More Consequences of Falsifying SETH and the Orthogonal Vectors Conjecture

Amir Abboud
IBM Almaden Research Center
San Jose, CA, USA
amir.abboud@ibm.com

Holger Dell
Saarland University
Cluster of Excellence (MMCI)
Saarbrücken, Germany
hdell@mmci.uni-saarland.de

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

Jesper Nederlof
Eindhoven University of Technology
Eindhoven, The Netherlands
j.nederlof@tue.nl

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 $\epsilon > 0$ for which an $O(N^{2-\epsilon})$ 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 $\epsilon > 0$ such that: (1) For all $d$ and all large enough $k$, there is a randomized algorithm that takes $O(n(1-\epsilon)^k)$ time to solve the Zero-Weight-k-Clique and Min-Weight-k-Clique problems on $d$-hypergraphs with $n$ vertices. As a consequence, the OV-conjecture is implied by the Weighted Clique conjecture. (2) For all $c$, the satisfiability of sparse TC$^1$ circuits on $n$ inputs (that is, circuits with $cn$ wires, depth $c log n$, and negation, AND, OR, and threshold gates) can be computed in time $O((2-\epsilon)^n)$.

CCS CONCEPTS
- Theory of computation → Problems, reductions and completeness;

KEYWORDS
fine-grained complexity, OV, clique, satisfiability, threshold circuits

1 INTRODUCTION

The Strong Exponential Time Hypothesis (SETH) is a cornerstone of contemporary algorithm design that was formulated by Impagliazzo and Paturi [34] and recently gained extensive popularity. It postulates that exhaustive search is essentially the fastest possible method to decide the satisfiability of bounded-width CNF formulas. SETH is used in the study of exact and fixed parameter tractable algorithms, see e.g. [23, 46] or the book by Cygan et al. [24]. In this area, it implies, among other things, tight lower bounds for problems on graphs that have small treewidth or pathwidth [25, 26, 41].

Closely related to SETH, the orthogonal vectors problem (OV) 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 $\mathbb{Z}^D$. If $D \leq O(N^{3.3})$ holds, the problem can be solved in time $\tilde{O}(N^2)$ using an algorithm based on fast rectangular matrix multiplication (see e.g. [31]). SETH implies [54] that this algorithm is essentially as fast as possible; in particular, SETH implies the following hardness conjecture, which was given its name by Gao et al. [32].

Conjecture 1.1 (Moderate-dimension OV Conjecture). There are no reals $\epsilon, \delta > 0$ such that $OV$ for $D = N^\delta$ can be solved$^4$ in time $\tilde{O}(N^{2-\epsilon})$.

The moderate-dimension OV conjecture is used to study the fine-grained complexity of problems in P, for which it has remarkably strong and diverse implications. If the conjecture is true, then dozens of important problems from all across computer science exhibit running time lower bounds that match existing upper bounds up to subpolynomial factors. These include pattern matching and other problems in bioinformatics [1, 7, 10, 40], graph

---

$^4$In this work we hardly distinguish between randomized and deterministic algorithms, as even randomized algorithms with the desired running times would constitute an important breakthrough.
algorithms [6, 32, 47], computational geometry [16], formal languages [11, 18], time-series analysis [2, 19], and even economics [42] (see [58] for a more comprehensive list).

Gao et al. [32] also named the low-dimension OV conjecture, which asserts that OV does not have subquadratic algorithms whenever \( D = \omega(\log n) \) holds. The low-dimension implies the moderate-dimension variant of the OV conjecture, and both are implied by SETH [54]. Recent results on the hardness of approximation problems, such as Maximum Inner Product [5], rely on the stronger conjecture (perhaps also [12, 14]). However, for the vast majority of OV-based hardness results, reducing the dimension only affects lower-order terms in the lower bounds and so it often suffices to assume the moderate-dimension variant. Doing so makes results stronger, and it is this variant of the OV conjecture that we strengthen further in the present work.

1.1 Other Conjectures

Two other popular conjectures in fine-grained complexity make assertions for the All-Pairs-Shortest-Path Problem (APSP) and the 3-SUM problem. It is an important and longstanding open question to determine the relationship between APSP, 3-SUM, and OV. In particular, it is open whether the APSP conjecture or the 3-SUM conjecture imply the moderate-dimension OV conjecture.

Closely related to APSP is the Min-Weight-k-Clique problem: Given a graph on \( n \) nodes with integer edge-weights in some polynomial range, the goal is to find a \( k \)-clique of minimum weight. The exhaustive search algorithm solves this problem in \( O(n^k) \) time, and for \( k = 3 \), the problem is subcubically-equivalent to the APSP conjecture [59]. That is, either both APSP and Min-Weight-3-Clique have algorithms running in time \( O(n^{3-\epsilon}) \) for some \( \epsilon > 0 \), or neither of them do.

For all integers \( k \geq 3 \), there is a simple reduction from Min-Weight-k-Clique to Min-Weight-3-Clique, and by combining it with the fastest known APSP algorithm [21, 56], we can solve Min-Weight-k-Clique in time \( n^k/\exp(\Omega(\sqrt{\log n})) \). The improvement of this algorithm over exhaustive search is subpolynomial. Due to the equivalence with APSP for \( k = 3 \), it is natural to conjecture that a truly polynomial advantage in the running time is impossible.

**Conjecture 1.2 (Weighted Clique Conjecture).** There is no real \( \epsilon > 0 \) and integer \( k \geq 3 \) such that the Min-Weight-k-Clique problem on \( n \)-vertex graphs and with edge-weights in \( \{-M, \ldots, M\} \) can be solved in time \( O(n^{1-\epsilon}k) \) polylog M.

This conjecture implies the APSP conjecture, and thus implies lower bounds for all problems that are known to be APSP-hard [3, 6, 8, 27, 48, 49, 60]. In addition, the Weighted Clique conjecture implies lower bounds for a variety of problems that are not known to be APSP-hard: the Local Alignment problem [7] from bioinformatics, the Maximum Rectangle problem [9] from computational geometry, the Viterbi problem [13] from machine learning and, the Tree Edit Distance problem [17].

1.2 Our Results for OV

We prove that the Weighted Clique conjecture (Conjecture 1.2) implies the moderate-dimension OV conjecture (Conjecture 1.1). To this end, we design a tight randomized reduction from Min-Weight-k-Clique to OV. The impact of this result on fine-grained complexity is depicted in Figure 1. As can be seen, we identify Min-Weight-k-Clique as a core problem in this landscape, since it tightly reduces to most problems that have known conditional lower bounds; the main exceptions are 3-SUM-hard problems and the few problems that do require the low-dimension version of the OV conjecture.

In fact, our result is even stronger: We show that improved algorithms for moderate-dimension OV 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 \( k \)-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 \( e \in E(G) \). The \( k \)-Clique problem is given \( G \) as input to find such a \( k \)-Clique of size \( k \). We also study weighted versions, where we are additionally given an edge-weight function \( w : E(G) \rightarrow \mathbb{Z} \) and a target integer \( t \in \mathbb{Z} \). The weight of a \( k \)-Clique in \( G \) is the sum \( \sum e \cdot w(e) \) over all edges \( e \in E(G) \) with \( e \subseteq X \). In the Exact-Weight-k-Clique problem, we need to find a \( k \)-Clique of weight exactly \( t \), and in Min-Weight-k-Clique we need one of weight at most \( t \). We are ready to formally state our first theorem.

**Theorem 1.3.** If the moderate-dimension OV conjecture is false, then there exists an \( \epsilon > 0 \) such that for every integer \( d \) there is a (large) integer \( k = k(d, \epsilon) \) satisfying the following statements:

- **k-Clique** can be solved on \( d \)-hypergraphs in time \( O(n^{1-\epsilon}k) \).
- **Exact-Weight-k-Clique** and **Min-Weight-k-Clique** can be solved on \( d \)-hypergraphs with weights in \( \{-M, \ldots, M\} \) in randomized time \( O(n^{1-\epsilon}k) \cdot \text{polylog } M \).

Clique problems on hypergraphs appear to be harder than on graphs. For example, in the unweighted case, \( k \)-Clique on graphs can be solved in \( O(n^{d/2}) \) time [29, 44], whereas on 3-hypergraphs no polynomial improvement over the \( O(n^k) \) exhaustive search algorithm is known. If the moderate-dimension OV conjecture is false, then Theorem 1.3 implies that \( k \)-Clique on 3-hypergraphs does have such an improvement for some large enough integer \( k \). This strengthens the OV conjecture.

Significantly improved algorithms for \( k \)-Clique on \( d \)-hypergraphs are not only unknown, they are in fact known to imply breakthrough algorithms for Max-\( d \)-SAT, the optimization version of the \( d \)-CNF-SAT problem that needs to find an assignment that satisfies as many clauses as possible. Using this known reduction, we obtain the following corollary to Theorem 1.3.

**Corollary 1.4.** If the moderate-dimension OV conjecture is false, then there exists an \( \epsilon > 0 \) such that, for all integers \( d \), there is an algorithm for Max-\( d \)-SAT that runs in time \( O^*(2^{\epsilon d}) \).

Due to this corollary, the moderate-dimension OV conjecture is not only implied by SETH, but even by its stronger cousin for the Max-SAT problem on bounded-width CNF. As a consequence, all
known lower bounds based on the moderate-dimension OV conjecture are now automatically based on the hardness of bounded-width Max-SAT rather than just bounded-width CNF-SAT. Corollary 1.4 subsumes some results [2, 8, 39] were this was done in special cases.

The implications of Theorem 1.3 for the weighted problems strengthen the moderate-dimension OV conjecture further. Any algorithm that solves Exact-Weight-$k$-Clique can in particular solve Zero-Weight-$k$-Clique where the target weight satisfies $t = 0$. In turn, any algorithm for the latter problem can also be used to solve Min-Weight-$k$-Clique without significant running time overhead [45]. While the best known algorithms for Min-Weight-$k$-Clique run in time $n^k/\exp(\Omega(\log n))$ [21, 56], such superpolylogarithmic shavings are open for Zero-Weight-$k$-Clique. The $k = 3$ case is particularly interesting, since solving Zero-Weight-3-Clique in $O(n^{3\gamma})$ time refutes not only the APSP conjecture but also the 3-SUM conjecture [38, 45, 60].

**Proof ideas.** We prove Theorem 1.3 by designing a tight reduction from Min-Weight-$k$-Clique on $d$-hypergraphs to OV. We sketch the reduction for $d = 2$. It has two main stages. In the first, we reduce Min-Weight-$k$-Clique on graphs to unweighted $k$-Clique on 4-hypergraphs, where each hyperedge has cardinality at most 4. We achieve this in a sequence of 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 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 instead use it to reduce edge weights, which however we only achieve by transforming the graph into a 4-hypergraph. Finally, once the weights are small enough, we remove them completely via an exhaustive search.

In the second stage, we reduce the unweighted $k$-Clique problem on 4-hypergraphs to a $k$-wise variant of OV. The reduction maps each node to a Boolean vector by encoding the incident hyperedges into the coordinates such that a disjointness check among $k$ vectors corresponds to checking that $k$ nodes form a hyperclique. Finally, we reduce the $k$-wise variant to the OV problem using a standard reduction.

### 1.3 Our Results for SETH and CNF-SAT

There are many algorithms that solve $d$-CNF-SAT, the satisfiability problem on $d$-CNF formulas, in time $O^*((2 - \varepsilon)'^d)$. As $d$ grows to infinity, the constants $\varepsilon_d$ for all these algorithms tend to 0 in the limit. SETH was conceived by this observation, and it asserts exactly this: If $\varepsilon_d$ is the largest real such that $d$-CNF-SAT can be solved in time $O^*((2 - \varepsilon)'^d)$, then $\lim_{d\to\infty} \varepsilon_d = 0$ holds. Thus SETH is not about the hardness of an individual problem, but about a sequence of problems each of which we know to have a faster algorithm than exhaustive search. This makes it easier prove lower bounds under SETH, but it is unfortunate if we want to have confidence that SETH and the implied lower bounds are true.

Indeed, there are algorithms that get substantial $n^{o(1)}$ speed-ups over $2^n$ for CNF formulas of unbounded width (see e.g. [15, 20]). If instead of CNF formulas, we consider more complex Boolean circuits, such as bounded-depth threshold circuits ($TC^0$-circuits), then we should get a computationally harder satisfiability problem. For linear-size threshold circuits of depth two, there are satisfiability algorithms that run in time $O^*((2 - \varepsilon)'^{c_2})$; here too, the constants $\varepsilon_c$ tend to 0 as $c$ grows [22, 35]. However, even for linear-size threshold circuits of depth 4, satisfiability algorithms with speed-up $n^{o(1)}$ are unknown. Obtaining such algorithms for linear-size $TC^0$ would resolve Williams’ question [57] of whether his circuit lower bound framework can prove $\text{NEXP} \not\subseteq TC^0$. We show that a refutation of SETH would constitute progress on 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 theorem is the newest member in a sequence of increasingly general results of Santhanam and Srinivasan [50], Dantsin and Wolpert [28], and Cygan et al. [23], who show that refuting SETH implies faster algorithms for the satisfiability of linear-size formulas,
linear-size $AC^0$-circuits, and linear-size VSP-circuits, respectively. Our class of linear-size TC\(^0\)-circuits contains the classes of linear-size formulas and $AC^0$-circuits, so we generalize these two results. The class VSP is less understood and little is known about its complexity properties. While algorithms with running time $2^n/n^{o(1)}$ are known for the satisfiability of linear-size $AC^0$-circuits, it is an open question whether algorithms are known for linear-size TC\(^0\)-circuits, even when the depth is 4 and the number of wires is 10n.

As usual with reductions, a pious believer is biased to view Theorem 1.5 as another confirmation that SETH and all its logical implications are indeed true, which includes the moderate-dimension OV conjecture. On the other hand, a skeptic who is resistant to believe SETH or one of its implications is now invited to start their refutation attempt by providing faster algorithms for linear-size TC\(^0\)-circuits, since any refutation of SETH would have to do that implicitly.

**Extension for CNF-SAT.** Much like most OV-based lower bounds in P can be based on the moderate-dimension OV conjecture rather than its low-dimensional variant, many SETH-based lower bounds for exponential time and parameterized problems can be based on the weaker assumption that satisfiability cannot be solved in time $O^*(2^{1−ε}n)$ for CNF formulas of unbounded width. This weaker assumption for CNF-SAT suffices, for example, in results for graph problems that have small treewidth or pathwidth [25, 26, 41]. We add further weight to these hardness results by showing that sufficiently fast algorithms for CNF-SAT imply improved satisfiability algorithms for linear-size threshold circuits of super-logarithmic depth, which is a larger class than TC\(^1\)-circuits.

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

If SETH is false, then not every problem in NEXP can be computed by linear-size VSP-circuits [36, 55]. From Theorem 1.6, it follows that solving CNF-SAT in time $O^*((2−ε)n)$ implies the perhaps more natural result that not every problem in NEXP has linear-size TC\(^1\)-circuits.

**Proof idea.** Similar to the analogous result by Cygan et al. [23] for VSP-circuits, we use a depth reduction technique introduced by Valiant [52], which shows that VSP-circuits embed nicely into CNF-formulas. We use an additional trick that allows us to get rid of threshold gates.

## 2 PRELIMINARIES

**Notation.** The $O^*(\cdot)$ and $\tilde{O}(\cdot)$ notations omit factors that are polynomial and polylogarithmic in the input size, respectively. We write $Z$ for the integers and $N$ for the non-negative integers. We let $[n] = \{1, \ldots, n\}$ for $n \in N$. If $S$ is a set, we write $[S]$ for the set of all subsets of $S$ that have size exactly $d$, and $\binom{S}{d}$ for the set of all subsets of size at most $d$. A d-hypergraph $G$ for $d \in N$ is a tuple $(V(G), E(G))$, where $V(G)$ is a finite set of vertices and $E(G) \subseteq \binom{V(G)}{d}$ 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 $E(G[X]) = E(G) \cap \binom{X}{d}$. A set $S \subseteq V(G)$ is called a clique in $G$ if $E(G[S]) = \binom{S}{d}$. A $k$-clique is a clique of size $k$.

A graph is a 2-hypergraph. In contrast to usual graph notation, there are also edges $\{v\}$ of size 1 and the edge $\emptyset$ of size 0; a $k$-clique is a set $\{v_1, \ldots, v_k\}$ for which every pair $\{v_i, v_j\}$ is in $E(G)$, every singleton $\{v_i\}$ is in $E(G)$, and $\emptyset \in E(G)$. This does not significantly change the problem of detecting whether $G$ contains a $k$-clique, since testing whether $\emptyset \in E(G)$ is in constant time, and we can assume without loss of generality that $\{v\} \in E(G)$ for all $v \in V(G)$, by deleting all other vertices.

**CNF-SAT.** The $d$-SAT problem is to determine whether a given $d$-CNF formula has a satisfying assignment. We denote the number of variables by $n$ and define $s_d$ as the real number

$$\inf\{\delta > 0 : \text{there is an } O(2^{\delta n}) \text{ time algorithm for } d\text{-SAT}\}.$$

Let $s_{\text{nec}} = \lim_{d \to \infty} s_d$. Impagliazzo and Paturi’s Strong Exponential Time Hypothesis (SETH) postulates that $s_{\text{nec}} \approx 1$ holds [34].

**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 $d_G^-(v)$ denote the in-degree with $d_G^-(v) = |N_G^-(v)|$. The depth of $G$ is the length of the longest directed path in it.

A **Boolean function** is any function $f : \{0, 1\}^d \to \{0, 1\}$. 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 over $B$ is a directed graph $G$ with $G$ 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 $\lambda_v$-gate, and we require that the in-degree of $v$ is equal to the arity of $\lambda_v$, 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 $V(G)$ for $V(G)$. The set of input gates $I(G)$ or $I(G)$ of $C$ consists of the vertices with in-degree 0, and the set of output gates $O(G)$ 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 $C_x(v)$ as the value of $C$ at $v \in V$ on input $x$ inductively: If $v \in I(G)$, let $C_x(v) = x_v$, and otherwise, let $C_x(v) = \lambda_v(C_t(v), \ldots, C_s(v))$, where $v_1, \ldots, v_k$ denotes the in-neighbors of $v$ in $G$; note that this is well-defined since $G$ is acyclic and $\lambda_v$ is symmetric. Slightly abusing notation, we may write $C$ also for the function $C : \{0, 1\}^{I(G)} \to \{0, 1\}^{O(G)}$ with $C(x) = (C_x)_v \circ v \in O(G)$. 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 $|\log r|$ gates.

A $(u, v)$-path in $C$ is a directed path $u_1, \ldots, u_r$ in $G$ with $u = u_1$ and $u_r = v$. If $A \subseteq V(G)$, we let $RC(A, v)$ denote the set of vertices

We remark that there is an alternative definition of hypergraphs and cliques, where each edge of a $d$-hypergraph has size exactly $d$ instead of at most $d$, and a clique is a set $S$ such that $E(G[S]) = \binom{S}{d}$. The two variants are equivalent in terms of the algorithmic problem of deciding whether $G$ contains a $k$-clique. Indeed, if we want to detect a set $S$ with $E(G[S]) = \binom{S}{d}$, then we can add all sets of size at most $d - 1$ to $E(G)$ and then detect a set $S$ with $E(G[S]) = \binom{S}{d}$. Similarly, if we want to detect a set $S$ with $E(G[S]) = \binom{S}{d}$, then we can build a new hypergraph $G' = (V(G), E')$ where $E'$ contains all sets $e \subseteq V(G)$ of size $d$ whose every subset is in $E(G)$, and then detect a set $S$ with $E(G'[S]) = \binom{S}{d}$. Similar equivalences hold for the weighted variants of the $k$-Clique problem. Thus, our choice of a variant is only for notational convenience.
We rely on some basic reductions: The first turns the hypergraph $M$ variables, at most $d$ reductions for Weighted $k$-Clique problem, 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 to 0 by using negative edge weights; the fourth uses a non-negative target value but removes negative weights. In the following statement, $M'$ denotes the maximum weight $M(w', t')$ of the respective output instance.

**FACT 3.1.** Let $d, k, n \in \mathbb{N}$ 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 $(G', w', k, t')$ with $V(G') = V(G)$, $E(G') = \{ (x, y) : x, y \in V(G) \}$, and $M' \leq \binom{k}{d} M$.
2. "Make $k$-partite": maps an instance $(G, w, k, t)$ to $(G', w', k, t)$ with $|V(G')| \leq k|V(G)|$ and $M = M'$, such that $G'$ is $k$-partite in the sense that $V(G')$ is partitioned into $k$ parts and every edge intersects each part in at most one vertex.

3. "Make target zero": maps a $k$-partite instance $(G, w, k, t')$ to $(G', w', k, t')$ with $t' = 0$ and $M' \leq 2M$.
4. "Make weights non-negative": maps an instance $(G, w, k, t)$ to $(G', w', k, t')$ with $w' : E(G) \to \mathbb{N}$ and $M' \leq 2^k \binom{d}{d} M$.

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

For the second claim, we define $V(G') = \{ 1, \ldots, k \} \times V(G)$. For every pairwise distinct $a_1, \ldots, a_d \in \{ 1, \ldots, k \}$ and every edge $\{v_1, \ldots, v_d\} \in E(G)$ of size $d'$, we add an edge $f$ with $f = \{ (a_1, v_1), \ldots, (a_d, v_d) \} \subseteq V(G')$ to $G'$. We set the weight $w'(f) = w(\{ v_1, \ldots, v_d \})$. 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, we start with the construction used in the second claim. For any edge of cardinality $d$, denoted by $f = \{ (a_1, v_1), \ldots, (a_d, v_d) \}$, we set $w'(f) = w(\{ v_1, \ldots, v_d \})$ if $|\{a_1, \ldots, a_d\} \cap \{1, \ldots, d\}| = 1$ and $w'(f) = w(\{ v_1, \ldots, v_d \}) - t$ if $|\{a_1, \ldots, a_d\} \cap \{1, \ldots, d\}| > 1$. 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 $E(G) = \{ V(G) \}$ using the first claim, which increases $M$ by at most a factor $\binom{k}{d}$. Let $L = \max \{ 0, w(e) : e \notin E(G) \}$, 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 $t' = t + L \binom{k}{d}$. If $t' < 0$ or $t' > \max \{ w(e) : \binom{k}{d} \}$, the instance is a trivial no-instance. Otherwise the reduction outputs $(G, w', k, t')$. $\square$

### 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 reduce the maximum weight from $M$ to $n^\Omega(k)$ in the following way.

**LEMMA 3.2.** Let $d, k, n \in \mathbb{N}$ with $1 \leq d \leq k$. For some constant $f(k, d) \in \mathbb{N}$ there is a randomized $f(k, d) \cdot \log M$ time self-reduction for the Weighted $d$-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 $w' : E(G) \to \mathbb{N}$, $t' \in N$, $M' \leq f(k, d) \cdot \log M$, 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 $M \geq n^\Omega(k)$ holds, then in time $O(n^\Omega(k) \log M) = \log M$ we brute-force the problem. In the remaining case, we sample a prime $p$ uniformly at random from a range specified later, and set $w'(e) = w(e) \mod p$ for all $e$. We query the oracle $(G, w, k, t')$ for all $t'$ with $t' = 0 + t \mod p$ and $j \in \{ 0, \ldots, \binom{k}{d} \}$, 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 $\sum_{e \in E(G[S])}(w(e) \mod p) = jp + (\sum_{e \in E(G[S])}w(e)) \mod p = jp + (t \mod p) = jp + t'$ holds for some $j$ in the specified range since $(\frac{k}{S})$ 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 $Q = 200n^k \log(k^4M)$ 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 $O(polylog Q) = O(polylog M)$ 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 $k^4M$ 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 $k^4M$, it has at most $\log(k^4M)$ prime divisors. Therefore the probability that for some $S$ it holds that $w(S)$ is congruent to $t$ modulo $p$ is at most $n^k \log(k^4M)/Q = 1/200$. Overall, we succeed at finding a prime with the desired property with probability (99.5%)$^k \geq 99\%$.

We indeed make at most $k^4$ queries to the oracle, and the largest weight in each query is bounded by $(\frac{k}{S})$. Since $p$ is bounded by $Q$ and $M \leq \exp(n^k)$, this is at most $\sum(k,d)n^{O(k)}$. 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 $O(dk \log n + \log M) = polylog M$ bits. Computing the weights modulo $p$ can be done in time $polylog M$. □

### 3.3 Weight Reduction: From Polynomial to Unweighted

We reduce the weights from $n^{O(k)} \cdot f(k,d) \cdot \log n$ using a deterministic argument, and then use exhaustive search to reduce to the unweighted case.

The $q$-expansion of a number $N \in \mathbb{N}$ is the unique sequence $N_0, N_1, \ldots \in \{0, \ldots, q-1\}$ with $N = \sum_{i \in \mathbb{N}} N_i q^i$. After applying the $q$-expansion, all $N_i$ are bounded by $q-1$. However, the smaller we choose $q$, the longer the relevant part of the encoding of $N$ gets; the precise length of this encoding is $p = \lfloor \log_q(N + 1) \rfloor$. The following lemma uses carries to split the weight constraint along the $q$-expansions of the edge weights $w$ and the target $t$.

**Lemma 3.3.** Let $G$ be a $d$-hypergraph with edge-weight function $w : E(G) \rightarrow \mathbb{N}$ and a set $S \subseteq V(G)$ with $|S| = k$. Let $t, q, p \in \mathbb{N}$ with $q \geq 2$ and $p = \log_q(t+1)$. The following are equivalent:

i. (Sum has target value.) We have $t = \sum_{e \in E(G[S])}w(e)$.

ii. (Expansions and carries satisfy linear constraints.) There is a sequence $c_0, c_1, \ldots \in \{0, \ldots, 2\frac{k}{S}\}$ such that $c_0 = 0$ and the following linear equations hold for all $\ell \in \mathbb{N}$:

$$qc_{\ell+1} + t\ell = c\ell + \sum_{e \in E(G[S])}w(e).$$

### Algorithm A (Weight reduction for the weighted $k$-clique problem)

Given a $d$-hypergraph $G$ with edge weights $w : E(G) \rightarrow \mathbb{N}$, a number $k$, a weight target $t \in \mathbb{N}$, 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 $p = \log_q M$.

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

iii. (Expansions and carries satisfy a quadratic equation.) There is a sequence $c_0, c_1, \ldots \in \{0, \ldots, 2\frac{k}{S}\}$ such that $c_0 = 0$ and the following quadratic equation holds:

$$\sum_{\ell \in \mathbb{N}} \left( c_\ell - t\ell - qc_{\ell+1} + \sum_{e \in E(G[S])}w(e) \right)^2 = 0. \quad (2)$$

**Proof.** The equivalence between ii. and iii. follows from the fact that a sum of squares is zero if and only if all summands are zero. To see that i. implies ii., suppose $t = \sum w(e)$, so the $q$-expansions are the same as well: $t\ell = \sum w(e)\ell$ for all $\ell$. We inductively set the carries so as to satisfy the linear equations; this choice is unique. It remains to argue that the $c_\ell$ are integers between $0$ and $2\frac{k}{S}$. The fact that they are non-negative integers is a standard property of $q$-expansions, so we only show the upper bound. We do so by induction: It clearly holds for $c_0$. In general, we have

$$c_{\ell+1} = \left( -\frac{t\ell}{q} + \sum_{e \in E(G[S])}w(e) \right) \mod q. \quad (3)$$

The first summand $-t\ell/q$ is at most 0, the second summand $c_\ell/q$ is at most $2\frac{k}{S}/q \leq \frac{k}{S}$ by induction and $q \geq 2$, and the third summand is at most $\frac{k}{S}$, because $w(e) < q$ holds and the sum has at most $\frac{k}{S}$ terms $e$.

To see that the second claim implies the first, we observe

$$\sum_{\ell \in \mathbb{N}} q^{\ell} \sum_{e \in E(G[S])}w(e) = \sum_{\ell} q^{\ell}(t\ell + qc_{\ell+1} - c\ell) = t + \sum_{\ell > 0} q^{\ell}c_\ell - \sum_{\ell} q^{\ell}c_\ell = t - c_0 = t. \quad \Box$$

The following algorithm uses (3) to reduce weights; in particular, we use the binomial theorem $(a + b)^2 = a^2 + 2ab + b^2$ in (2) (with $a = c_\ell - t\ell - qc_{\ell+1}$ 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, 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 hypergraphs).
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 \(O(p^d k^{2d} k^p)\) and makes at most \(k^p\) 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 \(E(G) = \{V(G) \leq d\}\), edge weight function \(w : E(G) \to \mathbb{N}\), and target value \(t \in \mathbb{N}\) after applying Fact 3.1. Let \(G'\) be the \(2d\)-hypergraph with \(E(G') = \{V(G) \leq 2d\}\).

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 \((c_\ell)_{\ell \in \mathbb{N}}\) satisfying (2). Now consider the weight of \(S\) in \(G'\). We abbreviate the terms in the left side of (2) with \(a_\ell = c_\ell - t - q \ell e + 1\) and \(b_\ell = \sum_{e \in E(G)} w(e)\), and have

\[
\sum_{e \in E(G)} w(e) = \sum_{\ell \in \mathbb{N}} (a_\ell + b_\ell)^2 = \sum_{\ell \in \mathbb{N}} (a_\ell^2 + 2a_\ell b_\ell + b_\ell^2) = \sum_{\ell \in \mathbb{N}} (a_\ell^2 + \sum_{e \in E(G)} b_\ell^2).
\]

The squares of \(b_\ell\) expand as follows:

\[
b_\ell^2 = \sum_{e \in E(G)} w(e) = \sum_{e \in E(G), e = e_2} w(e_1) \cdot w(e_2).
\]

Now we observe that \(t' + w'(f)\) were defined exactly as to satisfy

\[
\sum_{f \in E(G'[S])} (a_\ell + b_\ell)^2 = -t' + \sum_{f \in E(G'[S])} w'(f).
\]

By Lemma 3.3, the set \(S\) 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 \(O(k^p)\) iterations, and each iteration takes time \(O(n^d p^d)\) because of line A3a in which we need to compute \(w'(f)\) for every edge; overall the reduction takes time \(O(n^d k^{2d} k^p)\) and makes at most \(k^p\) oracle queries.

For the weights, note \(c_\ell \leq 2k\ell^d\) and so \(|t'| \leq O(k^{2d} q^p)\). To get the bound on \(w'(f)\), observe that the term

\[
\sum_{e_1 \cup e_2 = f} w(e_1) w(e_2)
\]

is bounded by \(4^d q^2\), and the term \(w'(f) \cdot (c_\ell - t - q \ell e + 1)\) 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 \(O(\text{log} n)\), 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 \([-n^d k, \ldots, n^d k]\) to \(n^{e(1)}\)-partite \(2d\)-hypergraph \(k\)-Clique. If the input has \(n\) vertices, every oracle query has \(n\) vertices and the reduction uses at most \(n^{e(1)}\) queries. Here the \(o(1)\) terms are of the form \(g(k, d) / \sqrt{\text{log} n}\).

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

Each query is now a \(k\)-partite instance \((G', w', k', t')\) with maximum weight \(M'\), where we treat \(k\) and \(d\) as constants. A solution \(S\) of \(G'\) satisfies \(\sum_{e \in E(G'[S])} w'(e) = t'\). 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 \(c_\ell \in E(G'[S])\) that intersects exactly the color classes in \(C\), and this edge contributes \(w'(c_\ell)\) to the total weight of \(S\). We want to simulate these weights by exhaustively guessing the contribution \(w'(c_\ell)\) 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 \((2M' + 1)^k\) \(= n^{e(1)}\) candidate weight vectors \(a = (a_C)_{C \subseteq \{1, \ldots, k\}}\). If the 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 \(s_\ell\) 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 \(n^{e(1)}\) candidate vectors when \(k\) is regarded as a constant, and each oracle query \(G_a\) is prepared 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 \(k\)-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, to find \(x_1 \in X_1, \ldots, x_k \in X_k\)
Lemma 3.6. Let $d, k \in \mathbb{N}$ with $1 \leq d \leq k$. There is a many-one reduction from (unweighted) $d$-partite $k$-hypergraph $G$-Clique to $k$-OV that runs in time $O(n^{d+1}) \cdot \text{polylog } n$; the number of produced vectors is $n$ and the dimension of the vectors is $n^d$.

Proof. Let $G$ be a $k$-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 every non-edges $h \in \overline{E}(G)$ satisfy $h \not\subseteq \{v_1, \ldots, v_k\}$. Here $\overline{E}(G)$ denotes $\{e \in (V(G) \setminus \{v\}) \mid \forall i : \{e \cap V_i \mid \leq 1\} \subseteq \overline{E}(G)\}$.

We construct the instance $X_1, \ldots, X_k$ of $k$-OV as follows. For each $v \in V_i$, we create a vector $x_v \in \{0, 1\}^{|E(G)|}$ 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 $x_v(h) = 0$. Otherwise we have $h \cap V_i = \emptyset \neq \{v\}$ for some $u$, and we set $x_u(h) = 0$. 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 are easily computed in $O(n^{d+1} \cdot \text{polylog } n)$ 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\} is a solution to the $k$-OV instance, that is, we claim $\sum_{h \in \overline{E}(G)} \prod_{v \in V_i} x_v(h) = 0$. 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 $v_i \in h$. By definition, we have $x_v(h) = 0$. Thus the entire sum is indeed zero.

For the reverse direction, let $x_v, \ldots, x_v$ with $v_i \in X_i$ 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 $x_v = 0$. By definition, this implies that $h \cap V_i = \emptyset$ for some $u$. Thus in particular, $h \not\subseteq \{v_1, \ldots, v_k\}$ and 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 [54].

Lemma 3.7. Let $k \in \mathbb{N}$. There is an $O(n^{k/2})$ time many-one-reduction from $k$-OV to 2-OV that maps instances with $n$ vectors in dimension $D$ to instances with $O(n^{k/2})$ 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 $S \subseteq X_1 \cup \cdots \cup X_{k/2}$ with $|S \cap X_i| = 1$ for all $i \in \{1, \ldots, [k/2]\}$, we create a vector $v_S \in X_{k/2}^\perp \subseteq \{0, 1\}^D$ by setting $v_S(u) = \prod_{u \in S} u_i$. Similarly, for each $S' \subseteq X_{k/2+1} \cup \cdots \cup X_k$ with $|S' \cap X_i| = 1$ for all $i \in \{[k/2] + 1, \ldots, k\}$, we create a vector $v_{S'} \in X_{k/2}^\perp \subseteq \{0, 1\}^D$ by setting $v_{S'}(u) = \prod_{u \in S'} u_i$. We obtain an instance $X_1, X_{k/2} \subseteq \{0, 1\}^D$ of 2-OV.

We claim that $X_1, X_{k/2}$ is a yes-instance of $k$-OV if and only if $X_1^\perp, X_{k/2}^\perp$ is a yes-instance of 2-OV. Suppose that $v_1, \ldots, v_k$ are orthogonal, that is, $\prod_{j=1}^{k} v_j(u) = 0$ holds for all $j \in \{1, \ldots, D\}$. We set $S = \{v_1, \ldots, v_{k/2}\}$ and $S' = \{v_{k/2+1}, \ldots, v_k\}$. Clearly $v_S^\perp \cdot v_{S'}^\perp = 0$ holds for all $j$, so $v_S^\perp, v_{S'}^\perp \in V'$ are indeed orthogonal.

Conversely, if $v_S^\perp \cdot v_{S'}^\perp = 0$ 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 (rephrased). If the moderate-dimension OV conjecture is false, then there exists an $\varepsilon > 0$ such that for every integer $d$ there is a (large) integer $k$ such that $k(d, \varepsilon)$ satisfying the following statements:

- $k$-Clique can be solved on $d$-hypergraphs in time $O(n^{1-\varepsilon}k)$. $\text{Exact-Weight}-k$-Clique and $\text{Min-Weight}-k$-Clique can be solved on $d$-hypergraphs with weights in $\{-M, \ldots, M\}$ in randomized time $O(n^{1-\varepsilon}k)$ · \text{polylog } M$.

Proof. Let $(G, w, k, t)$ be an instance of $\text{Min-Weight}-k$-Clique on $d$-hypergraphs. We summarize the lemmas of this section as follows:

1. Lemma 3.2 randomly reduces in $\text{polylog } (M)$ time from 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 $\text{poly}(k)$.

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

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

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

Composing the reducing gives a randomized $O(n^{k/2} n^d + o(1))$ time oracle reduction from Exact-Weight-$k$-Clique on $d$-hypergraphs with $n$ vertices and largest weight $M$ to $2$-OV on $n^{k/2}$ dimensions using $n^d + o(1)$ oracle calls.

To reduce from $\text{Min-Weight}-k$-Clique to Exact-Weight-$k$-Clique we use [43, Theorem 1], which allows us to perform a binary search for the minimum-weight-$k$-clique by making few queries to Exact-Weight-$k$-Clique. As the domain $E(G)$ of our weight function is of size $\text{poly}(n^d)$ and the maximum weight is upper bounded by $M$, this reduction requires $O(n^d \text{log } M)$ oracle calls. This yields a randomized oracle reduction from $\text{Min-Weight}-k$-Clique on $d$-hypergraphs with $n$ vertices and largest weight $M$ to $2$-OV, which runs in time $O(n^{k/2} n^d + o(1))$ polylog(cyl) and makes $n^d + o(1)$ log oracle calls.

To reduce from $k$-Clique on $d$-hypergraphs with $n$ vertices to $2$-OV, we only use steps 3 and 4, which takes time $O(n^{k/2} n^d + o(1))$.

All three reductions produce instances of $2$-OV with $n$ vectors in $D$ dimensions, where $N = n^{k/2}$ and $D = n^d$, for some $\delta = \delta(k, \varepsilon) > 0$. If $k$ is large enough compared to $\varepsilon$, then $\delta$ is arbitrarily close to zero. If the moderate-dimension OV conjecture is false, then $\delta > 0$ such that $2$-OV with $D = n^d$ has an $O(n^{2-\varepsilon''})$ time algorithm. Combined with any of the three reductions, we obtain three algorithms that run in time at most $O(n^{k/2} n^d + o(1) + n^{k/2}(2-\varepsilon'' - d + o(1)))$ polylog(M).

When $k$ is large enough compared to $d$, this is $O(n^{(1-\varepsilon'')} \cdot \text{polylog } (M))$ for some $\varepsilon' = \varepsilon'' > 0$. This proves the claim. □
And we have the following corollary to Theorem 1.3.

**Corollary 1.4 (restated).** If the moderate-dimension OV conjecture is false, then there exists an $\epsilon > 0$ such that, for all integers $d$, there is an algorithm for Max-d-SAT that runs in time $O^*(d(2 - \epsilon)^n)$.

The corollary follows from Theorem 1.3 with a reduction from Max-d-SAT to $k$-Clique on $d$-hypergraphs that was already sketched in e.g. [54]. We formally state and prove this reduction next.

**Lemma 3.8.** Let $d, k \in \mathbb{N}$ with $1 \leq d \leq k$. There is an $O^*(2d^2/n^2)$ time reduction from Max-d-SAT to Min-Weight-k-Clique on $d$-hypergraphs that maps $d$-CNF formulas with $n$ variables and $m$ clauses to $d$-hypergraphs with at most $k2^n/n$ vertices and integer edge weights between $-2m$ and $2m$.

**Proof.** Given an instance of Max-d-SAT consisting of a $d$-CNF formula $\phi$ 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 $V_1, \ldots, V_k$ where $|V_i| \leq n/k$. The reduction computes from $\phi$ an instance $H$ of Min-Weight-k-Clique.

We build a complete $k$-partite $d$-hypergraph $H$ with vertices $\bigcup_{j=1}^k V_i$ where $P_i$ contains a vertex $p_i^x$ for every vector $x \in \{0, 1\}^{|V_i|}$. Create an edge $e = \{x_1, \ldots, x_\ell\}$ for every set $\{i_1, \ldots, i_\ell\} \subset \{1, \ldots, k\}$ and tuple $(x_1, \ldots, x_\ell) \in P_{i_1} \times \cdots \times P_{i_\ell}$. Define the weight of $e$ to be $-1$ times the number of clauses that

1. are contained in $\bigcup_{j=1}^k V_i$,
2. contain a variable in $V_i$ for every $j = 1, \ldots, \ell$, and
3. are satisfied by the partial assignment obtained by setting the variables in $V_i$ according to $x_j$.

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 $(k2^n/n^2)^d$ and we compute their weights in polynomial time, the running time of this reduction is bounded by $O^*(d^2n^{2d})$.

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 $\phi$. We claim that the weight of $X$ is $-t$ times the number of clauses satisfied by $x$ and therefore $\phi$ has an assignment satisfying at least $t$ clauses. To see the claim, let $C$ be a clause of $\phi$ and $i : C$ contains variables from $V_i = \{i_1, \ldots, i_q\}$ be the set of variable groups intersecting $C$. Let $x_{i_1}, \ldots, x_{i_q}$ be the corresponding partial assignments that $x$ induces into $V_{i_1}, \ldots, V_{i_q}$. We see that $C$ contributes $-1$ to the weight of the hyperedge $(x_1, \ldots, x_q)$ and $0$ to all other edges.

For the reverse direction, suppose $x$ is an assignment satisfying at least $t$ clauses of $\phi$ and let $x_j$ be the projecting of $x$ onto $V_i$. Then by the above claim the weight of $X := \{p_{x_1}^1, \ldots, p_{x_q}^k\}$ is $-t$. □

**Proof of Corollary 1.4.** For a sufficiently large constant $k$, we combine the reduction in Lemma 3.8 with an $O(n^{1-(1-\epsilon)/2})$ polylog $M$ time algorithm for Min-Weight-k-Clique in $d$-hypergraphs. This yields an $O^*(2d(1-\epsilon)^n)$ time algorithm for Max-d-SAT. Together with Theorem 1.3 this proves the claim. □

**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 algorithms for sparse circuit satisfiability exist. We demonstrate how to do this in Section 4.1 for sparse formulas as a warm-up, reproving a result of [28]. In Section 4.2, we prove Theorem 1.5, the extension of this result to sparse TC^0-circuits. We also prove that Theorem 1.6 for sparse TC^1-circuits and CNF-SAT in Section 4.3.

**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 (NEG, AND, 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 [33, 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 $\ell$ 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, 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 $F'$.

1. **(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.

2. **(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 $y_0$ where $0 \in V(F)$ is the output gate of $F$.

3. **(Compute $k$-CNFs for small subcircuits)** For each $v \in A \cup \{0\}$, do the following:
   a. Let $F_{v, A}$ be the subcircuit of $F$ induced by the set $R_C(A, v)$ (see (1) in the preliminaries) and interpret the gates $a \in A$ as input variables $y_a$.
   b. Since $F_{v, A}$ depends on at most $2\ell$ variables, we can compute a $k$-CNF formula $F'_{v, A}$ with at most $2\ell + 1 \leq k$ variables that expresses the constraint "$y_0 = F_{v, A}(x, y)$".

4. **(Output)** Let $P' = y_0 \land \bigwedge_{v \in A} F'_{v, A}$, and output $F'$.

We prove the correctness and properties of this algorithm.

**Lemma 4.2.** Let $c, \epsilon > 0$. 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 $(1 + \epsilon) \cdot n$ variables.

**Proof.** We set $k = c/(2\epsilon)$. Let $F$ be the input formula with $n$ variables $x_1, \ldots, x_n$ and $m \leq cn$ 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 $(1 + \epsilon)n$ 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 $|A| \leq 24cn/k \leq cn$, so indeed $F'$ has at most $(1 + \epsilon)n$ variables. By Lemma 4.1, $F_{\ell,A}$ contains at most $\ell$ non-input gates and, since the fan-in is at most two, $F_{\ell,A}$ contains most $2\ell$ gates overall. So the constraint \( y_v = F_{\ell,A}(x, y) \) 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. □

Using this lemma, we prove that SETH is implied by an analogue of SETH for sparse formulas.

**Theorem 4.3 (Dantsin and Wolpert [28]).** If SETH is false, then there is an $\epsilon > 0$ such that, for all $c$, the satisfiability of Boolean formulas of size at most $cn$ can be solved in time $O((2-\epsilon)^n)$.

**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 $O((2-\delta)^{n'}) = O((2-\delta)^{(1+\alpha)n}) = O((2-\delta)^n)$ for some suitable $\epsilon, \alpha > 0$. □

### 4.2 Sparse TC0-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 $\epsilon \in (0,1)$ and a depth-$d$ threshold circuit with at most $cn$ wires, with $c \geq 1$, produces a $k$-CNF formula $\varphi$ with at most $(1 + \epsilon)n$ variables and width $k \leq (2000(c/\epsilon) \log(2c/\epsilon))^d$.

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 $C_{add} : \{0,1\}^{b\ell} \rightarrow \{0,1\}^{\lfloor b\log \ell \rfloor}$ over $\text{NEG}, \text{AND}, \text{OR}$ with at most $40b\ell$ gates and maximum fan-in 2 such that $C_{add}$ computes the binary representation of the sum of $\ell$ 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

\[
((b_1 + b_2) + (b_3 + b_4)) + ((b_5 + b_6) + (b_7 + b_8)) + \ldots
\]

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

\[
\sum_{i=1}^{\lfloor \ell/2 \rfloor} 20(b + i - 1)\ell/2^i \leq 20b\ell \sum_{i=1}^{\infty} i/2^i = 40b\ell.
\]

Our proof also needs a circuit computing the threshold function for binary inputs.

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

**Proof.** The circuit BINTH$_\theta$ is constructed by setting $t = \lfloor \log \theta \rfloor$ and converting the following expression to a circuit:

\[
\left( \bigvee_{i=t}^{r-1} x_i \right) \vee \left( x_{r-1} \wedge \text{BINTH}_{\theta^2^{r-1}}(x_0, \ldots, x_{t-2}) \right).
\]

We now have all tools needed to prove Lemma 4.4.

**Proof of Lemma 4.4.** Without loss of generality, threshold circuits only have input, NEG, and TH$_0$ gates, since TH$_0$ can directly simulate AND, OR, and MAJ gates. The algorithm to transform threshold circuits into 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 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$_\theta$-gate.
- b) Partition the children $N_C(v)$ of $v$ into blocks $B_1, \ldots, B_\ell$ of size at most $\beta$ 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 $i \in \{1, \ldots, \ell\}$, 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 $1$s 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 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$_\theta$ (from Lemma 4.6). This concatenated circuit takes the binary representation of $\ell$ 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 $\theta$. The circuit

\[
\text{BINTH$_\theta(C_{add}(b_1, \ldots, b_\ell))}
\]

has at most $40\ell + 2(b + \lceil \log \ell \rceil) \leq 44b\ell$ 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 $R_C(A, v)$ (see (1) 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 $\beta^d$ variables, so we can compute a $k$-CNF formula $F_v$ with at most $k = \beta^d$
variables that expresses the constraint “\( y_v = C_{v,A}(x,y) \)”, where \( x_1, \ldots, x_n \) are the input variables of \( C \).

\textbf{C4 (Output)} Let \( F = y_0 \land \bigland_{v \in A} F_v \), and output \( F \).

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 \( y_v = C_{v,A}(x,y) \) holds for all \( A \). This setting for \( y \) then satisfies the constraint “\( y_v = C_{v,A}(x,y) \)” and thus \( F_v(x,y) = 1 \). Moreover, \( y_v = C(x) = 1 \) 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 \( y_v = 1 \) and \( C_{v,A}(x,y) = y_v \) holds for all \( A \). 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 \( y_v \). However, for each such variable, we know by the induction hypothesis that \( y_v = C_{v,A}(x,y) = C_v(x) \) holds, and thus we have \( C_v(x) = C_{v,A}(x,y) \) by the definition of \( C_v \). In particular \( C(x) = y_0 = 1 \) holds and so \( x \) satisfies \( C \). This establishes the correctness of the reduction, except for proving that width \( k = \beta^d \) 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 \( u \in (I(C) \cup A) \cap R_C(A,v) \). 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 \((I(C) \cup A) \cap R_C(A,v)\). We need to prove that the set \((I(C) \cup A) \cap R_C(A,v)\) has size less than \( k = \beta^d \) 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 \( b_1 \)-gate \( v \) is determined by the \( B_1 \)-gates below it, so we can bound \(|(I(C) \cup A) \cap R_C(A,v)| \leq \sum_{u \in B_1} |(I(C) \cup A) \cap R_C(A,u)| \). Any gate \( u \in B_1 \) already existed in the original circuit. If \( u \) has degree at least \( \beta \) in the original circuit, then we ran step C2 on \( u \) and thus \( u \) belongs to \( A \). Otherwise, \( u \) has less than \( \beta \) 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 \( \beta^{d_u} \) nodes in \( A \) without going through any other node in \( A \), i.e., \(|(I(C) \cup A) \cap R_C(A,u)| < \beta^{d_u} \). Since \( u \) is a descendant of \( v \), we have \( d_u < d \). In total, we obtain \(|(I(C) \cup A) \cap R_C(A,v)| < |B_1| \beta^{d-1} < \beta^d \). It follows that the constraints in step C3b indeed consider at most \( k = \beta^d \) variables and can thus be expressed in \( k \)-CNF.

Bounding the number of variables. It remains to set \( \beta \) in such a way that \(|A| \geq m \), 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 \( d_C(v) \leq \beta \). For any such gate \( v \), we add at most \( 50\ell \beta^d \) gates to \( A \) (see step C2d), where \( \beta = \log \beta + 1 \) and \( \ell \leq 1 + d_C(v) \beta < 2d_C(v) / \beta \) since \( d_C(v) \geq \beta \). Hence, overall the number of gates ever added to \( A \) is at most

\[
\sum_{v \in \forall(G)} 100 \frac{d_C(v)}{\beta} \log \beta + 1 \leq 100 cn \log \beta + 1 \beta \cdot
\]

Thus we can set \( \beta = \beta(c,\varepsilon) \in N \) as a function of \( c \) and \( \varepsilon \) such that the size of \( A \) is at most \(en\). One can check that \( \beta \leq 2000(\varepsilon/c)(\log(2\varepsilon)/c) \) 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 gates, by replacing the BINTH 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 \( c(\delta) \)-size DeMorgan circuit when given \( s \in \{0, \ldots, d\} \) in binary.

Using Lemma 4.4 for an \( \varepsilon \) with \((1+\varepsilon)n < 1\) we obtain:

**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 improve the dependence of \( k \) on \( c, \varepsilon \) 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 [52]. We use the following variant [37, Lemma 1.4] (see also [53, Section 4.2]).

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

We use Lemma 4.7 to improve the dependence of $k$ in Lemma 4.4.

Lemma 4.8. There is an algorithm that, given $\epsilon > 0$ and a depth-$d$ threshold circuit with at most $cn$ wires, produces at most $2^{\epsilon n/2} k$-CNF formulas $\varphi_1, \ldots, \varphi_z$ on $(1 + \epsilon/2)n$ variables such that the circuit $C$ is satisfiable if and only if $\varphi_i$ is satisfiable for some $i \leq z$, and we have $k \leq 4000(c/\epsilon) lg(4c/\epsilon)^2 (2d)^{1-\epsilon/12c}$.

Proof. Let $C = (G = (V, E), \lambda)$ be the given circuit. Apply Lemma 4.7 to $G$ with $\delta = [\lg d]$ and $r = r/2c$. We obtain a set $R$ of size at most $cn/2$ such that $(V, E \setminus R)$ has depth at most $2^\delta (1-\epsilon/12c) \leq 2d^{1-\epsilon/12c}$. For every assignment $a \in \{0, 1\}^R$ we create a circuit where we require that $C_a(v) = a_v$ for every $v \in R$, remove their outgoing wires, and update their incident gate accordingly (i.e. if $a_v = 1$ and the incident gate is a $\text{TH}_0$ it becomes a $\text{TH}_{\theta-1}$ gate). The obtained circuit has depth at most $(2d)^{1-\epsilon/12c}$ 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 $(\log n)^{1+\epsilon(1)}$.

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

Proof. Apply Lemma 4.8 with $\epsilon/2$ to obtain $2^{\epsilon n/2} k$-CNF formulas with $k \leq 2^O((\log n)^{1+\epsilon/12c})$ on $(1 + \epsilon/2)$ variables. For sufficiently small $\delta = \delta(\epsilon, c) > 0$ we have $k \leq 2^{o((\log n))} = o(n/\log n)$ and thus the number of clauses $m$ is at most $(2n)^k = n^{\Theta(\log n)} \leq 2^\Theta(n)$. Therefore the assumed algorithm for $\text{CNF-SAT}$ determines the satisfiability of the produced CNF formula in time $2^{\epsilon n/2} \cdot O^*(1) \cdot 2^{(1-\epsilon')n/2}$, which is $O(2^{1-\epsilon'n})$ for any $\epsilon' < \epsilon^2/2$. □

Open Question. It is known that Lemma 4.7 cannot be significantly improved (see [51]). 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 that this can be exploited more: Given a DAG $G = (V, E)$ of depth $2^a$ 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 $\min(l(R) : |R| \leq \alpha m)$ better than $2^{(1-\epsilon)\delta}$ (which is implied by Lemma 4.7).

5 SCHEMATIC OVERVIEW OF OUR RESULTS

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

ACKNOWLEDGMENTS

Jesper Nederlof is supported by NWO Veni grant 639.021.438.

REFERENCES

[1] Amir Abboud, Arturs Backurs, Karl Bringmann, and Marvin Künnemann. 2017. Fine-Grained Complexity of Analyzing Compressed Data: Quantifying Improvements over Decompress-and-Solve. In 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017, Chris Umans (Ed.). IEEE Computer Society, 192–203. https://doi.org/10.1109/FOCS.2017.26

[2] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. 2015. Tight Hardness Results for LCS and Other Sequence Similarity Measures. In IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015, Venkatesan Guruswami (Ed.). IEEE Computer Society, 59–78. https://doi.org/10.1109/FOCS.2015.14

[3] Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. 2015. Subcubic Equivalences Between Graph Centrality Problems. APSP and Diameter. In Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015, Piotr Indyk (Ed.). SIAM, 1681–1697. https://doi.org/10.1137/1.9781611973730.112

[4] Amir Abboud, Kevin Lewi, and Ryan Williams. 2014. Losing Weight by Gaining Edges. In Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings (Lecture Notes in Computer Science), Andreas S. Schulz and Dorothea Wagner (Eds.), Vol. 8737. Springer, 1–12. https://doi.org/10.1007/978-3-662-44777-2_1
