More Consequences of Falsifying SETH and the Orthogonal Vectors Conjecture *[Full version]*

Amir Abboud* Karl Bringmann† Holger Dell‡ Jesper Nederlof§

November 3, 2017

Abstract

The Strong Exponential Time Hypothesis and the OV-conjecture are two popular hardness assumptions used to prove a plethora of lower bounds, especially in the realm of polynomial-time algorithms. The OV-conjecture in moderate dimension states there is no $\varepsilon > 0$ for which an $O(n^{2-\varepsilon}\text{poly}(d))$ time algorithm can decide whether there is a pair of orthogonal vectors in a given set of size n that contains d-dimensional binary vectors.

We strengthen the evidence for these hardness assumptions. In particular, we show that if the OV-conjecture fails then two problems for which we are far from obtaining even tiny improvements over exhaustive search would have surprisingly fast algorithms. If the OV conjecture is false, then there is a fixed $\varepsilon > 0$ such that:

1. for all d and all large enough k, there is a randomized $\tilde{O}(n^{(1-\varepsilon)k})$ time algorithm for the zero-weight k-clique problem and for the min-weight k-clique problem on d-hypergraphs with n vertices. In particular, this shows that the OV-conjecture is *implied* by the recent popular Weighted k-Clique conjecture.
2. for all c, the satisfiability of sparse TC^1 circuits on n inputs (i.e. circuits with cn wires, depth $c \log n$, and negation, AND, OR, and threshold gates) can be computed in time $O((2-\varepsilon)^n)$.

*IBM Almaden Research Center, CA, USA, amir.abboud@ibm.com

†Max Planck Institute for Informatics, Saarland Informatics Campus, Germany. kbringma@mpi-inf.mpg.de

‡Saarland University and Cluster of Excellence (MMCI), Germany. hdell@mmci.uni-saarland.de

§Eindhoven University of Technology, The Netherlands. j.nederlof@tue.nl. Supported by NWO Veni grant 639.021.438.

1 Introduction

The Strong Exponential Time Hypothesis (SETH) is a cornerstone of contemporary algorithm design that recently gained extensive popularity in usage. As originally formulated by Impagliazzo and Paturi [36], SETH postulates that exhaustive search is essentially best possible to decide the satisfiability of bounded-width CNF formulas. Closely related, the orthogonal vectors (OV) problem is, given two sets A and B of n vectors from $\{0,1\}^d$, to decide whether there are vectors $a \in A$ and $b \in B$ such that a and b are orthogonal (in $\mathbb{Z}^d$). It is known [57] that SETH implies the following hardness conjecture for the orthogonal vectors problem.[1]

Conjecture 1.1 (Moderate-dimension Orthogonal Vectors Conjecture[2] (OVC_{n^δ}))**.** *There are no $\varepsilon, \delta > 0$ such that OV with dimension $d = n^\delta$ can be solved in time $O(n^{2-\varepsilon})$.*

The Consequences of OVC_{n^δ} and SETH. The area to which OVC_{n^δ} (and SETH) has most impact is the study of fine-grained complexity of problems in P. Its consequences are remarkable in their strength, versatility, and diversity. If we assume it holds, we get lower bounds that match existing upper bounds (up to $n^{o(1)}$ factors) for dozens of important problems from areas all across computer science, including pattern matching and bioinformatics (e.g. [8, 10, 43, 1]), graph algorithms (e.g. [50, 6, 34]), computational geometry (e.g. [17]), formal languages [11, 19], time-series analysis [2, 20], and even economics [45] (a longer list can be found in [61]). While the original SETH implies hardness for OV for any dimension $d = \omega(\log n)$, very few lower bounds really benefit from the dimension being so small. One example are the recent hardness of approximation results for problems such as Max Inner Product [5] (perhaps also [12, 14]). For all other results, making the dimension smaller than in OVC_{n^δ} only affects lower order terms.

Another area to which SETH has a lot of impact is the study of exact or fixed parameter tractable algorithms, see e.g [24, 48] or the book by Cygan et al. [25]. An important subject where SETH played an especially important role is graph problems for graphs that have good decompositions such as small treewidth or pathwidth. Tight lower bounds based on SETH were introduced to this area by Lokshtanov et al. [44], and SETH has greatly benefited the development of optimal algorithms for e.g. connectivity problems [28, 26].

Evidence for OVC_{n^δ} and SETH. Since *unconditional* lower bounds of the type $\Omega(n^{1+\varepsilon})$ for any of our problems are far out of reach of current techniques, a central focus of fine-grained complexity is to search for (other forms of) evidence for the truth of its conjectures. All previous approaches for finding such results are discussed in Section 1.2. The focus of our work is on finding statements of the form: If OVC_{n^δ}/SETH is false then we get breakthrough algorithms for other famous problems. A highly desirable and longstanding open question is to prove that falsifying OVC_{n^δ} also refutes other popular conjectures such as APSP[3] or 3-SUM[4].

[1] Note that using fast rectangular matrix multiplication (see e.g. [33]) the problem can be solved in $\tilde{O}(n^2)$ time as long as $\delta < 0.3$.

[2] We adopt the naming introduced by Gao et al. [34].

[3] The APSP problem is to compute all pairwise distances in a graph (given by its adjacency matrix) on n nodes and with edges weights in some polynomial range. It is conjectured to require $n^{3-o(1)}$ time, and many problems, especially on graphs, are known to be equivalent or APSP-hard, e.g. [51, 62, 6, 3, 52, 7, 29].

[4] The 3-SUM problem is to decide if a given set of n integers contains three that sum to zero. It is conjectured that the problem requires $n^{2-o(1)}$ time. Many problems, especially in computational geometry, are known to be 3-SUM-hard, see [32].

The Weighted k-Clique Conjecture. Closely related to APSP is a conjecture about the complexity of Clique which has been increasingly popular in the last few years. Given a graph on n nodes and $O(n^2)$ edges with edge-weights in some polynomial range, how fast can we find the k-clique of minimum weight? The exhaustive search algorithm solves this Min-Weight-k-Clique problem in $O(n^k)$ time, and for $k = 3$ the problem is known to be subcubically-equivalent to APSP [63]: we can solve it in $O(n^{3-\varepsilon})$ time for some $\varepsilon > 0$ if and only if APSP is in $O(n^{3-\varepsilon'})$ time for some $\varepsilon' > 0$. For all integers $k \geq 3$, there is a simple reduction to the $k = 3$ case, and using the fastest known APSP algorithm of Williams [59, 22], we can solve Min-Weight-k-Clique in $n^k/2^{\Omega(\sqrt{\log n})}$ time. Due to the equivalence, the APSP Conjecture states that the $k = 3$ case requires $n^{3-o(1)}$ time, so it is natural to conjecture that for any constant $k \geq 3$ the complexity is also $n^{k-o(1)}$.

Conjecture 1.2 (Weighted k-Clique Conjecture)**.** *No algorithm finds a minimum-weight k-clique in a graph on n nodes with weights in $\{-M, \ldots, M\}$ in $O(n^{(1-\varepsilon)\cdot k} \cdot \operatorname{poly}\log M)$ time, for any $\varepsilon > 0$.*[5]

Note that this conjecture implies the APSP conjecture, and therefore all the known APSP lower bounds [51, 62, 6, 3, 52, 7, 29]. Further lower bounds under it (that are not known to hold under APSP) were shown for the Local Alignment problem from Bioinformatics by Abboud, Vassilevska W. and Weimann [8], the Max Rectangle problem for points in the plane, a basic problem in computational geometry, by Backurs, Dikkala, and Tzamos [9], the Viterbi problem from Machine Learning by Backurs and Tzamos [13], and recently the Tree Edit Distance problem by Bringmann, Gawrychowski, Mozes, and Weimann [18].

1.1 Our Results

Consequences of Falsifying OVC for Clique Problems. Using a combination of simple tricks and gadgets, we design a tight randomized reduction from the Min-Weight-k-Clique problem to OV, showing that: if OVC_{n^δ} is false, then so is the Weighted k-Clique conjecture! Thus, we prove that most "SETH-based" lower bounds in P can be based on this Clique conjecture. Since we already know that the weighted Clique conjecture implies the APSP conjecture, one can consider this to be a unification of the APSP and OV conjectures[6]. The impact of our results on (part of) the landscape of Hardness in P is depicted in Figure 1.1. We thus isolate Min-Weight-k-Clique as the core hardness for the majority of problems in P with known conditional lower bounds (the main exceptions are "3-SUM hard" problems).

Going beyond the implication to the Clique conjecture, we show that falsifying OVC_{n^δ} leads to improved algorithms for finding weighted cliques *even in hypergraphs*. A d-hypergraph is a hypergraph in which all edges are of size at most d. A *clique* of a d-hypergraph G is a subset $X \subseteq V(G)$ such that for every $e \subseteq X$ of size at most d we have $e \subseteq E$. Besides the unweighted version of clique in hypergraphs, we also study weighted versions of this problem where we are additionally given an edge weight function $w : E(G) \to \mathbb{Z}$, and a target integer $t \in \mathbb{Z}$. Specifically, in the *Exact-Weight-k-Clique* and *Min-Weight-k-Clique* problems we need to find a clique X of size k that satisfies respectively $\sum_{e\subseteq X} w(e) = t$ and $\sum_{e\subseteq X} w(e) \leq t$. Our first result now reads as follows:

Theorem 1.3. *If OVC_{n^δ} is false, then for some $\varepsilon > 0$ there is for every d a sufficiently large $k = k(d, \varepsilon)$ such that there are algorithms that solve*

- *k-Clique on d-hypergraphs in $O(n^{(1-\varepsilon)k})$ time.*

[5] In this work we will hardly distinguish between randomized and deterministic algorithms, as even randomized algorithms with the desired running times would constitute big breakthroughs.

[6] Previously, it was known that the $(\min, +)$-Convolution problem [27] provides a unification of APSP and 3-SUM: if we assume that it requires $n^{2-o(1)}$ time, both the APSP and 3-SUM conjectures follow.

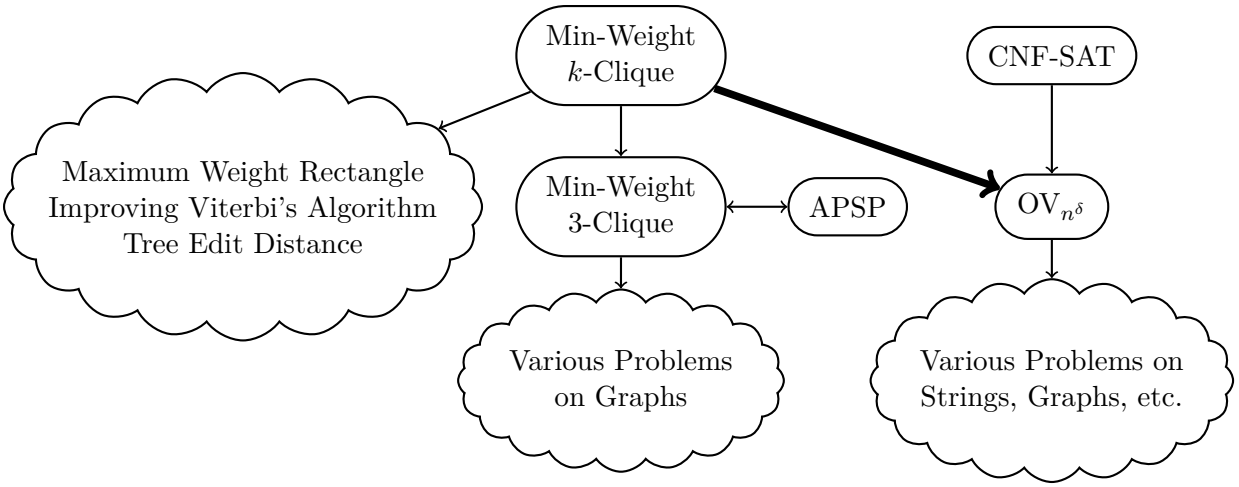

Figure 1: Illustration of the landscape of Hardness in P and the impact of Theorem 1.3, the bold black arrow. An arrow from problem A to problem B indicates that improving the runtime of problem B from $B(n)$ to $(B(n))^{1-\varepsilon}$ implies an improvement for problem A from $A(n)$ to $(A(n))^{1-\varepsilon'}$.

- *Exact-Weight-k-Clique on d-hypergraphs with weights in $\{-M, \ldots, M\}$ in randomized time at most $O(n^{(1-\varepsilon)k} \cdot \operatorname{polylog} M)$.*
- *Min-Weight-k-Clique on d-hypergraphs with weights in $\{-M, \ldots, M\}$ in randomized time at most $O(n^{(1-\varepsilon)k} \cdot \operatorname{polylog} M)$.*

Our reduction is composed of two main stages. In the first one, we reduce Min-Weight-k-Clique on graphs to *unweighted* k-clique on *4-hypergraphs*, where each hyper-edge has cardinality at most 4. More generally, we reduce Min-Weight-k-Clique in d-hypergraphs to k-Clique in $2d$-hypergraphs. This reduction, in turn, has multiple "weight reduction" steps: We start with a standard hashing trick to reduce the weights to a polynomial range. Then, to reduce the weights further, we chop the bits of the numbers into vectors and then use a squaring trick to combine all the coordinates. This trick is borrowed from [4], where it was used to reduce node weights in graphs. We show that it can also be used to reduce edge weights, albeit we have to transform our graph into a hypergraph. Finally, once the weights are small enough, we remove them completely via exhaustive search through tuples with certain sums. In the second stage, we reduce the unweighted hyper-graph problem to OV. Here, we map each node to a vector by designing a natural encoding of the incident hyper-edges into the coordinates, so that an orthogonality check (among k vectors) corresponds to checking that k nodes form a hyper-clique. (Our second set of results will use "number removal" tricks of a similar flavor in order to reduce SAT on circuits with threshold gates to SAT on CNF formulas, as in SETH.)

Even 3-hypergraphs can be harder to handle than graphs. For example, in the unweighted case, k-Clique on graphs can be solved in $O(n^{0.79k})$ time [47, 31], whereas on 3-uniform hypergraphs any $O(n^{k(1-\varepsilon)})$ time would be a breakthrough: it would imply faster algorithms for MAX-3-SAT, a longstanding open question, via a known reduction (outlined in Section 3). This reduction combined with our theorem allows us to base all OV lower bounds on the hardness of MAX-k-SAT rather than just k-SAT. Previously, authors have had to work harder to show that their lower bound is based on the better assumption that MAX-k-SAT is hard, rather than SETH [2, 7, 42]. Our results imply that this was unnecessary since there is a direct reduction from Max-k-SAT to OV.

Corollary 1.4. *If OVC_{n^δ} is false, then there exists an $\varepsilon > 0$ such that for any k Max-k-SAT can be solved $O^*((2-\varepsilon)^n)$ time.*

Finally, we remark that our reduction can also handle the Zero-Weight-k-Clique problem which asks for a k-clique of weight exactly zero, which is in fact at least as hard as the minimization version [46]. While the best known algorithms for Min-Weight-k-Clique run in time $n^k/2^{\Omega(\sqrt{\log n})}$ [59, 22], getting such superpolylogarithmic shavings for the Zero-Weight-k-Clique problem is an open question. The $k = 3$ case is particularly interesting, since solving Zero-Weight-3-Clique in $O(n^{3-\varepsilon})$ time does not only refute APSP but it also refutes the 3-SUM Conjecture [62, 49, 41].

Consequences of Falsifying SETH for SAT on Sparse Circuits. Next, we turn our attention to other forms of evidence, which are based on more general SAT problems. This yields evidence not only for OVC_{n^δ} but even for SETH, which is important since for some problems, especially in the exponential time regime, only SETH-based lower bounds are known.

In some sense, SETH is the strongest possible assumption about the hardness of SAT not know to be false: SETH asserts that if the best running time for k-CNF-SAT is $2^{(s_k \pm o(1))n}$, then $\lim_{k\to\infty} s_k = 1$. If we would simplify CNFs further, say to 10-CNFs or to DNFs, then it would be false. In other words, SETH claims that the weakest functions we do not know how to analyze for satisfiability in $O^*((2-\varepsilon)^n)$ time are hard. In fact, it does not even talk about a single problem that we think is hard, but about a sequence of problems, each of which we know to have a faster algorithm than exhaustive search. This is great from a perspective of reductions, but is bad if we want to have confidence. Indeed, there are even algorithms that get substantial $n^{\omega(1)}$ speed-ups over 2^n for CNF formulas of unbounded width (see e.g. [16, 21]) but if we go a bit higher in complexity, say to linear size constant depth circuits with majority gates, then getting such algorithms is a big open question: they would resolve Williams' question [60] of whether his framework for circuit lower bounds can prove that NEXP is not in TC^0 – a result that might be facing the natural proofs barrier.

Note that the satisfiability of cn-size TC^0-circuits of depth two has a known $O^*((2-\varepsilon_c)^n)$-time algorithm [37, 23], but the constant ε_c tends to 0 as c grows. For larger depths, such algorithms are known for AC^0 [35] but not for TC^0.

We show that a refutation of SETH would provide a big step forward on all these questions:

Theorem 1.5. *If SETH fails, then there is an $\varepsilon > 0$ such that, for all constants c and d, the satisfiability of depth-d threshold circuits with cn wires can be determined in time $O^*((2-\varepsilon)^n)$.*

This result is akin to results of Santhanam and Srinivasan [53] and Dantsin and Wolpert [30], and Cygan et al. [24] who showed that refuting SETH implies faster SAT algorithms for determining satifiability of linear-size formulas and of AC^0-circuits, respectively. Our result is qualitatively stronger, not only because we can handle a larger class of circuits, but also because, unlike CNFs, linear-size formulas, and linear-size AC^0-circuits, even $n^{\omega(1)}$ improvements are not known for SAT on linear-size circuits with threshold gates of depth 4 or more. The only previous connection of this form that we are aware of is the analogous result of Cygan et al. [24] for VSP-circuits. The latter result is facilitated by the depth reduction result of Valiant, which shows that VSP-circuits embed nicely into CNF-formulas. We use an additional trick that allows us to get rid of threshold gates.

Much like most lower bounds in P can be based on OVC_{n^δ} rather than the OV Conjecture with dimension $d = \omega(\log n)$, many SETH-based lower bounds for exponential time and parameterized problems can be based on the slightly weaker assumption that CNF-SAT cannot be solved in $O^*((2-\varepsilon)^n)$ time, e.g. for graph problems that have small treewidth or pathwidth [44, 28, 26].

We add weight to these hardness results by showing that sufficiently fast algorithms for CNF-SAT already imply fast algorithms for determining the satisfiability of sparse threshold circuits of super-logarithmic depth, that is, a larger class than TC^1-circuits:

Theorem 1.6. *If CNF-SAT can be solved in $O^*((2-\varepsilon')^n)$ time for some $\varepsilon' > 0$, then there is an $\varepsilon > 0$ such that for any $c > 0$ there is a $\delta > 0$ such that the satisfiability of threshold circuits with depth $(\log n)^{1+\delta}$ and at most cn wires can be determined in time $O^*((2-\varepsilon)^n)$.*

The same conclusion holds if OVC_{n^δ} fails, since this implies that CNF-SAT can be solved in $O^*((2-\varepsilon)^n)$ time for some $\varepsilon > 0$ via standard reductions [57].

1.2 Previous Work on Evidence for the Conjectures

All known consequences of falsifying OVC_{n^δ}/SETH that we are aware of are summarized next.

It is known that these conjectures hold under certain restrictions on the algorithms. Beck and Impagliazzo [15] proved that a version of the popular algorithmic technique of *resolution*, on which many SAT-solvers are based, is not sufficient for refuting SETH. Very recently, Kane and Williams [40] proved that no algorithms implementable by Boolean formulas or Branching Programs can solve OV on vectors of logarithmic dimension in subquadratic time.

More similar to our work, are results showing algorithmic consequences for other problems. Cygan et al. [24] showed that $O^*((2-\varepsilon)^n)$ time algorithms for CNF-SAT imply $O^*((2-\varepsilon')^n)$ time algorithms for four other NP-hard problems such as Hitting Set, Set-Splitting, Not-All-Equal-SAT, and SAT on linear-size VSP-circuits. Santhanam and Srinivasan [53] and Dantsin and Wolpert [30] showed the same for SAT on linear-size formulas and AC^0-circuits, respectively. More recently, Gao, Impagliazzo, Kolokolova, and Williams [34], showed that if OVC_{n^δ} is false then we can solve all model-checking problems over first order sentences with k quantifiers in hyper-graphs on m edges in $O(m^{k-\varepsilon})$ time. These are problems typically studied in logic and databases, and perhaps the most famous problem in this class from the perspective of the algorithms community is the k-Clique problem. Our work shows that even weighted versions of Clique are reducible to OV. This is significant because $O(n^{(1-\varepsilon)k})$ algorithms are known for the unweighted case [47, 31], and so the connection of Gao et al. is not tight in this case.

Another consequence of refuting SETH that is often used as evidence in a controversial way is that it would imply *new circuit lower bounds*. Specifically, it would imply that the powerful class NEXP is not contained in the class of functions computable by linear-size VSP-circuits [58, 38]. From our Theorem 1.6, it follows that solving CNF-SAT in $O^*((2-\varepsilon')^n)$ time also implies the perhaps more natural result that NEXP is not in linear-size TC^1.

Note we give a schematic overview of our results in Appendix A.

2 Preliminaries

Notation. We write $\mathbb{N}$ for the set of natural numbers, including 0, and $\mathbb{Z}$ for the set of integers. We let $[n]$ denote $\{1, \ldots, n\}$ for $n \in \mathbb{N}$. For a set S, we write $\binom{S}{d}$ for the set of all subsets of S that have size exactly d, and $\binom{S}{\leq d}$ for the set of all subsets of size at most d. A d-hypergraph G for $d \in \mathbb{N}$ is a tuple $(V(G), E(G))$, where $V(G)$ is a finite set of *vertices* and $E(G) \subseteq \binom{V(G)}{\leq d} \setminus \{\emptyset\}$ is a set of *edges*. If G is a d-hypergraph and $X \subseteq V(G)$, then $G[X]$ denotes the *subgraph induced by* X, that is, $V(G[X]) = X$ and $E(G[X]) = E(G) \cap \binom{X}{\leq d}$. A set $S \subseteq V(G)$ is called a clique in G if the graph $G[S]$ induced by S contains all edges from $\binom{S}{\leq d}$. A k-clique is a clique of size k. A *graph* is a 2-hypergraph in which every edge has size 2.

The $O^*(\cdot)$ notation omits factors that are polynomial in the input size.

CNF-SAT. The k-SAT problem is to determine whether a given k-CNF formula has a satisfying assignment. We denote the number of variables by n and define

$$s_k = \inf\{\, \delta > 0 : \text{ there exists an } O^*(2^{\delta n}) \text{ time algorithm for } k\text{-SAT} \,\}.$$

Let $s_\infty = \lim_{k\to\infty} s_k$. Impagliazzo and Paturi's *Strong Exponential Time Hypothesis (SETH)* postulates that $s_\infty = 1$ holds [36].

DAGs and Circuits. If G is a directed acyclic graph (DAG), we let $N^-_G(v)$ denote the set of in-neighbors of v and let $d^-_G(v) = |N^-_G(v)|$ be the *in-degree*. The *depth* of G is the length of the longest path in it.

A *Boolean function* is a function $f : \{0,1\}^d \to \{0,1\}$ for some $d \in \mathbb{N}$. It is *symmetric* if $f(x) = f(y)$ holds for all $x, y \in \{0,1\}^d$ whose Hamming weight is the same. Let B be a set of symmetric Boolean functions. A *(Boolean) circuit C over a basis B* is a pair (G, λ) where G is a directed acyclic graph and $\lambda \in B^V$ is a labeling of its vertex set V with elements from B. We say that v is a *λ_v-gate*, and we require that the in-degree of v is equal to the arity of λ_v, that is, we have $\lambda_v : \{0,1\}^{d^-_G(v)} \to \{0,1\}$. The edges of G are called *wires*, the in-degree of a gate is called its *fan-in*, and we write $V(C)$ for $V(G)$. The set of *input gates* $I(C)$ or $I(G)$ of C consists of the vertices with in-degree 0, and the set of *output gates* $O(C)$ or $O(G)$ of C consists of the vertices with out-degree 0. If $x \in \{0,1\}^{I(G)}$ is a setting for the input gates, we define the $C_v(x)$ as the *value of C at $v \in V$ on input x* inductively: If $v \in I(C)$, let $C_v(x) = x_v$, and otherwise, let $C_v(x) = \lambda_v(C_{v_1}(x), \ldots, C_{v_\ell}(x))$, where $v_1, \ldots, v_\ell$ denotes the in-neighbors of v in G; note that this is well-defined since G is acyclic and λ_v is symmetric. Slightly abusing notation, we may write C also for the function $C : \{0,1\}^{I(G)} \to \{0,1\}^{O(G)}$ with $C(x) = (C_v)_{v\in O(G)}$, or we may view circuits as mapping integers to integers in a fixed range $[r]$ for convenience while in fact this is implemented by storing the binary representation of these values with $\lceil \lg r \rceil$ gates.

A (u,v)-path in C is a directed path in G that starts in u and ends at v. If $A \subseteq V(C)$, we let $R_C(A, v)$ denote the set of vertices from which v is reachable without using vertices of A, that is,

$$R_C(A, v) = \Big\{\, u \in V : \text{ there is a } (u,v)\text{-path in } G[(V \setminus A) \cup \{u, v\}] \,\Big\}. \tag{1}$$

Finally, for a circuit C, a gate $v \in V(C)$, and a set $A \subseteq V(C)$, we define $C_{v,A}$ as the subcircuit of C that is induced by the set $R_C(A, v)$; note that v is the only output gate of $C_{v,A}$ and its input gates are contained in $A \cup I(C)$.

We use the Boolean functions $\mathrm{NEG}(x) = \neg x$, $\mathrm{AND}(x,y) = x \wedge y$, $\mathrm{OR}(x,y) = x \vee y$ and $\mathrm{TH}_\theta : \{0,1\}^d \to \{0,1\}$ which is, for every positive $\theta \le n$ defined to be 1 if $\sum_{i=1}^d x_i \ge \theta$ and to be 0 otherwise. Note that $\mathrm{AND}(x,y) = \mathrm{TH}_2(x,y)$ and $\mathrm{OR}(x,y) = \mathrm{TH}_1(x,y)$. We also use $\mathrm{MOD}_m(x_1, \ldots, x_d)$ for $m \le d$ which is defined to be 1 if m divides $\sum_{i=1}^d x_i$ and to be 0 otherwise, and $\mathrm{MAJ}(x_1, \ldots, x_d) = \mathrm{TH}_{d/2}(x_1, \ldots, x_d)$.

A Boolean circuit over the basis $\{\mathrm{NEG}, \mathrm{AND}, \mathrm{OR}, \mathrm{TH}_\theta\}$, where all gates (except for NEG) may have unbounded fan-in, is called a *threshold circuit* (TC); we use AND and OR only for syntactic convenience as they can be simulated by TH_θ. The problem TC-SAT is given a threshold circuit C with exactly one output gate to decide whether the circuit is satisfiable, that is, whether there exists a setting $x \in \{0,1\}^n$ for the n input gates such that $C(x) = 1$. For $d \in \mathbb{N}$ and $c > 0$, a *c-sparse-d-depth-TC* is a threshold circuit with n variables, at most cn wires, and depth at most d. For each $i \in \mathbb{N}$, a TC^i-circuit is a family of threshold circuits of depth $O(\log^i n)$ and size $\mathrm{poly}(n)$.

3 Weighted Cliques in Hypergraphs

Recall that in the Exact-Weight-k-Clique problem on d-hypergraphs we are given a d-hypergraph G and a target value t, and the task is to decide whether some size-k subset $S \subseteq V(G)$ forms a clique of total weight $\sum_{e \subseteq S} w(e) = t$. We denote by $M = M(w,t)$ the maximum weight in absolute value, that is, we have $M = \max(\{|t|\} \cup \{|w(e)| : e \in E(G)\})$. We write $n = |V(G)|$. Since in this section we will mostly deal with the Exact-Weight-k-Clique on d-hypergraphs problem, we will abbreviate it to "weighted d-hypergraph k-clique".

3.1 Preprocessing reductions

We rely on some basic reductions: The first makes the hypergraph a complete d-hypergraph, which shows that the graph structure is immaterial for this problem; the second makes the hypergraph k-partite, which will be useful in our constructions; the third reduces from "exact weight clique" to "zero weight clique", that is, it sets the target value t to 0 but makes use of negative edge weights; the fourth uses a non-negative target value but removes negative weights. In the following statement, we use M' to denote the maximum weight $M(w',t')$ of the output instance.

Fact 3.1. *Let $d, k \in \mathbb{N}$ with $1 \leq d \leq k$. There are $O(n^d)$-time self-reductions for weighted d-hypergraph k-clique with the following properties:*

1. *"Make complete": maps an instance (G,w,k,t) to (G',w',k,t') where $V(G') = V(G)$, $E(G') = \binom{V(G)}{\leq d}$, and the maximum weights satisfy $M' \leq \binom{k}{\leq d} M$.*
2. *"Make k-partite": maps an instance (G,w,k,t) to (G',w',k,t) where $|V(G')| \leq k|V(G)|$, the maximum weights satisfy $M = M'$, and G' is k-partite in the sense that $V(G')$ is partitioned such that every edge intersects each part in at most one vertex.*
3. *"Make target zero": maps a k-partite instance (G,w,k,t) to (G,w',k,t') where $t' = 0$ and the maximum weights satisfy $M' \leq 2M$.*
4. *"Make weights non-negative": maps an instance (G,w,k,t) to (G,w',k,t') where $w' : E(G) \to \mathbb{N}$ and $t' \in \mathbb{N}$ holds, and we have $M' \leq 2\binom{k}{\leq d}^2 M$.*

Proof. Let (G,w,k,t) be an instance for the problem.

For the first claim, we set $w(e) = \binom{k}{\leq d} M$ for edges e that are supposed to be absent; such edges cannot be used by any solution. Hence, we can assume $E(G) = \binom{V(G)}{\leq d}$ without loss of generality.

For the second claim, we define $V(G') = \{1,\dots,k\} \times V(G)$. For every pairwise distinct $a_1,\dots,a_{d'} \in \{1,\dots,k\}$ and every edge $\{v_1,\dots,v_{d'}\} \in E(G)$ of size d', we add an edge $f = \{(a_1,v_1),\dots,(a_{d'},v_{d'})\} \subseteq V(G')$ to G'. We set the weight $w'(f) = w(\{v_1,\dots,v_{d'}\})$. It is clear that this instance is equivalent to the input instance, and k-partite (the parts consist of vertices with equal first coordinate).

For the third claim, we slightly modify the weights by setting $t' = 0$ and subtracting t from certain edge weights. Specifically, for any edge of cardinality d, denoted by $f = \{(a_1,v_1),\dots,(a_d,v_d)\}$, we set $w'(f) = w(\{v_1,\dots,v_d\})$ if $\{a_1,\dots,a_d\} \neq \{1,\dots,d\}$ and $w'(f) = w(\{v_1,\dots,v_d\}) - t$ if $\{a_1,\dots,a_d\} = \{1,\dots,d\}$. Note that any k-clique in G' contains exactly one edge f that intersects the first d parts of the k-partition in exactly one vertex each.

For the fourth claim, we first ensure that $E(G) = \binom{V(G)}{\leq d}$ using the first claim, which increases M by at most a factor $\binom{k}{\leq d}$. Let $L = \max\{0, -w(e) : e \in E(G)\}$, that is, L is the absolute value of the smallest negative weight that occurs in the input, or 0 if there is none. We set $w'(e) = w(e) + L$

for all e and $t' = t + L\binom{k}{\leq d}$. If $t' < 0$ or $t' > \max\{w'(e)\} \cdot \binom{k}{\leq d}$, the instance is a trivial no-instance. Otherwise the reduction outputs (G, w', k, t'). □

3.2 Weight reduction: From arbitrary to polynomial

We proceed by reducing the weights of a given instance of the weighted d-hypergraph k-clique problem. By taking the numbers modulo a random prime, we can reduce the maximum weight from M to $n^{O(k)}$ in the following way:

Lemma 3.2. *Let $d, k \in \mathbb{N}$ with $1 \leq d \leq k$. For some constant $f(k,d) \in \mathbb{N}$ there is a randomized $f(k,d) \cdot \operatorname{polylog} M$ time self-reduction for the d-hypergraph k-clique problem that, on input an instance (G, w, k, t) with maximum weight M, makes at most $f(k,d)$ queries to instances (G, w', k, t') where $w' : E(G) \to \mathbb{N}$, $t' \in \mathbb{N}$, $M' \leq f(k,d) \cdot n^{O(k)}$, and the success probability of the reduction is at least 99%.*

Proof. If $M \leq n^k$ holds, we do not need to do anything. If $M \geq \exp(n^k)$ holds, then in time $O(n^k \log M) = \operatorname{polylog}(M)$ we can brute-force the problem. In the remaining case, we sample a prime p uniformly at random from a range specified later, and set $w'(e) = w(e) \bmod p$ for all e. We query the oracle (G, w', k, t') for all t' with $t' = jp + (t \bmod p)$ and $j \in \{0, \ldots, \binom{k}{\leq d}\}$, and we output *yes* if and only if at least one oracle query returns *yes*. To prove the completeness of this reduction, let S be a k-clique of weight t with respect to w. Then $\sum_{e \subseteq S}(w(e) \bmod p) = jp + ((\sum_{e \subseteq S} w(e)) \bmod p) = jp + (t \bmod p) = jp + t'$ holds for some j in the specified range since $\binom{k}{\leq d}$ is the number of terms in the sum. Hence yes-instances map to yes-instances with probability 1. Conversely, if such a j exists, then the weight of S modulo p is equal to t' modulo p.

For the soundness of the reduction, we need to specify the sampling process for p. This is implemented as follows: let $Q = 200n^k \log(k^d M)$ and sample positive integers bounded by $O(Q \ln Q)$ uniformly at random until we have found a prime (which we can verify, for example, deterministically in time $O(\operatorname{poly} \log Q) = O(\operatorname{poly} \log M)$ since $M > n$). By the prime number theorem, with probability at least 99.5%, after $O(\ln Q) \leq O(dk \log n)$ samples we have found a prime that is a uniform sample from a set of at least Q primes.

The weight of each k-clique S in G is at most $k^d M$ in absolute value, and there are at most n^k distinct sets S, and so n^k is also an upper bound on the number of distinct weights that appear. For the soundness of the reduction, it is sufficient that $w(S)$ is not congruent to t modulo p. As $|t - w(S)|$ is at most $k^d M$, it has at most $\log(k^d M)$ prime divisors. Therefore the probability that for some S it holds that $w(S)$ is congruent to t modulo p is at most $n^k \log(k^d M)/Q$. Overall, we succeed at finding a prime with the desired property with probability $(99.5\%)^2 \geq 99\%$.

We indeed make at most k^d queries to the oracle, and the largest weight in each query is bounded by $\binom{k}{\leq d} \cdot p$. Since p is bounded by Q and $M \leq \exp(n^k)$, this is at most $f(k,d)n^{O(k)}$. For the running time, we need to worry about the bitlength of the involved weights. The input weights use at most $\log M$ bits, and so Q (and thus any p) uses at most $O(dk \log n + \log M) = \operatorname{polylog} M$ bits, and computing the weights modulo p can be done in time $\operatorname{polylog} M$. □

3.3 Weight reduction: From polynomial to unweighted

Now we show a deterministic reduction that further reduces the weights from $n^{O(k)}$ to $f(k,d) \cdot \log n$. The *q-expansion* of a number $w \in \mathbb{N}$ is the unique sequence $w_0, w_1, \cdots \in \{0, \ldots, q-1\}$ with $w = \sum_{\ell \in \mathbb{N}} w_\ell q^\ell$. The following lemma uses carries to split the weight constraint along the q-expansions of the edge weights w and the target t; this will be used to reduce the magnitude of the weights.

Lemma 3.3. *Let $w : E(G) \to \mathbb{N}$ and $S \subseteq V(G)$ with $|S| = k$. Let $t \in \mathbb{N}$. Let $q \in \mathbb{N}$ with $q \geq 2$ and $p = \lceil \log_q t \rceil$. We write $(w_\ell)_\ell$ and $(t_\ell)_\ell$ for the q-expansion of w and t, respectively. The following are equivalent:*

1. *$\sum_{e \subseteq S} w(e) = t$.*
2. *There exists a unique sequence $(c_\ell)_{\ell \in \mathbb{N}}$ with $c_\ell \in \{0, \ldots, 2\binom{k}{\leq d}\}$ for all ℓ such that $c_0 = 0$ and $\sum_{e \subseteq S} w_\ell(e) + c_\ell = t_\ell + q \cdot c_{\ell+1}$ for all ℓ.*
3. *There exists a unique sequence $(c_\ell)_{\ell \in \mathbb{N}}$ with $c_\ell \in \{0, \ldots, 2\binom{k}{\leq d}\}$ for all ℓ such that $c_0 = 0$ and*

$$\sum_{\ell \in \mathbb{N}} \Big(c_\ell - t_\ell - q \cdot c_{\ell+1} + \sum_{e \subseteq S} w_\ell(e) \Big)^2 = 0 \,. \tag{2}$$

Proof. The equivalence between the last two claims follows from the fact that the 2-norm is a norm (or a sum of squares is only 0 when all squares are 0). To see that the first claim implies the second, note that we can inductively set the carries so as to satisfy the linear equations. Since these carries are unique non-negative integers, we just need to prove the $2\binom{k}{\leq d}$ upper bound. Note that $c_1 \leq \binom{k}{\leq d}$ since there are at most $\binom{k}{\leq d}$ edges in $G[S]$ and each edge e contributes a weight of $w_0(e) < q$. Inductively, we have $c_{\ell+1} \leq \binom{k}{\leq d} + c_\ell / q \leq 2\binom{k}{\leq d}$ since $q \geq 2$.

To see that the second claim implies the first, we observe $\sum_{\ell \in \mathbb{N}} q^\ell \cdot \sum_{e \subseteq S} w_\ell(e) = \sum_\ell q^\ell (t_\ell + q c_{\ell+1} - c_\ell) = t + \sum_{\ell > 0} q^\ell c_\ell - \sum_\ell q^\ell c_\ell = t - c_0 = t$. □

The following algorithm uses (2) to reduce weights; in particular, we use the binomial theorem $(a+b)^2 = a^2 + 2ab + b^2$ in (2) (with $a = c_\ell - t_\ell - q \cdot c_{\ell+1}$) and then collect terms depending on which vertices of G the weight terms depend on – the terms not depending on edge weights are collected into the target integer. As discussed in the introduction, note that this approach was used before to reduce weights of cliques by Abboud et al. [4] in the more specific setting of *node weights* in graphs (rather than hypergraps).

Algorithm A (*Weight reduction for the weighted k-clique problem*) *Given a d-hypergraph G with edge weights $w : E(G) \to \mathbb{Z}$, a number k, a weight target $t \in \mathbb{Z}$, a parameter $p \in \mathbb{N}$, and access to an oracle for weighted k-clique in $2d$-hypergraphs, the following algorithm finds a k-clique of weight t in G:*

A1 *(Make k-partite and non-negative)* Apply Fact 3.1 to make the instance complete and k-partite and all weights non-negative.

A2 *(Set parameters)* Let $M = \max(\{t\} \cup \{w(e) : e \in E(G)\})$ and let $q \in \mathbb{N}$ be such that $p = \lceil \log_q M \rceil$.

A3 *(Guess carries)* Exhaustively guess $c_\ell \in \{1, \ldots, 2\binom{k}{\leq d}\}$ for each $\ell \in \{1, \ldots, p\}$; set $c_0 = 0$. For each such guess do the following:

a *(Compute new weights)* For every set $f \in \binom{V(G)}{\leq 2d}$, set $w'(f) \in \mathbb{N}$ such that

$$w'(f) = \sum_{\ell=0}^{p} \Big(2 \cdot [f \in E(G)] \cdot w_\ell(f) \cdot (c_\ell - t_\ell - q \cdot c_{\ell+1}) + \sum_{\substack{e_1, e_2 \in E(G) \\ e_1 \cup e_2 = f}} w_\ell(e_1) \cdot w_\ell(e_2) \Big) \,, \tag{3}$$

and $t' = -\sum_{\ell=0}^{p-1} (c_\ell - t_\ell - q \cdot c_{\ell+1})^2$.

b *(Call oracle)* If the oracle detects a k-clique S in (G', w') of weight t', then halt and output *yes*; otherwise continue guessing carries.

A4 If no suitable carries were found, output *no*.

We prove the correctness and the required properties of this algorithm.

Lemma 3.4. *Let $d, k \in \mathbb{N}$ with $1 \leq d \leq k$. Algorithm A (with input parameter $p \in \mathbb{N}$) is an oracle reduction from weighted d-hypergraph k-clique to weighted $2d$-hypergraph k-clique. The algorithm runs in time $O(p4^d n^{2d} k^{dp})$ and makes at most k^{dp} oracle queries. Every query is a hypergraph on the same set of vertices. If M is the maximal weight among w and t, then the maximal weight M' of all queries satisfies $M' \leq O(k^{4d} M^{2/p} p)$.*

Proof. Let G be the d-hypergraph with $E(G) = \binom{V(G)}{\leq d}$, edge weight function $w : E(G) \to \mathbb{N}$, and target $t \in \mathbb{N}$ after applying Fact 3.1. Let G' be the $2d$-hypergraph with $E(G') = \binom{V(G)}{\leq 2d}$.

We first prove the correctness of the reduction. By Lemma 3.3, the instance (G, k, w, t) has a k-clique S of total weight t if and only if there exist $(c_\ell)_\ell$ satisfying (2). Now consider the weight of S in G'; a simple application of the binomial theorem yields that the left side of (2) equals

$$\begin{aligned}
&\sum_{\ell \in \mathbb{N}} \left(2 \left(\sum_{e \subseteq S} w_\ell(e) \right) (c_\ell - t_\ell - q \cdot c_{\ell+1}) + (c_\ell - t_\ell - q \cdot c_{\ell+1})^2 + \left(\sum_{e \subseteq S} w_\ell(e) \right)^2 \right) \\
&= -t' + \sum_{\ell \in \mathbb{N}} \left(2 \left(\sum_{e \subseteq S} w_\ell(e) \right) (c_\ell - t_\ell - q \cdot c_{\ell+1}) + \left(\sum_{e \subseteq S} w_\ell(e) \right)^2 \right) \\
&= -t' + \sum_{f \in E(G'[S])} \sum_{\ell \in \mathbb{N}} \left(2 \cdot [f \in E(G)] \cdot w_\ell(f) \cdot (c_\ell - t_\ell - q \cdot c_{\ell+1}) + \sum_{\substack{e_1, e_2 \in E(G) \\ e_1 \cup e_2 = f}} w_\ell(e_1) \cdot w_\ell(e_2) \right) \\
&= -t' + \sum_{f \in E(G'[S])} w'(f),
\end{aligned}$$

where the second equality holds, since every pair e_1, e_2 has a unique union f where its contribution is accounted for. By Lemma 3.3, S is a k-clique of weight t in (G, w) if and only if the right side of the latter equation is equal to 0, which in turn holds if and only if the weight of S with respect to w' is t'.

For the running time, note that the preprocessing takes $O(n^d)$ time. Exhaustive search for the carries takes $O(k^{dp})$ iterations, and each iteration takes time $O(n^{2d} p 4^d)$ because of line A3a in which we need to compute $w'(f)$ for every edge; overall the reduction takes time $O(n^{2d} k^{dp})$ and makes at most k^{dp} oracle queries.

For the weights, note that $c_\ell \leq 2k^d$ holds, and so $|t'| \leq O(k^{2d} q^2 p)$. To get the bound on $w'(f)$, observe that the term $\sum_{e_1 \cup e_1 = f} w_\ell(e_1) w_\ell(e_2)$ is bounded by $2^d q^2$, and the term $w_\ell(f) \cdot (c_\ell - t_\ell - q \cdot c_{\ell+1})$ is bounded by $O(q^2 k^d)$ in absolute value. The preprocessing step relying on Fact 3.1 may have added an additional factor of k^{2d}; overall, all weights are bounded by $O(k^{4d} q^2 p)$. □

We apply Lemma 3.4 to reduce the maximum weights from $\text{poly}(n)$ to $O(\log n)$, which is small enough to allow for exhaustive enumeration to reduce to the problem without weights.

Lemma 3.5. *Let $d, k \in \mathbb{N}$ with $1 \leq d \leq k$ and $f(d,k) \in \mathbb{N}$. There is an $n^{2d+o(1)}$-time oracle reduction from weighted d-hypergraph k-clique with weights in $\{-n^{f(k,d)}, \ldots, n^{f(k,d)}\}$ to unweighted k-partite $2d$-hypergraph k-clique. If the input has n vertices, every oracle query has n vertices and the reduction uses at most $n^{o(1)}$ queries. Here the $o(1)$ terms are of the form $g(k,d)/\sqrt{\log n}$.*

Proof. Let G be a given k-partite d-hypergraph with edge weights $w : E(G) \to \mathbb{Z}$ and target value $t \in \mathbb{Z}$. We apply Lemma 3.4. In particular, setting $p = \sqrt{\log n}$, we get $k^{dp} = n^{o(1)}$ queries and maximum weight $M' \leq O(k^{4d} M^{2/p} p) = n^{o(1)}$.

Each query is now a k-partite instance (G', w', k, t') with maximum weight M'. Note that we treat k and d as constants here. A solution S of G' satisfies $\sum_{e \subseteq S} w'(e) = t'$. Since G' is k-partite, S intersects each part in exactly one vertex, and for each set $C \subseteq \{1, \ldots, k\}$ with $1 \leq |C| \leq d$, there is a unique edge $e_C \subseteq S$ that intersects exactly the color classes in C, and this edge contributes $w'(e_C)$ to the total weight of S. We want to simulate these weights by exhaustively guessing the contribution $w(e_C)$ of each C – to do so, we only keep the edges of color type C that have the guessed weight.

More precisely, for each $C \subseteq \{1, \ldots, k\}$ with $1 \leq |C| \leq d$, we exhaustively guess a weight $a_C \in [-M', M']$. In total, this requires iterating through at most $(2M'+1)^{k^d} = n^{o(1)}$ candidate weight vectors $a = (a_C)_{C \subseteq \{1,\ldots,k\}}$. If the sum $\sum_C a_C$ is not equal to t', we reject the candidate vector and move to the next one. Otherwise, for each C and each edge e intersecting exactly the color classes prescribed by C, we keep e in the graph if and only if $w'(e) = a_C$. In this way, we obtain a k-partite $2d$-hypergraph G_a. For each candidate vector a of weight t', we query the (unweighted) k-clique oracle for $2d$-hypergraphs and output yes if and only if at least one query outputs yes.

The claim on the running time follows, since there are only $n^{o(1)}$ candidate vectors when k is regarded as a constant, and preparing each oracle query G_a can be done in time $n^{2d+o(1)}$, which is almost-linear in the description length of G_a. For the correctness, note that G has a solution S if and only if G' has a solution S. If G' has a solution S, then there is a setting of the a_C corresponding to the solution such that all edges in $G[S]$ survive in G_a, and the oracle finds a k-clique. On the other hand, if S is a k-clique in some G_a, then the used edges have the desired weight in G'. The correctness of the reduction follows. □

3.4 Reduction to Orthogonal Vectors

In this section, we reduce from clique in d-hypergraphs via the k-OV problem to 2-OV. Recall that the k-OV problem is given k sets $X_1, \ldots, X_k \subseteq \{0,1\}^D$ of Boolean vectors and the goal is to find $x_1 \in X_1, \ldots, x_k \in X_k$ such that $\sum_{j=1}^{D} \prod_{i=1}^{k} x_{ij} = 0$ holds, where the sum and product are the usual operations over the integers $\mathbb{Z}$.

Lemma 3.6. *Let $d, k \in \mathbb{N}$ with $1 \leq d \leq k$. There is an $O(n^{d+1} \operatorname{polylog} n)$ time many-one reduction from (unweighted) k-partite d-hypergraph k-clique to k-OV; the number of produced vectors is n and the dimension of the vectors is n^d.*

Proof. Let G be a given k-partite d-hypergraph with parts $V_1, \ldots, V_k$. Let $v_1, \ldots, v_k$ be vertices with $v_i \in V_i$ for all $i \in \{1, \ldots, k\}$. Then $\{v_1, \ldots, v_k\}$ forms a k-clique in G if and only if all non-edges $h \in \overline{E}(G)$ satisfy $h \not\subseteq \{v_1, \ldots, v_k\}$. Here $\overline{E}(G)$ denotes $\{e \in \binom{V(G)}{\leq d} \mid \forall i : |e \cap V_i| \leq 1\} \setminus E(G)$.

We construct the instance $X_1, \ldots, X_k$ of k-OV as follows. For each $v \in V_i$, we create a vector $x_v \in X_i \subseteq \{0,1\}^{\overline{E}(G)}$ as follows: If $h \in \overline{E}(G)$ is disjoint from V_i, we set $x_{v,h} = 1$. If $h \cap V_i = \{v\}$, we set $x_{v,h} = 1$. Otherwise we have $h \cap V_i = \{u\} \neq \{v\}$ for some u, and we set $x_{v,h} = 0$. Clearly the sets $X_1, \ldots, X_k$ contain a total of n Boolean vectors, each with $|\overline{E}(G)| \leq n^d$ dimensions. Moreover,

the sets can be easily computed in $O(n^{d+1} \operatorname{polylog} n)$ time. It remains to prove the correctness of the reduction.

To this end, let $v_1, \ldots, v_k$ with $v_i \in V_i$ for all i be vertices that form a k-clique $\{v_1, \ldots, v_k\}$ in the k-partite d-hypergraph G. We claim that $\{x_{v_i}\}$ is a solution to the k-OV instance, that is, we claim $\sum_{h \in \overline{E}(G)} \prod_{i=1}^{k} x_{v_i,h} = 0$. To see this, let $h \in \overline{E}(G)$ be arbitrary. Since $\{v_1, \ldots, v_k\}$ is a k-clique in G and h is a non-edge of G, there exists a part V_i that satisfies $V_i \cap h \neq \emptyset$ and $v_i \notin h$. By definition, we have $x_{v_i,h} = 0$. Thus the entire sum is indeed zero.

For the reverse direction, let $x_{v_1}, \ldots, x_{v_k}$ with $x_{v_i} \in X_i$ for all i be vectors that form a solution to the k-OV instance. This means that for all $h \in \overline{E}(G)$, there exists some $i \in \{1, \ldots, k\}$ such that $x_{v_i,h} = 0$ holds. By definition, this implies that $h \cap V_i = \{u\} \neq \{v_i\}$ for some u holds. Thus in particular, $h \not\subseteq \{v_1, \ldots, v_k\}$ and so the set $\{v_1, \ldots, v_k\}$ does not contain any non-edges of G and must be a clique. □

The last step of our reduction is reminiscent of the classic SETH-based lower bound for 2-OV [57].

Lemma 3.7. *Let $k \in \mathbb{N}$. There is an $O(n^{\lceil k/2 \rceil} D)$ time many-one reduction from k-OV to 2-OV that maps instances with n vectors in dimension D to instances with $O(n^{\lceil k/2 \rceil})$ vectors in dimension D.*

Proof. Let $X_1, \ldots, X_k \subseteq \{0,1\}^D$ be the input for k-OV with $n = \sum_{i=1}^{k} |X_i|$. The idea is to split the instance into two halves and list all candidate solutions in each half. For each candidate solution $S \subseteq X_1 \cup \cdots \cup X_{\lfloor k/2 \rfloor}$ with $|S \cap X_i| = 1$ for all $i \in \{1, \ldots, \lfloor k/2 \rfloor\}$, we create a vector $v^S \in X_1' \subseteq \{0,1\}^D$ by setting $v_i^S = \prod_{u \in S} u_i$. Similarly, for each $S' \subseteq X_{\lfloor k/2 \rfloor + 1} \cup \cdots \cup X_k$ with $|S' \cap X_i| = 1$ for all $i \in \{\lfloor k/2 \rfloor + 1, \ldots, k\}$, we create a vector $v^S \in X_2' \subseteq \{0,1\}^D$ by setting $v_i^S = \prod_{u \in S} u_i$. We obtain an instance $X_1', X_2' \subseteq \{0,1\}^D$ of 2-OV.

We claim that $X_1, \ldots, X_k$ is a yes-instance of k-OV if and only if X_1', X_2' is a yes-instance of 2-OV. Suppose that $v_1, \ldots, v_k$ are orthogonal, that is, $\prod_{i=1}^{k} (v_i)_j = 0$ holds for all $j \in \{1, \ldots, D\}$. We set $S = \{v_1, \ldots, v_{\lfloor k/2 \rfloor}\}$ and $S' = \{v_{\lfloor k/2 \rfloor + 1}, \ldots, v_k\}$. Clearly $v_j^S \cdot v_j^{S'} = 0$ holds for all j, so $v^S, v^{S'} \in V'$ are indeed orthogonal. Conversely, if $v_j^S \cdot v_j^{S'} = 0$ holds for all j, then the k vectors in $S \cup S'$ are orthogonal. □

3.5 Tying things together

We now formally prove Theorem 1.3.

Theorem 1.3 (restated)**.** *If OVC_{n^δ} is false, then for some $\varepsilon > 0$ there is for every d a sufficiently large $k = k(d, \varepsilon)$ such that there are algorithms that solve*

- *k-Clique on d-hypergraphs in $O(n^{(1-\varepsilon)k})$ time.*
- *Exact-Weight-k-Clique on d-hypergraphs with weights in $\{-M, \ldots, M\}$ in randomized time at most $O(n^{(1-\varepsilon)k} \cdot \operatorname{polylog} M)$.*
- *Min-Weight-k-Clique on d-hypergraphs with weights in $\{-M, \ldots, M\}$ in randomized time at most $O(n^{(1-\varepsilon)k} \cdot \operatorname{polylog} M)$.*

Proof. Let (G, w, k, t) be a given instance of Min-Weight-k-Clique on d-hypergraphs. We summarize the lemmas of this section as follows:

1. Lemma 3.2 randomly reduces in $O(\operatorname{polylog}(M))$ time Exact-Weight-k-Clique with weights up to M to a constant (which depends on k and d) number of instances of Exact-Weight-k-Clique on G with weights up to $n^{O(k)}$.

2. Lemma 3.5 reduces this in $n^{2d+o(1)}$ time to $n^{o(1)}$ instances of k-Clique on $2d$-hypergraphs.

3. Lemma 3.6 reduces in $n^{2d+1+o(1)}$ time one such an instance of k-Clique on $2d$-hypergraphs to one instance of k-OV with n vectors in n^{2d} dimensions.

4. Lemma 3.7 reduces in time $n^{\lceil k/2\rceil+2d+o(1)}$ one such an instance of k-OV to one instance of 2-OV with $O(n^{\lceil k/2\rceil})$ vectors in n^{2d} dimensions.

Composing these reductions gives a randomized $O(n^{\lceil k/2\rceil+2d+o(1)} + \operatorname{polylog}(M))$ time oracle reduction from Exact-Weight-k-Clique on d-hypergraphs with n vertices and largest weight M to 2-OV on $n^{\lceil k/2\rceil}$ vectors of dimension n^{2d} using $n^{o(1)}$ oracle calls.

To reduce from Min-Weight clique to Exact-Weight clique, we use [46, Theorem 1], which allows us to perform a binary search for the minimum-weight clique by making few queries to exact-weight clique. As the domain $E(G)$ of our weight function is of size at most n^d and the maximum weight is upper bounded by M this reduction requires $O(n^d \log M)$ oracle calls. This yields a randomized $O(n^{\lceil k/2\rceil+2d+o(1)} \operatorname{polylog}(M))$ time oracle reduction from Min-Weight-k-Clique on d-hypergraphs with n vertices and largest weight M to 2-OV on $n^{\lceil k/2\rceil}$ vectors of dimension n^{2d} using $n^{d+o(1)} \log M$ oracle calls.

Using only steps 3 and 4, we obtain a $O(n^{\lceil k/2\rceil+2d+o(1)})$ time many-one reduction from k-Clique on d-hypergraphs with n vertices to 2-OV on $n^{\lceil k/2\rceil}$ vectors of dimension n^{2d}.

Now we compose these reductions with the assumed $N^{2-\varepsilon'}$ time algorithm for 2-OV on N vectors in dimension N^δ. This yields algorithms running in time

$$O\Big(\big(n^{\lceil k/2\rceil+2d+o(1)} + n^{\lceil k/2\rceil(2-\varepsilon')+d+o(1)}\big) \operatorname{polylog}(M)\Big).$$

For sufficiently large $k = k(d, \varepsilon')$, this is $O(n^{k(1-\varepsilon)} \operatorname{polylog}(M))$ for some $\varepsilon > 0$. □

We obtain the following corollary to Theorem 1.3.

Corollary 1.4 (restated)**.** *If the OV Conjecture is false, then there is some $\varepsilon > 0$ such that, for all $d \in \mathbb{N}$, there is an $O^*((2-\varepsilon)^n)$ time algorithm for Max-d-SAT.*

The corollary follows from Theorem 1.3 with a reduction from k-Clique on d-hypergraphs to Max-d-SAT that was already sketched in e.g. [57]. We formally state and prove this reduction next. Note that for constant k sufficiently larger than d, combining this reduction with an $O(n^{(1-\varepsilon)k}\operatorname{poly}\log M)$ time algorithm for the Min-Weight-k-Clique problem in d-hypergraphs yields an $O^*(2^{(1-\varepsilon)n})$ time algorithm for Max-d-SAT. Combined with Theorem 1.3 this proves Corollary 1.4.

Lemma 3.8. *Let $d, k \in \mathbb{N}$ with $1 \le d \le k$. There is an $O^*(2^{dn/k})$ time reduction from Max-d-SAT to Min-Weight-k-Clique for d-hypergraphs that maps d-CNF formulas with n variables and m clauses to d-uniform hypergraphs with at most $k2^{n/k}$ vertices and maximum weight $M \le 2m$.*

Proof. Given an instance of Max-d-SAT consisting of a d-CNF formula φ on variable set V of size n and m clauses, and an integer t indicating the required number of satisfied clauses, partition V into sets $V_1, \dots, V_k$ where $|V_k| \le n/k$.

Now build a d-hypergraph H with vertices $\bigcup_{i=1}^k P_i$ where P_i contains a vertex p_x^i for every vector $x \in \{0,1\}^{V_i}$. For every set $\{i_1, \dots, i_\ell\} \in \binom{[k]}{\le d}$, and tuple $(x_1, \dots, x_\ell) \in P_{i_1} \times P_{i_2} \dots \times P_{i_\ell}$, create an edge $f = \{x_1, \dots, x_\ell\}$. Define the weight of f to be -1 times the number of clauses that

1. are contained in $\bigcup_{j=1}^\ell V_{i_j}$,

2. contain a variable in V_{i_j} for every $j = 1, \ldots, \ell$, and

3. are satisfied by the partial assignment obtained by setting the variables in V_{i_j} according to x_j.

To ensure that a small-weight clique intersects with every P_i in at most one vertex, we also add edges (u, v) for every $u \neq v$ with $u, v \in P_i$ with weight $2m$. The target instance is H, and the goal is to decide whether the minimum weight of any k-clique is at most $-t$. As the number of edges of H is at most $(k2^{n/k})^d$ and for every edge we can in polynomial time compute its weight, the running time of this reduction can be bounded by $O^*(2^{dn/k})$.

To see that this is a correct reduction, let $X \subseteq V(H)$ be a k-clique of H of weight at most $-t$. We see that X contains at most one vertex from every P_i, and as $|X| = k$ we have that X intersects in exactly one vertex with every P_i. By definition of the sets P_i, the set X thus corresponds to an assignment x of the variables of φ. We claim that the weight of X is -1 times the number of clauses satisfied by x and therefore φ has an assignment satisfying at least t clauses. To see the claim, let C be a clause of φ and $\{i : C \text{ contains variables from } V_i\} = \{i_1, \ldots, i_\ell\}$ be the set of variable groups intersecting C. Let $x_{i_1}, \ldots, x_{i_\ell}$ be the corresponding partial assignments that x induces to $V_{i_1}, \ldots, V_{i_\ell}$. We see that C contributes -1 to the weight of the hyperedge $(x_1, \ldots, x_\ell)$ and 0 to all other edges.

For the reverse direction, suppose x is an assignment satisfying at least t clauses of φ and let x_i be the projecting of x onto V_i. Then by the above claim the weight of $X := \{p^1_{x_1}, \ldots, p^k_{x_k}\}$ is $-t$. □

4 Reducing Sparse Satisfiability Problems to CNF-SAT

A dream theorem would be to reduce the sparse circuit satisfiability problem over the De Morgan basis to the CNF-SAT problem in such a way that a violation of SETH implies that faster algorithm for sparse circuit satisfiability exist. We demonstrate how to do this for sparse formulas as a warm-up, reproving a result of [30], and then prove that it holds for sparse TC^0-circuits as well. We also prove that, if CNF-SAT with unbounded clause width can be decided in time $O^*((2-\varepsilon)^n)$ for some $\varepsilon > 0$, then this is also true for TC^1-circuits. This significantly extends previous equivalences of SETH with the satisfiability problem of sparse formulas [30], VSP-circuits [24], and AC^0-circuits [35].

4.1 Sparse Formulas

Formulas are circuits that become a tree when the input gates are removed. We consider formulas over the De Morgan basis $\{\mathrm{NEG}, \mathrm{AND}, \mathrm{OR}\}$; in particular, we assume the corresponding trees to be binary, that is, the fan-in of every gate is at most two. We use the following simple decomposition lemma for binary trees:

Lemma 4.1 (Impagliazzo, Meka, and Zuckerman [35, Claim 4.4])**.** *Let T be a binary tree with m nodes and let $\ell \in \mathbb{N}$ with $\ell \leq m$. There exists a set $A \subseteq V(T)$ with $|A| \leq 6m/\ell$ such that every connected component $C \subseteq V(T)$ of $T - A$ has at most ℓ nodes and at most three vertices of A are adjacent to vertices of C. Moreover, such a set A can be computed in polynomial time.*

Using this lemma, we can observe that the satisfiability of sparse Boolean formulas reduces to the satisfiability of k-CNF formulas with only a small overhead in the running time.

Algorithm B (*Transform sparse formula to k-CNF*) *Given a Boolean formula F and an positive integer $k \in \mathbb{N}$, this algorithm computes an equivalent k-CNF formula F'.*

B1 *(Compute decomposition)* Let $A \subseteq V(F)$ be the set guaranteed by Lemma 4.1 where $\ell = k/4$ and $T = F - I(F)$ is the tree obtained from F by removing its input gates.

B2 *(Create variables)* Let $x_1, \ldots, x_n$ be the input variables of F; for each $a \in A$, create a variable y_a. Also create a variable y_o where $o \in V(F)$ is the output gate of F.

B3 *(Compute k-CNFs for small subcircuits)* For each $v \in A \cup \{o\}$, do the following:

- **a** Let $F_{v,A}$ be the subcircuit of F induced by the set $R_C(A, v)$ and interpret the gates $a \in A$ as input variables y_a.
- **b** Since $F_{v,A}$ depends on at most 2ℓ variables, we can compute a k-CNF formula F'_v with at most $2\ell + 1 \leq k$ variables that expresses the constraint "$y_v = F_{v,A}(x, y)$".

B4 *(Output)* Let $F' = y_o \wedge \bigwedge_{v \in A} F'_v$, and output F'.

We prove the correctness and properties of this algorithm in the following lemma.

Lemma 4.2. *Let $c, \varepsilon > 0$. There exists a $k \in \mathbb{N}$ such that algorithm B is a polynomial-time many-one reduction for the satisfiability problem that maps formulas with n variables and at most cn gates to a k-CNF formula with at most $(1 + \varepsilon) \cdot n$ variables.*

Proof. We set $k = c/(24\varepsilon)$. Let F be the input formula with n variables $x_1, \ldots, x_n$ and at most $m \leq cn$ gates. We claim that F' has a satisfying assignment if and only if F does, and F' is a k-CNF formula with at most $(1 + \varepsilon)n$ variables.

Let T be the tree obtained when removing the input gates, and let $A \subseteq V(T)$ be the vertex set guaranteed by Lemma 4.1 for $\ell = k/4$. Since $m \leq cn$, we have $|A| \leq 24cn/k \leq \varepsilon n$, so indeed F' has at most $(1 + \varepsilon)n$ variables. By Lemma 4.1, $F_{v,A}$ contains at most ℓ non-input gates and, since the fan-in is at most two, $F_{v,A}$ contains most 2ℓ gates overall. So the constraint "$y_v = F_{v,A}(x, y)$" indeed depends on at most $2\ell + 1$ variables and can be expressed trivially by a $(2\ell + 1)$-CNF formula. It is clear that F' can be computed in polynomial time. Moreover, $F(x) = 1$ holds if and only if there is a setting for y such that $F'(x, y) = 1$ holds, so F and F' are equisatisfiable. We obtain the claimed reduction. □

The lemma has the following consequence:

Theorem 4.3. *If SETH is false, then there is an $\varepsilon > 0$ such that, for all c, the satisfiability of Boolean formulas of size at most cn can be solved in time $O((2 - \varepsilon)^n)$.*

In other words, SETH is implied by Sparse-Formula-SETH. This reproves a result of [30].

Proof. Suppose that SETH is false. Then there is some $\delta > 0$ such that k-CNF SAT can be solved in time $O^*((2 - \delta)^n)$ for all $k \in \mathbb{N}$. Let $c > 0$. In order to solve Formula-SAT for cn-size formulas, we apply Lemma 4.2 to reduce to a k-CNF formula with $n' = (1 + \alpha)n$ variables. We can solve this instance using the assumed algorithm in time $(2 - \delta)^{n'} \leq (2 - \delta)^{(1+\alpha)n} \leq (2 - \varepsilon)^n$ for some suitable $\varepsilon, \alpha > 0$. □

After this warm-up, we next consider TC^0-circuits.

4.2 Sparse TC^0-circuits

The goal of this section is to prove the following:

Lemma 4.4. *There is a polynomial-time many-one reduction from TC-SAT to CNF-SAT that, given $\varepsilon \in (0, 1)$ and a depth-d threshold circuit with at most cn wires, with $c \geq 1$, produces a k-CNF formula φ on at most $(1 + \varepsilon)n$ variables and with $k \leq \big(2000(c/\varepsilon)\log(2c/\varepsilon)\big)^d + 1$.*

In the next subsection we will improve the dependence on d by applying depth reduction first. Our proof of Lemma 4.4 relies on a linear-size adder circuit as provided by the following lemma.

Lemma 4.5 (Adder circuit)**.** *Let $b, \ell \in \mathbb{N}$. There is a circuit $C_{add} : \{0,1\}^{b\ell} \to \{0,1\}^{b+\lceil \log \ell \rceil}$ over $\{\mathrm{NEG}, \mathrm{AND}, \mathrm{OR}\}$ with at most $40b\ell$ gates and maximum fan-in 2 such that C_{add} computes the binary representation of the sum of ℓ given b-bit integers.*

Proof. The proof of this lemma is standard. It is well known that there is a circuit C_{FA} (the full adder) that adds b'-bit numbers using $20b'$ gates and constant fan-in. Describing the computation of C_{add} using parentheses, the circuit C_{add} computes the sum $\sum_{i=1}^{\ell} b_i$ in a binary-tree-like way as

$$(((b_1 + b_2) + (b_3 + b_4)) + ((b_5 + b_6) + (b_7 + b_8))) + \dots$$

Using C_{FA} for every addition, the number of gates needed is at most

$$\sum_{i=1}^{\lceil \lg \ell \rceil} 20(b+i)\ell/2^i \leq 20bs \sum_{i=1}^{\infty} i/2^i = 40b\ell \,. \qquad \square$$

We will use the following variant of a threshold gate with binary input.

Lemma 4.6 (BINTH circuit)**.** *Let $r, \theta \in \mathbb{N}$. There is a circuit $\mathrm{BINTH}_\theta : \{0,1\}^r \to \{0,1\}$ over $\{\mathrm{NEG}, \mathrm{AND}, \mathrm{OR}\}$ with at most $2r$ gates and maximum fan-in 2 such that $\mathrm{BINTH}_\theta(x_0, \dots, x_{r-1})$ computes whether $\sum_{i=0}^{r-1} x_i 2^i$ is at least θ.*

Proof. The circuit BINTH_θ is constructed by setting $t = \lceil \lg \theta \rceil$ and converting the following formula into a circuit:

$$\mathrm{BINTH}_\theta(x_0, \dots, x_{r-1}) = \Big(\bigvee_{i=t}^{r-1} x_i \Big) \vee \Big(x_{t-1} \wedge \mathrm{BINTH}_{\theta - 2^{t-1}}(x_0, \dots, x_{t-1}) \Big) \,. \qquad \square$$

We are ready to prove Lemma 4.4.

Proof of Lemma 4.4. Without loss of generality, threshold circuits only have input, NEG, and TH_θ gates, since TH_θ can directly simulate AND, OR, and MAJ gates. The algorithm to transform threshold circuits into k-CNF formulas is implemented in Algorithm C. Intuitively, it replaces threshold gates of large fan-in by a circuit of bounded fan-in in such a way that the circuit can be be simulated by a k-CNF formula without introducing too many new variables.

Algorithm C (*Reduce sparse threshold circuit to k-CNF*) *Given a threshold circuit C of depth d and a positive integer $\beta \in \mathbb{N}$, this algorithm computes an equivalent k-CNF formula F.*

C1 *(Initialize gates to be replaced with variables)* Let $A = \{o\}$ where $o \in V(C)$ is the output gate of C.

C2 *(Replace large threshold gates by the circuit in Figure 2)* For each $v \in V(C)$ of fanin $d_C^-(v) \geq \beta$:

- **a** Let $\theta \in \mathbb{N}$ such that v is a TH_θ-gate.
- **b** Partition the children $N_C^-(v)$ of v into blocks $B_1, \dots, B_\ell$ of size at most β with $\ell \leq \lceil d_C^-(v)/\beta \rceil$ and remove all wires leading into v.
- **c** *(Construct adder circuit for each block)* For each $i \in \{1, \dots, \ell\}$, create a circuit $C_{add}(B_i)$ that uses the gates of B_i as input gates and has $\log \beta + 1$ output gates b_i such that b_i represents the number of 1s in B_i in binary. Here, C_{add} is the circuit from Lemma 4.5.

d *(Simulate threshold gate by circuit of fan-in two)* Add a circuit $\mathrm{BINTH}_\theta(C_{add}(b_1, \ldots, b_\ell))$ with inputs $b_1, \ldots, b_\ell$ and output gate v, that is, the inputs $b_1, \ldots, b_\ell$ are fed into C_{add} (from Lemma 4.5), whose outputs are fed into BINTH_θ (from Lemma 4.6). This concatenated circuit takes the binary representation of ℓ integers $b_1, \ldots, b_\ell \in \{0, 1, \ldots, \beta\}^\ell$ on $b = \log \beta + 1$ bits each and it outputs true if and only if the sum of the given integers is at least θ. The circuit $\mathrm{BINTH}_\theta(C_{add}(b_1, \ldots, b_\ell))$ has at most $40b\ell + 2(b + \lceil \log \ell \rceil) \leq 44b\ell$ gates and fan-in at most 2. We add all of its gates, including the b_i's, to A.

C3 *(Compute k-CNFs)* For all $v \in A$, add a new variable y_v and do the following:

a Let $C_{v,A}$ be the subcircuit of C induced by the set $R_C(A, v)$ (as defined in the preliminaries) and interpret the gates $a \in A$ as input variables y_a.

b We will show that $C_{v,A}$ depends on at most β^d variables, so we can compute a k-CNF formula F_v with at most $k = \beta^d + 1$ variables that expresses the constraint "$y_v = C_{v,A}(x, y)$", where $x_1, \ldots, x_n$ are the input variables of C.

C4 *(Output)* Let $F = y_o \wedge \bigwedge_{v \in A} F_v$, and output F.

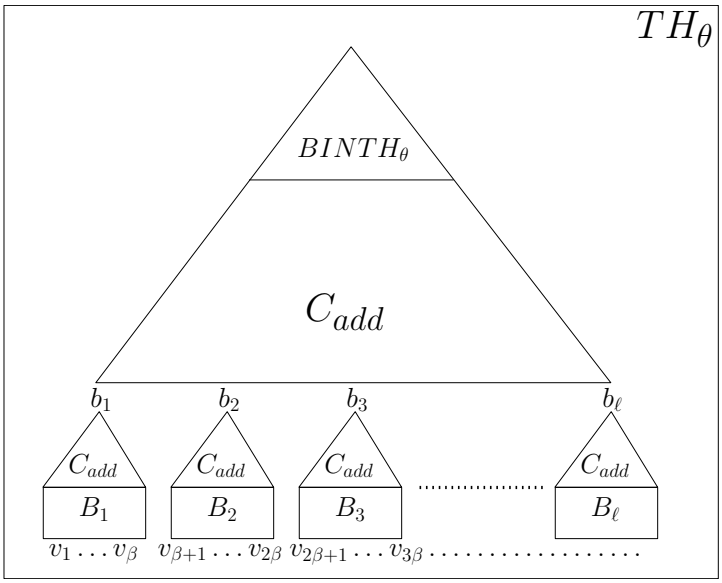

Figure 2: Overview of replacement of TH_θ gate.

Correctness of rewriting To prove correctness, let us first observe that the transformation in C2 does not change the functionality of C since we just explicitly simulate threshold gates with large fan-in by constructing a small Boolean circuit over the De Morgan basis in a straightforward way. We do this transformation in a block-wise fashion in order to save the number of additional variables we add in step C3. So let C be the circuit after its transformation in C2 and consider step C3. Let x be a setting for the n input gates $I(C)$. We claim that $C(x) = 1$ holds if and only if there is a setting for the y-variables such that $F(x, y) = 1$. Indeed, if $C(x) = 1$, we set y such that $y_a = C_a(x)$ holds for all $a \in A$. This setting for y then satisfies the constraint "$y_v = C_{v,A}(x, y)$" and thus $F_v(x, y) = 1$. Moreover, $y_o = C(x) = 1$ holds as well, and so the formula F constructed in C4 is satisfied by (x, y). For the reverse direction, let (x, y) be such that $F(x, y) = 1$. We claim that $C(x) = 1$ holds as

well. Indeed, by construction of F, we know that $y_o = 1$ and $C_{v,A}(x) = y_v$ holds for all $v \in A$. We can see by induction on the depth of v (starting at the bottom) that $C_v(x) = C_{v,A}(x, y)$ holds. In the base case, the only input gates of $C_{v,A}$ are the original x-input gates from $I(C)$, and thus $y_v = C_{v,A}(x, y) = C_v(x)$ holds. In the inductive case, $C_{v,A}$ may depend on variables y_a. However, for each such variable, we know by the induction hypothesis that $y_a = C_{a,A}(x, y) = C_a(x)$ holds, and thus we have $C_v(x) = C_{v,A}(x, y)$ by the definition of C_v. In particular $C(x) = y_o = 1$ holds and so x satisfies C. This establishes the correctness of the reduction, except for proving that width $k = \beta^d + 1$ is sufficient (in step C3b), which we will show next.

Bounding the width In C3a, note that by the definition of $C_{v,A}$ and $R_C(A, v)$ (see Section 2), the value of gate v on input x is determined by the set of values of the gates $u \in A \cap R_C(A, v)$. Therefore, in C3b we can ensure the value y_v equals the value of gate v on input x by adding clauses on y_v and the variables corresponding to the gates in $A \cap R_C(A, v)$. We need to prove that the set $A \cap R_C(A, v)$ has size less than $k = \beta^d + 1$ so that this can be done in k-CNF. For most gates v added to A this is clear because there are only two gates feeding into v after the replacement step C2d and both of these gates are in A as well. The only exceptions are the $b_1, \ldots, b_\ell$-gates. Note that any b_i-gate v is determined by the B_i-gates below it, so we can bound $|A \cap R_C(A, v)| \leq \sum_{u \in B_i} |A \cap R_C(A, u)|$. Any gate $u \in B_i$ already existed in the original circuit. If u has degree at least β in the original circuit, then we ran step C2 on u and thus u belongs to A. Otherwise, u has less than β children, which already existed in the original circuit, and on which we recurse. It follows that if gate u has depth d_u in the original circuit, then it can be reached from less than β^{d_u} nodes in A without going through any other node in A, i.e., $|A \cap R_C(A, u)| < \beta^{d_u}$. Since u is a descendant of v, we have $d_u < d$. In total, we obtain $|A \cap R_C(A, v)| < |B_i| \cdot \beta^{d-1} \leq \beta^d$. It follows that the constraints in step C3b indeed consider at most $k = \beta^d + 1$ variables and can thus be expressed in k-CNF.

Bounding the number of variables It remains to set β in such a way that $|A|$ is at most εn, which implies that F has at most $(1 + \varepsilon)n$ variables. Recall that the loop at C2 iterates over all gates v with fan-in $d_C^-(v) \geq \beta$. For any such gate v, we add at most $50b\ell$ gates to A (see step C2d), where $b = \log \beta + 1$ and $\ell \leq 1 + d_C^-(v)/\beta \leq 2d_C^-(v)/\beta$ since $d_C^-(v) \geq \beta$. Hence, the overall the number of gates ever added to A is at most

$$\sum_{\substack{v \in V(G) \\ d_C^-(v) \geq \beta}} 100\, d_C^-(v) \cdot \frac{\log \beta + 1}{\beta} \leq 100\, cn \frac{\log \beta + 1}{\beta}\,.$$

Thus we can set $\beta = \beta(c, \epsilon) \in \mathbb{N}$ as a function of c and ϵ such that the size of A is at most ϵn. One can check that $\beta \leq 2000(c/\varepsilon) \log(2c/\varepsilon)$ suffices for $\varepsilon \leq 1 \leq c$, where all logs are base 2. Thus, we get the claimed upper bound on k. This concludes the proof of the lemma. □

Let us remark that in Lemma 4.4 we can also handle several other gates, such as MOD_m gates, by replacing the BINTH_θ circuit in the proof with a circuit that checks whether a given integer is a multiple of a given m. In fact, we can handle any symmetric gate $f(x_1, \ldots, x_d) = g(\sum_{i=1}^d x_i)$ where $g(s)$ can be expressed as a $d^{o(1)}$-size DeMorgan circuit when given $s \in \{0, \ldots, d\}$ in binary.

Using Lemma 4.4 with ε such that $(1 + \varepsilon)s_\infty < 1$ we obtain the following.

Theorem 1.5 (restated)**.** *If SETH fails, then there is an $\varepsilon > 0$ such that, for all constants c and d, the satisfiability of depth-d threshold circuits with cn wires can be determined in time $O((2 - \varepsilon)^n)$.*

4.3 Improving the dependence in depth to sub-exponential

In this section we are going to improve the dependence of k on c, ε and d in Lemma 4.4. As this dependence is exponential in d, it is natural to employ existing techniques for depth reduction of circuits, such as the following result due to Valiant [55]. We use a for us more convenient slight variant as stated in [39, Lemma 1.4] (see also [56, Section 4.2]):

Lemma 4.7. *In any directed graph with m edges and depth 2^δ (where δ is integral), it is possible to remove in polynomial time a set R of rm/δ edges so that the depth of the resulting graph does not exceed $2^{\delta - r}$.*

Lemma 4.7 can be directly used to improve the dependence of k in Lemma 4.4:

Lemma 4.8. *There is an algorithm that, given $\varepsilon > 0$ and a depth-d threshold circuit with at most cn wires, produces at most $2^{\varepsilon n/2}$ k-CNF formulas $\varphi_1, \ldots, \varphi_z$ on $(1+\varepsilon/2)n$ variables such that the circuit C is satisfiable if and only if φ_i is satisfiable for some $i \leq z$, and we have $k \leq (4000(c/\varepsilon)\lg(4c/\varepsilon))^{(2d)^{1-\varepsilon/(2c)}} + 1$.*

Proof. Let $C = (G = (V, E), \lambda)$ be the given circuit. Apply Lemma 4.7 to G with $\delta = \lceil \lg d \rceil$ and $r = \varepsilon\delta/(2c)$. We obtain a set R of size at most $\varepsilon n/2$ such that $(V, E \setminus R)$ has depth at most $2^{\delta(1-\varepsilon/(2c))} \leq (2d)^{1-\varepsilon/(2c)}$. For every assignment $a \in \{0,1\}^R$ we create a circuit where we require that $C_x(v) = a_v$ for every $v \in R$, remove the outgoing wires and update their incident gate accordingly (i.e. if $a_v = 1$ and the incident gate is a TH_θ it becomes a $\mathrm{TH}_{\theta-1}$ gate). The obtained circuit has depth at most $(2d)^{1-\varepsilon/(2c)}$ and applying Lemma 4.4 gives the claimed result. □

The improved dependence of k in Lemma 4.8 allows for the following consequence, yielding an exponential speedup for sparse threshold circuits of any depth $(\log n)^{1+o(1)}$.

Theorem 1.6 (restated)**.** *If CNF-SAT can be solved in $O^*(2^{(1-\varepsilon')n})$ time for some $\varepsilon' > 0$, then there is an $\varepsilon > 0$ such that for any $c > 0$ there is a $\delta > 0$ such that the satisfiability of threshold circuits with depth $(\log n)^{1+\delta}$ and at most cn wires can be determined in time $O(2^{(1-\varepsilon)n})$.*

Proof. Apply Lemma 4.8 with $\varepsilon/2$ to obtain $2^{\varepsilon n/2}$ k-CNF-formulas with $k = 2^{O((\lg n)^{(1+\delta)(1-\varepsilon/(2c))})}$ on $(1+\varepsilon/2)$ variables. For sufficiently small $\delta = \delta(\varepsilon, c) > 0$ we have $k = 2^{o(\log n)} = o(n/\log n)$ and thus the number of clauses m is at most $(2n)^k = n^{o(n/\log n)} = 2^{o(n)}$. Therefore the assumed algorithm for CNF-SAT determines the satisfiability of the produced CNF formula in time $2^{\varepsilon n/2} \cdot m^{O(1)} \cdot 2^{(1-\varepsilon)(1+\varepsilon/2)n}$, which is $O(2^{(1-\varepsilon')n})$ for $\varepsilon' < \varepsilon^2/2$. □

Open Question. It is known that Lemma 4.7 cannot be significantly improved (see [54]). However, this does not stop us from using the power of branching to get improvements. Specifically, when we try an assignments of the truth-value on edges in R in Lemma 4.4, all gates that are not connected to inputs are constant so these and their wires can already be computed and removed from the circuit. A natural question is whether this can be exploited more: Given a DAG $G = (V, E)$ of depth 2^δ on m edges and a real number $0 < \alpha < 0$. For a set R of edges, denote $l(R)$ as the length of the longest path in $(V, E \setminus R)$ starting at a vertex v which is a source in G. Give an upper bound on $\min\{l(R) : |R| \leq \varepsilon m\}$ better than $2^{(1-\varepsilon)\delta}$ (which is implied by Lemma 4.7).

References

[1] Amir Abboud, Arturs Backurs, Karl Bringmann, and Marvin Kunnemann. Fine-grained complexity of analyzing compressed data: Quantifying improvements over decompress-and-solve. In *FOCS, to appear*, 2017.

[2] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *Proc. of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 59–78, 2015.

[3] Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1681–1697. SIAM, 2015.

[4] Amir Abboud, Kevin Lewi, and Ryan Williams. Losing weight by gaining edges. In *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, pages 1–12, 2014.

[5] Amir Abboud, Aviad Rubinstein, and Ryan Williams. Distributed pcp theorems for hardness of approximation in p. In *FOCS, to appear*, 2017.

[6] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proc. of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 434–443, 2014.

[7] Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *Proc. of the 47th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 41–50, 2015.

[8] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Proc. of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 39–51, 2014.

[9] Arturs Backurs, Nishanth Dikkala, and Christos Tzamos. Tight hardness results for maximum weight rectangles. *CoRR*, abs/1602.05837, 2016.

[10] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proc. of the 47th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 51–58, 2015.

[11] Arturs Backurs and Piotr Indyk. Which regular expression patterns are hard to match? In *Proc. of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 457–466, 2016.

[12] Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. On the fine-grained complexity of empirical risk minimization: Kernel methods and neural networks. In *NIPS, to appear*, 2017.

[13] Arturs Backurs and Christos Tzamos. Improving Viterbi is hard: Better runtimes imply faster clique algorithms. *CoRR*, abs/1607.04229, 2016.

[14] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In *Proc. of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC), to appear*, 2017.

[15] Christopher Beck and Russell Impagliazzo. Strong ETH holds for regular resolution. In *Proc. of the 45th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 487–494, 2013.

[16] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[17] Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 661–670, 2014.

[18] Karl Bringmann, Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Tree edit distance cannot be computed in strongly subcubic time (unless APSP can). *CoRR*, abs/1703.08940, 2017.

[19] Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. A dichotomy for regular expression membership testing. *CoRR*, abs/1611.00918, 2016.

[20] Karl Bringmann and Marvin Kunnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proc. of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 79–97, 2015.

[21] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A duality between clause width and clause density for SAT. In *Proc. of 21st Conference on Computational Complexity (CCC)*, pages 252–260, 2006.

[22] Timothy M. Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1246–1255, 2016.

[23] Ruiwen Chen and Rahul Santhanam. Improved algorithms for sparse MAX-SAT and MAX-k-CSP. In *Theory and Applications of Satisfiability Testing - SAT 2015 - 18th International Conference, Austin, TX, USA, September 24-27, 2015, Proceedings*, pages 33–45, 2015.

[24] Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3):41:1–41:24, 2016.

[25] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[26] Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast Hamiltonicity checking via bases of perfect matchings. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 301–310. ACM, 2013.

[27] Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. On problems equivalent to (min, +)-convolution. In *ICALP*, pages 22:1–22:15, 2017.

[28] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159. IEEE Computer Society, 2011.

[29] Søren Dahlgaard. On the hardness of partially dynamic graph problems and connections to diameter. *arXiv preprint arXiv:1602.06705*, 2016.

[30] Evgeny Dantsin and Alexander Wolpert. Exponential complexity of satisfiability testing for linear-size boolean formulas. In *CIAC*, volume 7878 of *LNCS*, pages 110–121, 2013.

[31] Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1):57–67, 2004.

[32] Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995.

[33] François Le Gall. Faster algorithms for rectangular matrix multiplication. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 514–523. IEEE Computer Society, 2012.

[34] Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and Ryan Williams. Completeness for first-order properties on sparse structures with algorithmic applications. In *Proc. of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2162–2181, 2017.

[35] Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for AC^0. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 961–972, 2012.

[36] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[37] Russell Impagliazzo, Ramamohan Paturi, and Stefan Schneider. A satisfiability algorithm for sparse depth two threshold circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 479–488, 2013.

[38] Hamid Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. In *Proc. of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 749–760, 2015.

[39] Stasys Jukna. *Boolean Function Complexity: Advances and Frontiers*. Springer Publishing Company, Incorporated, 2012.

[40] Daniel Kane and Ryan Williams. The orthogonal vectors conjecture for branching programs and formulas. *arXiv preprint arXiv:1709.05294*, 2017.

[41] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In *Proc. of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1272–1287, 2016.

[42] Robert Krauthgamer and Ohad Trabelsi. Conditional lower bounds for all-pairs max-flow. *arXiv preprint arXiv:1702.05805*, 2017.

[43] Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the fine-grained complexity of one-dimensional dynamic programming. *arXiv preprint arXiv:1703.00941*, 2017.

[44] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs on bounded treewidth are probably optimal. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 777–789. SIAM, 2011.

[45] Daniel Moeller, Ramamohan Paturi, and Stefan Schneider. Subquadratic algorithms for succinct stable matching. In *International Computer Science Symposium in Russia*, pages 294–308. Springer, 2016.

[46] Jesper Nederlof, Erik Jan van Leeuwen, and Ruben van der Zwaan. Reducing a target interval to a few exact queries. In Branislav Rovan, Vladimiro Sassone, and Peter Widmayer, editors, *Mathematical Foundations of Computer Science 2012 - 37th International Symposium, MFCS 2012, Bratislava, Slovakia, August 27-31, 2012. Proceedings*, volume 7464 of *Lecture Notes in Computer Science*, pages 718–727. Springer, 2012.

[47] Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Math. Universitatis Carolinae*, 26(2):415–419, 1985.

[48] Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1065–1075. SIAM, 2010.

[49] Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *Proc. of the 42nd Annual ACM Symposium on Theory Of Computing (STOC)*, pages 603–610, 2010.

[50] Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proc. of the 45th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 515–524, 2013.

[51] Liam Roditty and Uri Zwick. On dynamic shortest paths problems. In *Proc. of the 12th ESA*, pages 580–591, 2004.

[52] Barna Saha. Language edit distance and maximum likelihood parsing of stochastic grammars: Faster algorithms and connection to fundamental graph problems. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 118–135. IEEE, 2015.

[53] Rahul Santhanam and Srikanth Srinivasan. On the limits of sparsification. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, pages 774–785, 2012.

[54] Georg Schnitger. A family of graphs with expensive depth-reduction. *Theoretical Computer Science*, 18(1):89 – 93, 1982.

[55] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *Mathematical Foundations of Computer Science 1977, 6th Symposium, Tatranska Lomnica, Czechoslovakia, September 5-9, 1977, Proceedings*, pages 162–176, 1977.

[56] Emanuele Viola. On the power of small-depth computation. *Foundations and Trends in Theoretical Computer Science*, 5(1):1–72, 2009.

[57] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2–3):357–365, 2005.

[58] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013.

[59] Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 664–673, 2014.

[60] Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014.

[61] Virginia Vassilevska Williams. Hardness of easy problems: basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 43, 2015.

[62] Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *Proc. of the 41st Annual ACM Symposium on Theory Of Computing (STOC)*, pages 455–464, 2009.

[63] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 645–654. IEEE Computer Society, 2010.

A Schematic Overview of Our results

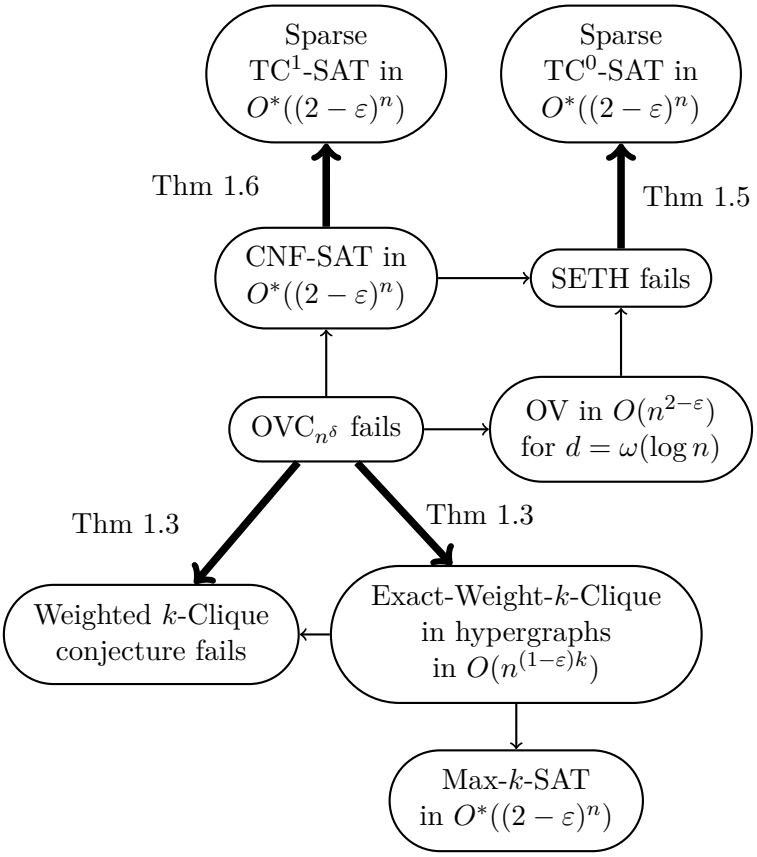

Figure 3: An overview of the relevant implications. New implications presented in this paper are displayed with bold arcs and labeled with the appropriate theorem statement.