leq k$ and every split partition (C_F, I_F) of $G \oplus F$ we have:*

- *$v \in C$ if and only if $v \in C_F$ and*
- *$v \in I$ if and only if $v \in I_F$.*

Proof. Observe that the two statements are equivalent and that it is sufficient to prove the forward direction of both statements. Let $F \subseteq [V(G)]^2$ such that $G \oplus F$ is a threshold graph and $|F| \leq k$ and let (C_F, I_F) be the split partition certifying that $G \oplus F$ is a threshold graph. First, we prove that $v \in C$ implies that $v \in C_F$. Let Y be the set of outlying vertices in $I \cap N_G(v)$ and recall that $|Y| > 2k + 1$ by the definition (Definition 4.9). Observe that at most $2k$ vertices in Y are incident to F and hence there are two vertices u, u' in Y that are untouched by F and $u \in Y \subseteq N_G(v)$. Clearly, u and u' are not adjacent in $G \oplus F$ and hence we can assume without loss of generality that u is in I_F. Since u is untouched by F, v is adjacent to u by the definition of outlying vertices and hence v is not in I_F. A symmetric argument gives that $v \in I$ implies that $v \in I_F$ and hence our argument is complete. $\square$

4.1.5. The Irrelevant Vertex Rule

We have now obtained the structure necessary to give our irrelevant vertex rule. But before stating the rule, we need to define these consecutive levels with similar neighborhood and what it means for a vertex to be in the middle of such a collection of levels.

Definition 4.15 (Large strips, central vertices)**.** Let G be a graph, X a threshold-modulator and $(\mathcal{C}, \mathcal{I})$ a threshold partition of $G - X$. A *strip* is a maximal set of consecutive levels which are all regular and we say that a strip is *large* if it contains at least $16k + 13$ vertices. For a strip $S = ([C_a, I_a], \ldots, [C_b, I_b])$ with $a < b$, a vertex $v \in C_i$ is *central* if $a \leq i \leq b$ and $|\bigcup_{j \in [a,i-1]} C_j| \geq 2k + 2$ and $|\bigcup_{j \in [i+1,b]} C_j| \geq 2k + 2$. Similarly we say that a vertex $v \in I_i$ is *central* if $a \leq i \leq b$ and $|\bigcup_{j \in [a,i-1]} I_j| \geq 2k + 2$ and $|\bigcup_{j \in [i+1,b]} I_j| \geq 2k + 2$. Furthermore, we say that a vertex v is *central in G* if there exists a threshold-modulator X of size at most $4k$ and a threshold decomposition of $G - X$ such that v is central in a large strip.

Lemma 4.16. *If a strip is large it has a central vertex.*

Proof. Let $S = ([C_a, I_a], \ldots, [C_b, I_b])$ be a large strip. First, we consider the case when $|\bigcup_{i \in [a,b]} C_i| \geq |\bigcup_{i \in [a,b]} I_i|$. Observe that $|\bigcup_{i \in [a,b]} C_i| \geq 8k + 7$. Let i be the smallest number such that $|\bigcup_{j \in [a,i-1]} C_j| \geq 2k + 2$. It follows immediately from $|C_{i-1}| \leq 2k + 2$ that $|\bigcup_{j \in [a,i-1]} C_j| \leq 4k + 3$. Furthermore, since $|C_i| \leq 2k + 2$ it follows that $|\bigcup_{j \in [i+1,b]} C_j| \geq 8k+7-(2k+2+4k+3) = 2k+2$. And hence any vertex in C_i is central. A symmetric argument for the case $|\bigcup_{i \in [a,b]} C_i| < |\bigcup_{i \in [a,b]} I_i|$ completes the proof. $\square$

Rule 2 (Irrelevant vertex rule)**.** *If (G, k) be an instance of* Threshold Completion *or* Threshold Editing *and v is a central vertex in G, reduce to $(G - v, k)$.*

Lemma 4.17. *Let (G, k) be an instance, X a threshold-modulator of size at most $4k$, and v a central vertex in G. Then (G, k) is a yes-instance of* Threshold Editing *(*Threshold Completion*) if and only if $(G - v, k)$ is a yes-instance.*

Proof. For readability we only consider Threshold Editing, however the exact same proof works for Threshold Completion. For the forwards direction, for any vertex v, if (G, k) is a yes-instance, then $(G - v, k)$ is also a yes-instance. This holds since threshold graphs are hereditary.

For the reverse direction, let $(G - v, k)$ be a yes-instance and assume for a contradiction that (G, k) is a no-instance. Let F be a solution of $(G - v, k)$ satisfying Lemma 2.5, and let $G' = G \oplus F$. By assumption, (G, k) is a no-instance, so specifically, G' is not a threshold graph. Let W be an obstruction in G'. Clearly $v \in W$ since otherwise there is an obstruction in $(G - v) \oplus F$, so let $Z = V(W) - v$.

For convenience we will use N' to denote neighborhoods in G' and specifically for any set $Y \subseteq V(G')$, $N'_Y(v) = N_{G'}(v) \cap Y$. Furthermore, let $(\mathcal{C}, \mathcal{I})$ be a threshold decomposition of $G - X$ such that there is a large strip S for which v is central. We will now consider the case when v is in the clique of $G - X$. Since $|F| \leq k$ and S is a large strip it follows immediately that there are two clique vertices w and w' in S in higher levels than v that is not incident to F. Observe that $\{w, w', v\}$ forms a triangle and that W contains no such subgraph. Hence, we can assume without loss of generality that $w \notin V(W)$. Similarly, we obtain a clique vertex u in a lower level than v in S such that $u \notin W$.

Observe that $G'[Z \cup \{u\}]$ is not an obstruction and hence $N_Z(u) = N'_Z(u) \neq N'_Z(v) = N_Z(v)$. Since u and v are clique vertices from the same strip it is true that $N_X(v) = N_X(u)$ (see Lemma 4.8) and hence there is an independent vertex a in Z such that $\text{lev}(u) \leq \text{lev}(a) < \text{lev}(v)$ (see Definition 2.3). In other words u is adjacent to a while v and w are not. By a symmetric argument we obtain a vertex b such that $\text{lev}(v) \leq \text{lev}(b) < \text{lev}(w)$, meaning that both u and v are adjacent to b while w is not. Let y be last vertex of Z, meaning that $\{v, y, a, b\} = V(W)$. Observe that a and b are regular vertices and hence it follows from Lemma 4.14 that for every threshold partition of G' it holds that $\{a, b\}$ are independent vertices.

Recall that u, v, w, a, b are all regular and hence they are in the same partitions in G' as in $G - X$ by Lemma 4.14. Furthermore, since W is an obstruction and a is neither adjacent to v nor b in G' it holds that y and a are adjacent in G'. It follows that y is a clique vertex in G' and hence it is adjacent to both u and w in G'. Since u and w are not incident to F by definition, they are adjacent to y also in G. Since u, v, w are regular and from the same strip it follows that v is adjacent to y in both G and G'. Observe that the only possible adjacency not yet decided in W is the one between b and y. However, for W to be an obstruction it should not be present. Hence y is adjacent to a but not to b in G'. By definition $N_G(a) \subseteq N_G(b)$, however by the last observation this is not true in G'. This contradicts that F satisfies Lemma 2.5. A symmetric argument gives a contradiction for the case when v is an independent vertex and hence the proof is complete. $\square$

The above lemma shows the soundness of the irrelevant vertex rule, Rule 2, and we may therefor apply it exhaustively. The following theorem wraps up the goal of this section.

Theorem 6. Threshold Deletion, Threshold Completion *and* Threshold Editing *admit kernels with at most* $336k^2 + 388k + 92$ *vertices* .

Proof. Assume that Rules 1 and 2 have been applied exhaustively. If this process does not produce a threshold-modulator, we can safely output a trivial no-instance by Lemma 4.5. Hence, we can assume that we have a threshold-modulator X of size at most $4k$ and that the reduction rules cannot be applied. By Lemma 4.13 we know that there are at most $80k^2 + 112k + 32$ vertices in $G - X$ that are not regular. Furthermore, every regular vertex is contained in a strip and by Lemma 4.10 there are at most $16k + 5$ such strips. Since the reduction rules cannot be applied, no strip is large, and hence they contain at most $16k + 12$ vertices each. Since every

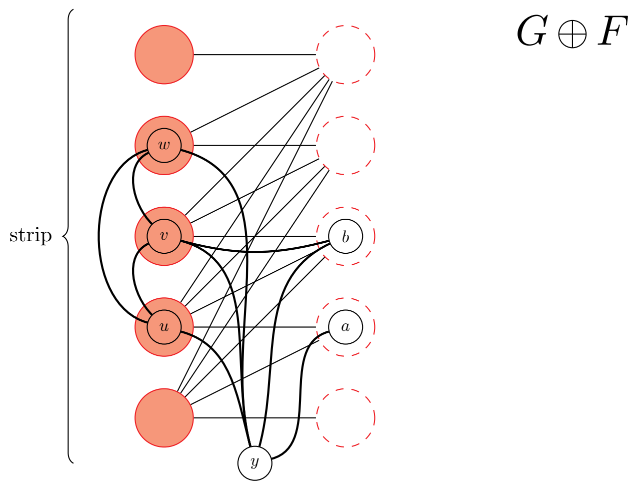

Figure 8: The vertex v was a center vertex in a strip and $W = \{v, a, b, y\}$ was assumed to be an obstruction.

vertex in G is either in X, or considered regular, outlying or important this gives us $4k + 80k^2 + 112k + 32 + (16k + 5)(16k + 12) = 336k^2 + 388k + 92$ vertices in total. □

4.2. Adapting the Kernel to Modification to Chain Graphs

In this section we provide kernels with quadratically many vertices for Chain Deletion, Chain Completion and Chain Editing. Due to the fundamental similarities between modification to chain and threshold graphs we omit the full proof and instead highlight the differences between the two proofs. Observe that the only proofs for the threshold kernels that explicitly applies the obstructions are those of Lemmata 4.5, 4.6 and 4.17 and hence these will receive most of our attention.

The twin reduction rule goes through immediately and hence our first obstacle is the modulator. Luckily, this is a minor one. Recall from Definition 4.1 that the obstructions now are $\mathcal{H} = \{2K_2, C_3, C_5\}$; We thus get a chain-modulator X of size $5k$, as the largest obstruction contains five vertices. Besides this detail, the proof goes through exactly as it is.

4.2.1. An Additional Step

Before we continue with the remainder of the proof we need an additional step. Namely to discard all vertices that are isolated in $G - X$. We will prove that by doing this we discard at most $O(k^2)$ vertices. Now, if the irrelevant vertex rule concludes that the graph is small, then the graph is small also when we reintroduce the discarded vertices. And if we find an irrelevant vertex, we remove it and

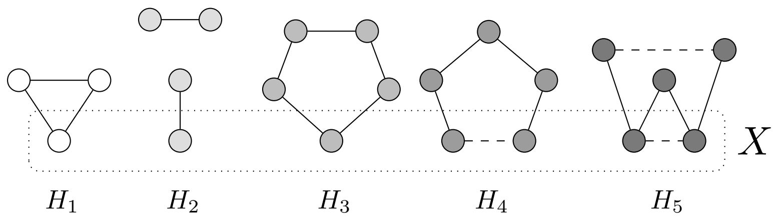

Figure 9: Some of the intersections of an obstruction with a chain-modulator X that by definition will not occur. Dashed edges represent edges that could or could not be there. These are the intersections necessary for the proof of the kernel.

reintroduce the discarded vertices before we once again apply our reduction rules. Due to the locality of our arguments, this is a valid approach.

Lemma 4.18. *For a graph G and a corresponding chain-modulator X of size at most $5k$ there are at most $20k^2 + 20k$ isolated vertices in $G - X$.*

Proof. Let I be the set of isolated vertices in $G - X$. We will prove that the set system $\mathcal{F} = \{N_X(v) \mid v \in I\}$ is laminar (see Definition 2.8) and hence by Lemma 2.9 it holds that $|\mathcal{F}| \leq 2|X| \leq 10k$. It follows immediately, due to the twin reduction rule (Rule 1, Lemma 4.3), that there are at most $10k \cdot (2k + 2) = 20k^2 + 20k$ independent vertices in $G - X$.

Assume for a contradiction that there are vertices u, v, and w in I such that there exists $u' \in N_X(u) \setminus N_X(v)$ and $v' \in N_X(v) \setminus N_X(u)$ with $\{u', v'\} \subseteq N_X(w)$. These vertices intersect with the modulator as a variant of the forbidden H_5 in Figure 9 and hence we get a contradiction. □

4.2.2. Nested Neighborhoods

From now on we will assume in all of our arguments that there are no isolated vertices in $G - X$. The next difference is with respect to Lemma 4.6, which is just not true anymore. The lemma provided us with the nested structure of the neighborhoods in the modulator and was crucial for most of the proofs. As harmful as this appears to be at first, it turns out that we can prove a weaker version that is sufficient for our needs.

Lemma 4.19 (New, weaker version of Lemma 4.6)**.** *Let G be a graph and X a chain-modulator. For every pair of vertices u and v in the* same bipartition *of $G - X$ it holds that either $N(u) \subseteq N(v)$ or $N(v) \subseteq N(u)$.*

Proof. Let u and v be two vertices from the same bipartition of $G - X$. By the definition of chain graphs we can assume that $N_{G-X}(u) \subseteq N_{G-X}(v)$. Assume for a contradiction that the lemma is not true. Then there is a vertex $u' \in N_X(u) \setminus N_X(v)$ and a vertex v' in $N_X(v) \setminus N_X(u)$. By definition, u and v are not adjacent. Since there are no isolated vertices in $G - X$ there is a vertex $a \in N_{G-X}(u) \subseteq N_{G-X}(v)$. Observe that if a is adjacent to either u' or v' we get a C_3 that only has one vertex in X, which is a contradiction (see H_1 in Figure 9). However, if a is not adjacent to both u' and v' then $\{u, v, u', v', a\}$ forms the same interaction with the modulator as H_4 in Figure 9 and hence our proof is complete. $\square$

One can observe that Lemma 4.19 is a sufficiently strong replacement for Lemma 4.6 since all proofs are applying the lemma to vertices from only one partition of $G - X$. The only exception is the proof of Lemma 4.7, but by applying Lemma 4.19 on one partition at the time we obtain the following bound instead:

$$|\{N_X(v) \text{ for } v \in V(G) \setminus X\}| \leq 2|X| + 2.$$

4.2.3. An Irrelevant Vertex Rule

It only remains to prove that the irrelevant vertex rule can still be applied with this new set of obstructions. Although the strategy is the same, the details are different and hence we provide the proof in full detail.

Lemma 4.20. *Let (G, k) be an instance, X a chain-modulator of size at most $5k$, and v a central vertex in G. Then (G, k) is a yes-instance of* Chain Editing *(*Chain Completion*) if and only if $(G - v, k)$ is a yes-instance.*

Proof. For readability we only consider Chain Editing, however the exact same proof works for Chain Completion. For the forwards direction, for any vertex v, if (G, k) is a yes-instance, then $(G - v, k)$ is also a yes-instance. This holds since chain graphs are hereditary.

For the reverse direction, let $(G - v, k)$ be a yes-instance and assume for a contradiction that (G, k) is a no-instance. Let F be a solution of $(G - v, k)$ satisfying Lemma 2.5, and let $G' = G \oplus F$. By assumption, (G, k) is a no-instance, so specifically, G' is not a chain graph. Let W be an obstruction in G'. Clearly $v \in W$, since otherwise there is an obstruction in $(G - v) \oplus F$. Let $Z = V(W) - v$. For convenience we will use N' to denote neighborhoods in G' and specifically for any set $Y \subseteq V(G')$, $N'_Y(v) = N_{G'}(v) \cap Y$. Furthermore, let $(\mathcal{A}, \mathcal{B})$ be a chain decomposition of $G - X$ such that there is a large strip S for which v is central. Let $A = \cup\mathcal{A}$ and $B = \cup\mathcal{B}$. We will now consider the case when v is in A. Since $|F| \leq k$ and S is a large strip it follows immediately that there are two vertices w and w' in $A \cap S$ in higher levels than v that is not incident to F. Observe that $\{w, w', v\}$ forms an independent set of size three and that W contains no such subgraph. Hence, we can assume without loss of generality that $w \notin V(W)$. Similarly, we obtain a vertex u in A at a lower level than v in S such that $u \notin W$.

Observe that $G'[Z \cup \{u\}]$ is not an obstruction and hence $N_Z(u) = N'_Z(u) \neq N'_Z(v) = N_Z(v)$. Since u and v are vertices in A from the same strip it is true that $N_X(v) = N_X(u)$ and hence there is a vertex a in $Z \cap B$ such that $\text{lev}(u) \leq \text{lev}(a) < \text{lev}(v)$. In other words u is adjacent to a, while v and w are not. By a symmetric argument we obtain a vertex b such that $\text{lev}(v) \leq \text{lev}(b) < \text{lev}(w)$, meaning that both u and v are adjacent to b while w is not. We now fix a chain decomposition $(\mathcal{A}', \mathcal{B}')$ and let $A' = \cup\mathcal{A}'$ and $B' = \cup\mathcal{B}'$. Observe that a and b are regular vertices and hence it follows from the chain version of Lemma 4.14 that $\{a, b\}$ is in B'. This yields immediately that W is not a C_3 (since a and b are not adjacent) and hence we are left the cases of W being a $2K_2$ or a C_5.

We now consider the case when W is isomorphic to a $2K_2$. Let y be the last vertex of Z, meaning that $\{v, y, a, b\} = V(W)$. Observe that since W is a $2K_2$ it holds that y is adjacent to a, but not to b. However, in G it holds that $N(a) \subseteq N(b)$ and hence F is not satisfying Lemma 2.5, which is a contradiction.

Hence we are left with the case that W is isomorphic to a C_5. Let y, x be the last vertices of Z. Observe that all vertices in W should be of degree two and hence a is adjacent to both x and y. Recall that a is in B' and observe that u is in A' by the same reasoning. Due to their adjacency to a, also x and y is in A'. It follows immediately that u, x and y form an independent set in $(G-v) \oplus F$. Since u and v are not touched by F and in the same strip it follows that v, x and y form an independent set in G'. We observe that by this W can not be isomorphic to a C_5. The argument for the case when $v \in B$ is symmetrical and hence the proof is complete. $\square$

We immediately obtain our kernelization results for modifications into chain graphs by the same wrap up as for threshold graphs.

Theorem 7. *The following three problems admit kernels with at most $O(k^2)$ vertices:* Chain Deletion, Chain Completion *and* Chain Editing.

5. Subexponential Time Algorithms

5.1. Threshold Editing in Subexponential Time

In this section we give a subexponential time algorithm for Threshold Editing. We also show that we can modify the algorithm to work with Chain Editing. Combined with the results of Fomin and Villanger [15] and Drange et al. [11], we now have complete information on the subexponentiality of edge modification to threshold and chain graphs. In this section we aim to prove the following theorem:

Theorem 8. Threshold Editing *admits a $2^{O(\sqrt{k}\log k)} + \mathrm{poly}(n)$ subexponential time algorithm.*

The additive $\mathrm{poly}(n)$ factor comes from the kernelization procedure of Section 4. The remainder of the algorithm operates on the kernel, and thus has running time that only depends on k.

We will throughout refer to a *solution* F. In this case, we are assuming a given input instance (G, k), and then F is a set of at most k edges such that $G \oplus F$ is a threshold graph. In the next section, Section 5.2, we will assume $G \oplus F$ to be a chain graph. Furthermore, after Section 5.1.2, we will be working with the problem Split Threshold Editing, so we assume $F \subseteq C \times I$ when (C, I) is the split partition of G.

Definition 5.1 (Potential split partition)**.** Given a graph G and an integer k (called the budget), for C and I a partitioning of $V(G)$ we call (C, I) a *potential split partition* of G provided that

$$\binom{|C|}{2} - E(C) + E(I) \leq k.$$

That is, the cost of making G into a split graph with the prescribed partitioning does not exceed the budget.

A brief explanation of the algorithm for Theorem 8.. The algorithm consists of four parts, the first of which is the kernelization algorithm described in Section 4. This gives in polynomial time an equivalent instance (G, k) with the guarantee that $|V(G)| = O(k^2)$. We may observe that this is a *proper kernel*, i.e., the reduced instance's parameter is bounded by the original parameter. This allows us to use time subexponential in the kernelized parameter.

The second step in the algorithm selects a potential split partitioning of G. We show that the number of such partitionings is bounded subexponentially in k, and that we can enumerate them all in subexponential time. This step actually also immediately implies that editing[2], completing and deleting to split graphs can be solved in subexponential time, however all of this was known [19, 16]. The main part of this step is Lemma 5.4. For the remainder of the algorithm, we may thus assume that the input instance is a split graph, and that the split partition needs to be preserved, that is, we focus on solving Split Threshold Editing.

The third and fourth steps of the algorithm consists of repeatedly finding special kind of separators and solving structured parts individually; Step three consists of locating so-called *cheap vertices* (see Definition 5.6 for a formal explanation). These are vertices, v, whose neighborhood is almost correct, in the sense that there is an optimal solution in which v is incident to only $O(\sqrt{k})$ edges. The dichotomy of cheap and expensive vertices gives us some tools for decomposing the graph. Specific configurations of cheap vertices allow us to extract three parts, one part is a highly structured part, the second part is a provably small part which we may brute force, and the last part we solve recursively. All of which is done in subexponential time $2^{O(\sqrt{k}\log k)}$.

Henceforth we will have in mind a "target graph" $H = G \oplus F$ with threshold partitioning $(\mathcal{C}, \mathcal{I})$. We refer to the set of edges F as the *solution*, and assume $|F| \leq k$.

5.1.1. Enumerating small sets

A crucial part of the algorithm is to enumerate all vertex sets of size at most $O(\sqrt{k})$. The following observation shows that this is indeed doable and we will use this result throughout this section without necessarily referring to it.

Observation 5.2. *For every $c \in \mathbb{N}$ there is an algorithm that, given an input instance (G, k) with $|V(G)| = k^{O(1)}$ enumerates all vertex subsets of size $c\sqrt{k}$ in time $2^{O(\sqrt{k}\log k)}$.*

Proof. Given an input graph $G = (V, E)$, with $|V| = n = k^{O(1)}$ and a natural number k we can simply output the family of sets $\mathcal{X} \subseteq 2^V$ of size at most $c\sqrt{k}$,

[2]Indeed, editing to split graphs is solvable in linear time [19].

which takes time

$$\sum_{\kappa \leq c\sqrt{k}} \binom{n}{\kappa} \leq c\sqrt{k}\binom{n}{c\sqrt{k}} \leq c\sqrt{k} \cdot n^{c\sqrt{k}} = 2^{O(\sqrt{k}\log n)} = 2^{O(\sqrt{k}\log k)},$$

where the first inequality follows since $\binom{n}{i}$ is increasing for i from 1 to $c\sqrt{k}$. $\square$

5.1.2. Enumerating the potential partitions

After kernelizing the instance the next step of the subexponential time algorithm is to compute the potential split partitionings of the input instance. Since we are given a general graph, we do not know which vertices will go to the clique and which will go to the independent set. However, we now show that there is at most subexponentially many potential split partitionings. That is, there are subexponentially many partitionings of the vertex set into (C, I) such that it is possible to edit the input graph to a threshold graph with the given partitioning while not exceeding the prescribed budget.

Lemma 5.4 will be crucial in our algorithm, as our algorithm presupposes a fixed split partition. Using this result, we may in subexponential time compute every possible split partition within range, and run our algorithm for editing to threshold graphs on each of these split graphs.

But first, we show a small observation.

Lemma 5.3. *Let G be a split graph with split partition (C, I), and let $C' \subseteq C$ and $I' \subseteq I$ be two vertex sets with $|C'| = |I'| = s \geq 2\sqrt{k}$ for some $k \geq 2$. It then holds that $\sum_{v \in C'} \deg(v) > \sum_{v \in I'} \deg(v) + 2k$.*

Proof. Since the edges going from C' to I' contributes the same amount to each side of the inequality, we will without loss of generality assume that there are no edges between C' and I'. Now we count the degrees of each of the sets. Each vertex $v \in C'$ has degree at least $|C| - 1$ since C is a clique, and each vertex $v \in I'$ has degree at most $|C| - |C'| = |C| - s$. The latter follows from the fact that vertices in I' only has edges to C, but by the assumption above does not have any edges to C'. It follows that

$$\sum_{v \in C'} \deg(v) \geq s(|C| - 1) = s|C| - s, \text{ and}$$
$$\sum_{v \in I'} \deg(v) \leq s(|C| - s) = s|C| - s^2.$$

Hence, the lemma holds if and only if $-s > -s^2 + 2k$, that is, if $s^2 - 2k > s$. But this is indeed the case for every $s \geq 2\sqrt{k}$ as long as $k \geq 2$. $\square$

Lemma 5.4. *Given a graph $G = (V, E_G)$ and an integer k with $|V| = k^{O(1)}$ one can in $2^{O(\sqrt{k}\log k)}$ time generate a set $\mathcal{P}$ of partitions of V such that for every graph H such that*

- *H is a split graph and*
- $|E(H) \oplus E(G)| \leq k$

it holds that every split partition (C, I) of H is an element of $\mathcal{P}$.

Proof. Let $G = (V, E)$ be a graph and k a natural number. The first thing we do is to guess the size s_c of the clique and $s_i = n - s_c$. In the case that $\min\{s_c, s_i\} \leq 2\sqrt{k}$ we can simply enumerate every partitioning by Observation 5.2, so we assume from now on that $\min\{s_c, s_i\} > 2\sqrt{k}$.

We now claim that by ordering the vertices in G from high degree to low degree, and referring to the s_c highest degree vertices as $\hat{C}$ and the rest as $\hat{I}$, any partitioning $(C, I) \in \mathcal{P}$ where $|C| = s_c$ and $|I| = s_i$, $|\hat{C} \cap I| = |\hat{I} \cap C| \leq 2\sqrt{k}$. This follows immediately from Lemma 5.3; Let $\ell > 2\sqrt{k}$ be the number of vertices from $\hat{C}$ that appears in some I of size s_i, and vice versa. Since the vertices of $\hat{C}$ have higher degrees than the vertices of $\hat{I}$, it follows that $\sum_{v \in \hat{C}} \deg(v) \geq \sum_{v \in \hat{I}} \deg(v)$. But in any such split graph H, $\sum_{v \in \hat{I}} \deg(v) > \sum_{v \in \hat{C}} \deg(v) + 2k$, which contradicts the fact that we can only edit k edges. $\square$

We would like to remark that this lemma also gives a simpler algorithm for Split Completion (equivalently Split Deletion). Ghosh et al. [16] showed that Split Completion can be solved in time $2^{O(\sqrt{k} \log k)} \cdot \text{poly}(n)$ using the framework of Alon, Lokshtanov and Saurabh [1]. However, the following observation immediately yields a very simple combinatorial argument for the existence of such an algorithm. Together with the polynomial kernel by Guo [18], the following result is immediate from the above lemma.

Corollary 5.5. *The problem* Split Completion *is solvable in time* $2^{O(\sqrt{k} \log k)} + \text{poly}(n)$.

Proof. The algorithm is as follows. On a kernelized [18] input (G, k) we compute, using Lemma 5.4, every potential split partitioning (C, I) at most k edges away from G. Then we in linear time check that I is indeed independent and that C lacks at most k edges from being complete. $\square$

5.1.3. Cheap or Expensive?

We will from now on assume that all our input graphs $G = (V, E)$ are split graphs provided with a split partition (C, I), and that we are to solve Split Threshold Editing, that is, we have to respect the split partitioning. We are allowed to do this with subexponential time overhead, as per the previous section and specifically Lemma 5.4. In addition, we assume that $|V(G)| = O(k^2)$.

Given an instance (G, k) and a solution F, we define the *editing number* of a vertex v, denoted $\text{en}_G^F(v)$, to be the number of edges in F incident to a vertex v. When G and F are clear from the context, we will simply write $\text{en}(v)$. A vertex v will be referred to as *cheap* if $\text{en}(v) \leq 2\sqrt{k}$ and *expensive* otherwise. We will call a set of vertices $U \subseteq V$ *small* provided that $|U| \leq 2\sqrt{k}$ and *large* otherwise.

Definition 5.6. Given an instance (G,k) with solution F, we call a vertex v *cheap* if $\mathrm{en}(v) \leq 2\sqrt{k}$.

The following observation will be used extensively.

Observation 5.7. *If $U \subseteq V(G)$ is a large set, then there exists a cheap vertex in U, or contrapositively: if a set $U \subseteq V(G)$ has only expensive vertices, then U is small. Specifically it follows that in any yes-instance (G,k) where F is a solution, there are at most $2\sqrt{k}$ expensive vertices.*

This gives the following win-win situation: If a set X is small, then we can "guess" it, which means that we can in subexponential time enumerate all candidates, and otherwise, we can guess a cheap vertex inside the set and its "correct" neighborhood. In particular, since the set of expensive vertices is small, we can guess it in the beginning. For the remainder of the proof we will assume that the graph G is a labeled graph, where some vertices are labeled as cheap and others as expensive. There will never be more than $2\sqrt{k}$ vertices labeled expensive. The idea is that we guess the expensive vertices at the start of the algorithm and then bring this information along when we recurse on subgraphs.

From now on, we assume that we are solving Split Threshold Editing on a graph with at most $O(k^2)$ vertices and a set of at most $2\sqrt{k}$ vertices are labeled expensive.

5.1.4. Splitting Pairs and Unbreakable Segments

Definition 5.8 (Splitting pair)**.** Let G be a graph, k an integer, F a solution of (G,k), and $(\mathcal{C},\mathcal{I})$ a threshold decomposition of $G \oplus F$. We then say that the vertices $u \in I_a$ and $v \in C_b$ is a *splitting pair* if

- $a \leq b$,
- u and v are cheap,
- $\bigcup_{a<i<b} L_i$ consists of only expensive vertices. Recall from Definition 2.3 that $L_i = C_i \cup I_i$.

Definition 5.9 (Unbreakable)**.** Let G be a graph, k an integer, F a solution of (G,k), $(\mathcal{C},\mathcal{I})$ a threshold decomposition of $G \oplus F$, and $a \leq b$. We then say that a sequence of levels $(C_a, I_a), (C_{a+1}, I_{a+1}), \ldots, (C_b, I_b)$ is an *unbreakable segment* if there is no splitting pair in the vertex set $\bigcup_{i\in[a,b]}(C_i \cup I_i)$. Furthermore, we say that an instance (G,k) is *unbreakable* if there exists an optimal solution F and a threshold decomposition $(\mathcal{C},\mathcal{I})$ of $G \oplus F$ such that the entire decomposition is an unbreakable segment. We also say that such a decomposition is a *witness* of G being unbreakable.

Definition 5.10. Let G be a graph and $(\mathcal{C},\mathcal{I})$ a threshold decomposition of $G \oplus F$ for some solution F. Then we say that i is a *transfer level* if

- for every $j > i$ it holds that C_j contains no cheap vertices and
- for every $j < i$ it holds that I_j contains no cheap vertices.

Lemma 5.11. *Let (G, k) be a yes-instance of* Split Threshold Editing *with solution F such that G is unbreakable and $(\mathcal{C}, \mathcal{I})$ a witness. Then there is a transfer level in $(\mathcal{C}, \mathcal{I})$.*

Proof. Suppose towards a contradiction that G is unbreakable and $(\mathcal{C}, \mathcal{I})$ a witness, and that there is no transfer level. Let a be maximal such that C_a contains a cheap vertex and b minimum such that I_b contains a cheap vertex. Since $i = a$ clearly satisfies the first condition, it must be the case that $b < a$, for otherwise we would have a splitting pair which contradicts the assumption that the graph is unbreakable. Increment b as long as $b + 1 < a$ and there is a cheap vertex in $\bigcup_{i \in (b,a)} I_i$. Then decrement a as long as $b+1 < a$ and there is a cheap vertex in $\bigcup_{i \in (b,a)} C_i$. Let u be a cheap vertex in C_a and v a cheap vertex in C_b. It follows from the procedure that they both exist. Observe that u, v is indeed a splitting pair, which is a contradiction to G being unbreakable and $(\mathcal{C}, \mathcal{I})$ being a witness. $\square$

Lemma 5.12. *Let (G, k) be an instance of* Split Threshold Editing *such that G is unbreakable and $(\mathcal{C}, \mathcal{I})$ a witness of this. Then the number of levels in $(\mathcal{C}, \mathcal{I})$ is at most $2\sqrt{k} + 1$.*

Proof. Let i be the transfer level in $(\mathcal{C}, \mathcal{I})$. It is guaranteed to exist by Lemma 5.11. Observe that for every $j > i$ it holds that C_j consists of expensive vertices and for every $j < i$ it holds that I_j consists of expensive vertices. It follows from Definition 2.3 that every level besides i contains at least one expensive vertex. As there are at most $2\sqrt{k}$ such vertices the result follows. $\square$

Lemma 5.13. *Let (G, k) be an instance of* Split Threshold Editing *such that G is unbreakable, $(\mathcal{C}, \mathcal{I})$ is a witness of this and F a corresponding solution. If X is the set of cheap vertices in G then $(G \oplus F)[X]$ forms a complete split graph.*

Proof. Let t be the transfer level of the decomposition, u a cheap vertex in C_i and v a cheap vertex in I_j for some i and j. By the definition of t it holds that $i \leq t \leq j$. It follows immediately that u and v are adjacent in $G \oplus F$ and the proof is complete. $\square$

We will now describe the algorithm `unbreakAlg`, which takes as input an instance $(G, (C, I), k)$ of Split Threshold Editing, with the assumption that G is unbreakable and has split partition (C, I), and returns either an optimal solution F for (G, k) where $|F| \leq k$, or correctly concludes that (G, k) is a no-instance. Assume that (G, k) is a yes-instance. Then there exists an optimal solution F and a threshold decomposition $(\mathcal{C}, \mathcal{I})$ of $G \oplus F$ that is a witness of G being unbreakable. First, we guess the number of levels ℓ in the decomposition, and by Lemma 5.12, we have that $\ell \in [0, 2\sqrt{k} + 1]$ and the transfer level $t \in [0, \ell]$. Then we guess where the at most $2\sqrt{k}$ vertices that are expensive in G are positioned in $(\mathcal{C}, \mathcal{I})$. Observe that from this information we can obtain all edges between expensive vertices in F. Finally, we put every cheap vertex in the level that minimizes the cost of fixing its adjacencies into the expensive vertices while respecting that t is the transfer level. From this information we can obtain all adjacencies between cheap and expensive vertices in F. Since the cheap vertices (by Lemma 5.13) induce a complete split graph, we reconstructed F and hence we return it.

Lemma 5.14. *Given an instance (G, k) of* Split Threshold Editing *with G being unbreakable,* `unbreakAlg` *either gives an optimal solution or correctly concludes that (G, k) is a no-instance in time $2^{O(\sqrt{k} \log k)}$.*

Proof. Since the algorithm goes through every possible value for ℓ and t (according to Lemmata 5.11 and 5.12), and every possible placement of the expensive vertices, the only thing remaining to ensure is that the cheap vertices are placed correctly. However, since the cheap vertices form a complete split graph (according to Lemma 5.13), the only cost associated with a cheap vertex is the number of expensive vertices in the opposite side it is adjacent to. However, their placement is fixed, so we simply greedily minimize the cost of the vertex by putting it in a level that minimizes the number of necessary edits.

If we get a solution from the above procedure, this solution is optimal. On the other hand, if in every branch of the algorithm we are forced to edit more than k edges, then either (G, k) is a no-instance, or G is not unbreakable. Since the assumption of the algorithm is that G is unbreakable, we conclude that the algorithm is correct. $\square$

5.1.5. Divide and Conquer

We now explain the main algorithm. The algorithm takes as input a graph G, together with a split partition (C, I) and a budget k. To assist the divide and conquer algorithm, we also take as input the current subset of vertices we are working on—a set S such that the algorithm should find an optimal solution for $G[S]$. Of course, we initialize the algorithm with $S = V(G)$. The algorithm is recursive and either finds a splitting pair, and recurses on a subset of S, and if there is no splitting pair, then $G[S]$ is unbreakable, and thus it simply runs `unbreakAlg` on $G[S]$. To avoid unnecessary recomputations, it uses memoization to solve already computed inputs.

The algorithm `solveAlg`$(G, (C, I), k, S)$ returns an optimal solution of the input instance $(G[S], k)$, respecting the given split partition (C, I) in the following manner:

(1) Run `unbreakAlg`$(G[S], (C \cap S, I \cap S), k)$.

(2) For every pair of cheap vertices $u \in I$ and $v \in C$, together with their correct neighborhoods N_u and N_v, and every pair of subsets $C_X \subseteq C$ and $I_X \subseteq I$ of expensive vertices we do the following: Let $X = I_X \cup C_X$, $R_C = N_u$, $U_I = N_v \cap I$, $R_I = I \setminus (X \cup U_I)$ and $U_C = S \setminus (X \cup R_C \cup U_I \cup R_I)$. Furthermore, let $U = U_I \cup U_C$ and $R = R_I \cup R_C$.

Now, in this guess we try to see whether U is an unbreakable segment, u and v a splitting pair, and X the set of of expensive vertices—R is simply the remaining vertices. We now

(a) run `unbreakAlg`$(G[U], (C \cap U, I \cap U), k)$ yielding a solution F_U,

(b) solve $G[X]$ optimally by brute force since it has size at most $2\sqrt{k}$, giving a solution F_X, and

(c) recursively call `solveAlg`$(G, (C, I), k, R)$ to solve the instance corresponding to the remaining vertices yielding F_R.

Finally we return F, the union of F_U, F_X, and F_R together with all edges from $C \cap R$ and $I \cap (X \cup U)$, and all edges from $C \cap X$ to $I \cap U$.

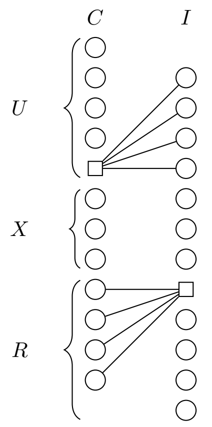

Figure 10: The partitioning of the vertex sets according to `solveAlg`. The square bags are the bags containing the splitting pair, U is an unbreakable segment and the bags of X contains exclusively expensive vertices. The edges drawn indicates the neighborhoods of the splitting pair across the partitions.

In *(1)* we consider the option that there are no splitting pairs in G. In *(2)* (see Figure 10) we guess the uppermost splitting pair in the partition and the neighborhood of these two vertices. Then we guess all of the expensive vertices that live in between the two levels of the splitting pair. Observe that these expensive vertices together with the splitting pair partition the levels into three consecutive sequences. The upper one, U is an unbreakable segment, the middle, X are the expensive vertices and the lower one, R is simply the remaining graph. When we apply `unbreakAlg` on the upper part, brute force the middle one and recurse with `solveAlg` on the lower part, we get individual optimal solutions for each three, finally we may merge the solutions and add all the remaining edges (see end of *(2)*).

Lemma 5.15. *Given a split graph $G = (V, E)$ with split partition (C, I),* `solveAlg` *either returns an optimal solution for* Split Threshold Editing *on input $(G, (C, I), k, V)$, or correctly concludes that (G, k) is a no-instance.*

Proof. If (G, k) with split partition (C, I) is a yes-instance of Split Threshold Editing there is a solution F with threshold decomposition $(\mathcal{C}, \mathcal{I})$ and a sequence of pairs $(u_1, v_1), (u_2, v_2), \ldots, (u_t, v_t)$ such that u_1, v_1 is the splitting pair highest in $(\mathcal{C}, \mathcal{I})$, and u_2, v_2 in the highest splitting pair in the graph induced by the vertices in and below the level of v_1, etc. Since we in a state $(G, (C, I), k, S)$ try every possible pair of such cheap vertices and every possible neighborhood and set of expensive vertices, we exhaust all possibilities for any threshold editing of S of at most k edges. Hence, if there is a solution, an optimal solution is returned.

Thus, if ever an F is constructed of size $|F| > k$, we can safely conclude that there is no editing set $F^\star \subseteq C \times I$ of size at most k such that $G \oplus F^\star$ is a threshold graph. □

Lemma 5.16. *Given a split graph $G = (V, E)$ with split partition (C, I) and an integer k with $|V(G)| = O(k^2)$, the algorithm* `solveAlg` *terminates in time $2^{O(\sqrt{k} \log k)}$ on input $(G, (C, I), k, V)$.*

Proof. By charging a set S for which `solveAlg` is called with input $(G, (C, I), k, S)$ every operation except the recursive call, we need to *(i)* show that there are at most $2^{O(\sqrt{k} \log k)}$ many sets $S \subseteq V$ for which `solveAlg` is called, and *(ii)* that the work done inside one such call is at most $2^{O(\sqrt{k} \log k)}$.

For Case *(i)*, we simply note that when `solveAlg` is called with a set S, the sets R on which we recurse are uniquely defined by u, v, N_u, N_v, X, and there are at most $O(k^4) \cdot 2^{O(\sqrt{k} \log k)^3} = 2^{O(\sqrt{k} \log k)}$ such configurations, so at most $2^{O(\sqrt{k} \log k)}$ sets are charged. Case *(ii)* follows from the fact that we guess two vertices, u and v and three sets, N_u, N_v and X. For each choice we run `unbreakAlg`, which runs in time $2^{O(\sqrt{k} \log k)}$ by Lemma 5.14, and the brute force solution takes time $2^{O(\sqrt{k} \log(\sqrt{k}))}$. The recursive call is charged to a smaller set, and merging the solutions into the final solution we return, F, takes polynomial time.

The two cases show that we charge at most $2^{O(\sqrt{k} \log k)}$ sets with at most $2^{O(\sqrt{k} \log k)}$ time, and hence `solveAlg` completes after $2^{O(\sqrt{k} \log k)}$ steps. □

To conclude we observe that Theorem 8 follows directly from the above exposition. Given an input (G, k) to Threshold Editing, from the previous section we can in polynomial time obtain an equivalent instance with at most $O(k^2)$ vertices. Furthermore, by Lemma 5.4 we may in time $2^{O(\sqrt{k} \log k)}$ time assume we are solving the problem Split Threshold Editing. Finally, by Lemmata 5.15 and 5.16, the theorem follows.

5.2. Editing to Chain Graphs

We finally describe which steps are needed to change the algorithm above into an algorithm correctly solving Chain Editing in subexponential time. The main difference between Chain Editing and Threshold Editing is that it is far from clear that the number of bipartitions is subexponential, that is, is there a bipartite equivalent of the bound of the potential split partitions as in Lemma 5.4? If we were able to enumerate all such "potential bipartitions" in subexponential time, we could simply run a very similar algorithm to the one above on the problem Bipartite Chain Editing, where we are asked to respect the bipartition (see Section 3.2.1 for the definition of this problem).

It turns out that we indeed are able to enumerate all such potential bipartitions within the allowed time:

Lemma 5.17. *There is an algorithm which, given an instance (G,k) for* Chain Editing*, enumerates $\binom{|V|}{O(\sqrt{k})} = 2^{O(\sqrt{k}\log|V|)}$ bipartite graphs $H=(A,B,E')$ with $|E \oplus E'| \leq k$ such that if (G,k) is a yes-instance, then one output (H,k) will be a yes-instance for* Bipartite Chain Editing*, and furthermore if any yes-instance (H,k) is output, then (G,k) is a yes-instance. This also holds for the deletion and completion versions.*

Proof. We first mention that it is trivial to change the below proof into the proofs for the deletion and completion versions; One simply disallow one of the operations. So we will prove only the editing version. Furthermore, it is clear to see that if any output instance (H,k) is a yes-instance for Bipartite Chain Editing, then (G,k) was a yes-instance for Chain Editing.

Consider any solution $H=(A,B,E')$ for an input instance (G,k). If $\min\{|A|,|B|\} \leq 5\sqrt{k}$, then we can simply guess every such in subexponential time. Hence, we assume that both sides of H are large. But this means, by Observation 5.7, that both A and B have cheap vertices. Let v_A be a cheap vertex as low as possible in A and v_B be a cheap vertex as high as possible in B. It immediately follows from the same observation that the set of vertices below v_A, A_X is a set of expensive vertices, and the same for the vertices above v_B, B_X. Since v_A and v_B, we know that we can in subexponential time correctly guess their neighborhoods in H and we can similarly guess A_X and B_X.

Now, since we know v_A, v_B, $N_H(v_A)$ and $N_H(v_B)$, as well as A_X and B_X, the only vertices we do now know where to place, are the vertices in A which are in the levels above $\text{lev}(v_B)$, call them A_Y, and the vertices in B which are in the levels below $\text{lev}(v_B)$, call them B_Y. However, we know which set this is, that is, we know $Z = A_Y \cup B_Y$. Define now $A_M = A \setminus (A_Y \cup A_X \cup \{v_A\})$ and similarly $B_M = B \setminus (B_Y \cup B_X \cup \{v_B\})$. These are the vertices living in the middle of A and B, respectively.

We now know that the vertices of Z should form an independent set. This follows from the fact that A_M and B_M are both non-empty. Hence, the vertices of A_Y are in higher levels than all of B_Y, and since there are no edges going from a vertex in A to a vertex lower in B, and each of A and B are independent sets, Z must be an independent set.

The following is the crucial last step. We can in subexponential time guess the partitioning of levels of both A_X and of B_X, since they are both of sizes at most $2\sqrt{k}$. When knowing these levels, we can greedily insert each vertex in Z into either A and B by pointwise minimizing the cost; A vertex $z \in Z$ can safely be places in the level of A or B which minimizes the cost of making it adjacent to only the vertices of B_X above its level, or by making it adjacent to only the vertices below its level in A_X. $\square$

Given the above lemma, we may work on the more restricted problem, Bipartite Chain Editing. The rest of the algorithm actually goes through without any noticeable changes:

Theorem 9. Chain Editing *is solvable in time* $2^{O(\sqrt{k}\log k)} + \text{poly}(n)$.

Proof. On input (G, k) we first run the kernelization algorithm from Section 4.2, and then we enumerate every potential bipartition according to Lemma 5.17. Now, for each bipartition (A, B) we make A into a clique, and run the Split Threshold Editing algorithm from Section 5.1 (see also Proposition 2.7).

Now, (G, k) is a yes-instance if and only if there is a bipartition (A, B) such that when making A into a clique, the resulting instance is a yes-instance for Split Threshold Editing. □

Corollary 5.18. Chain Deletion *and* Chain Completion *are both solvable in time* $2^{O(\sqrt{k}\log k)} + \text{poly}(n)$.

6. Conclusion

In this paper we showed that the problems of editing edges to obtain a threshold graph and editing edges to obtain a chain graph are NP-complete. The latter solves a conjecture in the positive from Natanzon et al. [28] and both results answer open questions from Sharan [32], Burzyn et al. [4], and Mancini [24].

On the positive side, we show that both Threshold Editing and Chain Editing admit quadratic kernels, i.e., given a graph (G, k), we can in polynomial time find an equivalent instance (G', k) where $|V(G')| = O(k^2)$, and furthermore, G' is an induced subgraph of G. We also show that these results hold for the deletion and completion variants as well, and these results answer open questions by Liu et al. in a recent survey on kernelization complexity of graph modification problems [20].

Finally we show that both problems admit subexponential algorithms of time complexity $2^{O(\sqrt{k}\log k)} + \text{poly}(n)$. This answers a recent open question by Liu et al. [22]. These are the only two graph classes known towards which completion, deletion, and editing all are NP-complete, admit polynomial kernels and are subexponential parameterized time solvable.

In addition, we give a proof for the NP-hardness of Chordal Editing which has been announced several places but which the authors have been unable to find. However, our NP-completeness proof for Chordal Editing suffers a quadratic blow-up from 3Sat, i.e., $k = \Theta(|\varphi|^2)$, so we cannot get better than $2^{o(\sqrt{k})} \cdot \text{poly}(n)$ lower bounds from this technique. The current best algorithm for Chordal Editing[3] runs in time $2^{O(k \log k)} \cdot \text{poly}(n)$ [6], and so this leaves a big gap. It would be interesting to see if we can achieve tighter lower bounds, e.g., $2^{o(k)} \cdot \text{poly}(n)$ time lower bounds for Chordal Editing assuming ETH together with a $2^{O(k)} \cdot \text{poly}(n)$ time algorithm.

[3]Here, the authors take Chordal Editing to allow vertex deletions.

7. Funding

This work was supported by the Bergen Research Foundation under the project Beating Hardness by Preprocessing; the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) [ERC Grant Agreement n. 267959]; and the Gordon & Betty Moore Foundation under DDD Investigator Award [GBMF4560] and the DARPA GRAPHS program under [SPAWAR Grant N66001-14-1-4063]. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of DARPA, SSC Pacific, or the Moore Foundation.

Acknowledgment

The authors would like to express their gratitude to Ulrik Brandes and Mehwish Nasim for helpful comments on an early draft of this paper, as well as the editor and the anonymous reviewers for critically reading and providing many insightful comments and suggestions that improved the manuscript.

References

[1] Alon, N., Lokshtanov, D. & Saurabh, S. (2009) Fast FAST. In *ICALP*, volume 5555 of *Lecture Notes in Computer Science*, pages 49–58. Springer.

[2] Brandes, U. (2014) Social Network Algorithmics. ISAAC.

[3] Brandstädt, A., Le, V. B. & Spinrad, J. P. (1999) *Graph Classes. A Survey.* SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, USA.

[4] Burzyn, P., Bonomo, F. & Durán, G. (2006) NP-completeness results for edge modification problems. *Discrete Appl. Math.*, **154**(13), 1824–1844.

[5] Cai, L. (1996) Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, **58**(4), 171–176.

[6] Cao, Y. & Marx, D. (2014) Chordal Editing is Fixed-Parameter Tractable. In *STACS*, volume 25 of *LIPIcs*, pages 214–225. Schloss Dagstuhl.

[7] Chen, Z.-Z., Jiang, T. & Lin, G. (2003) Computing phylogenetic roots with bounded degrees and errors. *SIAM Journal on Computing*, **32**(4), 864–879.

[8] Damaschke, P. (2006) Parameterized enumeration, transversals, and imperfect phylogeny reconstruction. *Theoretical Computer Science*, **351**(3), 337–350.

[9] Dehne, F., Langston, M. A., Luo, X., Pitre, S., Shaw, P. & Zhang, Y. (2006) The cluster editing problem: Implementations and experiments. In *Parameterized and Exact Computation*, pages 13–24. Springer.

[10] Drange, P. G., Dregi, M. S., Lokshtanov, D. & Sullivan, B. D. (2015a) On the Threshold of Intractability. In *ESA*, volume 9294 of *Lecture Notes in Computer Science*, pages 411–423. Springer.

[11] Drange, P. G., Fomin, F. V., Pilipczuk, M. & Villanger, Y. (2015b) Exploring the Subexponential Complexity of Completion Problems. *ACM Trans Comput Theory*, **7**(4), 14:1–14:38.

[12] Drange, P. G. & Pilipczuk, M. (2018) A Polynomial Kernel for Trivially Perfect Editing. *Algorithmica*, **80**(12), 3481–3524.

[13] Feder, T., Mannila, H. & Terzi, E. (2009) Approximating the minimum chain completion problem. *Information Processing Letters*, **109**(17), 980–985.

[14] Flum, J. & Grohe, M. (2006) *Parameterized complexity theory*. Springer-Verlag New York Inc.

[15] Fomin, F. V. & Villanger, Y. (2013) Subexponential parameterized algorithm for minimum fill-in. *SIAM J Comput*, **42**(6), 2197–2216.

[16] Ghosh, E., Kolay, S., Kumar, M., Misra, P., Panolan, F., Rai, A. & Ramanujan, M. S. (2015) Faster parameterized algorithms for deletion to split graphs. *Algorithmica*, **71**(4), 989–1006.

[17] Golumbic, M. C. (2004) *Algorithmic graph theory and perfect graphs*, volume 57. Elsevier.

[18] Guo, J. (2007) Problem Kernels for NP-Complete Edge Deletion Problems: Split and Related Graphs. In *ISAAC*, volume 4835 of *Lecture Notes in Computer Science*, pages 915–926. Springer.

[19] Hammer, P. L. & Simeone, B. (1981) The splittance of a graph. *Combinatorica*, **1**(3), 275–284.

[20] Liu, Y., Wang, J. & Guo, J. (2014) An overview of kernelization algorithms for graph modification problems. *Tsinghua Sci. Technol.*, **19**(4), 346–357.

[21] Liu, Y., Wang, J., Guo, J. & Chen, J. (2012) Complexity and parameterized algorithms for cograph editing. *Theor. Comput. Sci.*, **461**, 45–54.

[22] Liu, Y., Wang, J., You, J., Chen, J. & Cao, Y. (2015) Edge deletion problems: Branching facilitated by modular decomposition. *Theoretical Computer Science*, **573**, 63–70.

[23] Mahadev, N. & Peled, U. (1995) *Threshold graphs and related topics*, volume 56. North Holland.

[24] Mancini, F. (2008) *Graph modification problems related to graph classes*. PhD thesis, University of Bergen.

[25] Masuda, N., Miwa, H. & Konno, N. (2004) Analysis of scale-free networks based on a threshold graph with intrinsic vertex weights. *Physical Review E*, **70**(3), 036124.

[26] Nastos, J. & Gao, Y. (2013) Familial groups in social networks. *Soc. Netw.*, **35**(3), 439–450.

[27] Natanzon, A. (1999) *Complexity and approximation of some graph modification problems*. PhD thesis, Tel Aviv University.

[28] Natanzon, A., Shamir, R. & Sharan, R. (2001) Complexity classification of some edge modification problems. *Discrete Applied Mathematics*, **113**(1), 109–128.

[29] Nocaj, A., Ortmann, M. & Brandes, U. (2015) Untangling the Hairballs of Multi-Centered, Small-World Online Social Media Networks. *J. Graph Algorithms Appl.*, **19**(2), 595–618.

[30] Schoch, D. & Brandes, U. (2015) Stars, Neighborhood Inclusion, and Network Centrality. In *SIAM Workshop on Network Science*.

[31] Shamir, R., Sharan, R. & Tsur, D. (2004) Cluster graph modification problems. *Discrete Appl. Math.*, **144**(1-2), 173–182.

[32] Sharan, R. (2002) *Graph modification problems and their applications to genomic research*. PhD thesis, Tel-Aviv University.

[33] Yannakakis, M. (1981) Computing the minimum fill-in is NP-complete. *SIAM J. Algebr. Discrete Methods*, **2**(1), 77–79.