Finding Small Satisfying Assignments Faster Than Brute Force: A Fine-Grained Perspective into Boolean Constraint Satisfaction

Marvin Künnemann
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
marvin@mpi-inf.mpg.de

Dániel Marx
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
dmarx@mpi-inf.mpg.de

Abstract

To study the question under which circumstances small solutions can be found faster than by exhaustive search (and by how much), we study the fine-grained complexity of Boolean constraint satisfaction with size constraint exactly k. More precisely, we aim to determine, for any finite constraint family, the optimal running time $f(k)n^{g(k)}$ required to find satisfying assignments that set precisely k of the n variables to 1.

Under central hardness assumptions on detecting cliques in graphs and 3-uniform hypergraphs, we give an almost tight characterization of $g(k)$ into four regimes:

1. Brute force is essentially best-possible, i.e., $g(k) = (1 \pm o(1))k$,
2. the best algorithms are as fast as current k-clique algorithms, i.e., $g(k) = (\omega/3 \pm o(1))k$,
3. the exponent has sublinear dependence on k with $g(k) \in [\Omega(\sqrt[3]{k}), O(\sqrt{k})]$, or
4. the problem is fixed-parameter tractable, i.e., $g(k) = O(1)$.

This yields a more fine-grained perspective than a previous FPT/W[1]-hardness dichotomy (Marx, Computational Complexity 2005). Our most interesting technical contribution is a $f(k)n^{4\sqrt{k}}$-time algorithm for SubsetSum with precedence constraints parameterized by the target k – particularly the approach, based on generalizing a bound on the Frobenius coin problem to a setting with precedence constraints, might be of independent interest.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Fine-grained complexity theory, algorithmic classification theorem, multivariate algorithms and complexity, constraint satisfaction problems, satisfiability

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.27

Funding *Dániel Marx*: Research of the second author was supported by funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement SYSTEMATICGRAPH (No. 725978).

1 Introduction

Extensive research in complexity theory has established methods to give precise qualitative results on the computational hardness of problems. In this context, a basic question that we would like to answer is: When are there algorithms better than a brute force search, and if there are, how much improvement is possible compared to brute force? In problem settings where the task is to find a solution of size k, typically it is easy to obtain algorithms with running time of the form $\mathcal{O}(n^{k+\mathcal{O}(1)})$ by a brute force search of every possible solution. In such cases, beating brute force could involve having an algorithm with a term $(1-\epsilon)k + O(1)$

in the exponent for some $\epsilon > 0$, or having sublinear (e.g, $O(k/\log k)$ or $O(\sqrt{k})$) dependence on k in the exponent, or we might be able to completely remove k from the exponent of n with an $f(k)n^{O(1)}$ time algorithm.

In this paper, we study the above question in the context of the class of Boolean Constraint Satisfaction problems. Fixing a *constraint family* $\mathcal{F}$ of Boolean functions, the task is to determine an assignment to Boolean variables $x_1, \ldots, x_n$ satisfying a given conjunction of constraints of the form $f(x_{i_1}, \ldots, x_{i_r})$ with $f \in \mathcal{F}$ and $i_1, \ldots, i_r \in [n]$. Here, the natural notion of the solution size k is the number of variables set to 1 and we consider the task of determining a satisfying assignment with precisely k ones. This class indeed contains a variety of problems: basic graph problems such as the vertex cover problem ($\mathcal{F}$ consists of the binary OR) and the independent set problem in graphs ($\mathcal{F}$ consists of the binary NAND) or d-uniform hypergraphs ($\mathcal{F}$ consists of the d-ary NAND), but also other natural problems such as a formulation of SubsetSum parameterized by the target k ($\mathcal{F}$ consists of binary equality)[1], finding a solution of a (sparse) linear system over GF(2) where each linear equality involves at most a constant number r of variables and the solution must have precisely k ones ($\mathcal{F}$ consists of all linear constraints of arity at most r), as well as finding a closed set of size k in a directed graph ($\mathcal{F}$ consists of the binary implication). Note that the last problem can be seen to be equivalent to a variant of SubsetSum that prescribes precedence constraints on the items and uses an unary encoding for all item sizes.

The time complexity inside this class varies widely: Vertex cover is famously fixed-parameter tractable when parameterized by k, with a best current running time bound of $\mathcal{O}(kn + 2^{O(k)})$ [16]. It is even simpler to solve the SubsetSum formulation in time $\mathcal{O}(m + k^2) = \mathcal{O}(n^2)$ (where m is the number of edges in the graph) by a straightforward algorithm[2]. The fastest known algorithm for independent set [29], however, relies on the sophisticated techniques for matrix multiplication, and achieves a running time of $\mathcal{O}(n^{(\omega/3)k})$ for k divisible by 3, where $\omega \leq 2.373$ is the matrix multiplication exponent. For finding closed sets of size k, a surprisingly simple $\mathcal{O}(n^{k/2})$-time algorithm[3] improves over brute force even without matrix multiplication, but a priori there is little indication for the optimality of this approach. Finally, for finding independent sets in 3-uniform hypergraphs, no substantially faster-than-brute-force algorithm is known.

The central purpose of this paper is to give a detailed understanding of the time complexity of Boolean constraint satisfaction parameterized by solution size k, particularly when k is considered a (large) constant: How precisely can we determine the running time $f(k)n^{g(k)}$, with $g(k)$ as small as possible? Note that for large constant k, we have $f(k)n^{g(k)} = \mathcal{O}(n^{g(k)})$ and aim to determine its optimal polynomial-time complexity.

A classification of the second author [28] resolves the qualitative question for which $\mathcal{F}$ the problem is solvable in FPT time (assuming $\mathsf{FPT} \neq \mathsf{W}[1]$), i.e., when $g(k)$ can be bounded by a constant independent of k. In particular, from this classification, we obtain that among

[1] To see the correspondence, note that if $\mathcal{F}$ consists of the binary equality, $\mathsf{SAT}(\mathcal{F})$ asks to find a union of connected components of total size k. By representing each connected component by its size (after linear-time preprocessing), this is precisely the SubsetSum problem with target k.

[2] Determine all connected components in time $\mathcal{O}(m)$ and solve a SubsetSum instance on the component sizes in time $\mathcal{O}(k^2)$ using Bellman's pesudopolynomial-time algorithm or recent improvements [23, 8].

[3] Without loss of generality, it suffices to solve the following problem: given a node-weighted DAG $G = (V, E)$ and $k \in \mathbb{N}$, find a weight-k subset $S \subseteq V$ such that $u \in S$ and $(u, v) \in E$ implies $v \in S$. If S contains a set S' of at most $k/2$ *sources* (i.e., vertices that have no incoming edges from other vertices in S), we can simply guess S' and check that S' and the set of all descendants of S' have total weight k. If S contains no such set S' of size at most $k/2$, we can guess all $\leq k/2$ non-sources S'', remove all incoming edges to S'' and find a weight-$(k - |S''|)$ set of vertices with out-degree 0.

the above examples, vertex cover, SubsetSum with target k, and the sparse linear systems over GF(2) can be solved in time $f(k)n^c$, while for independent set (in both graphs and hypergraphs) as well as SubsetSum with precedence constraints, the exponent of n must depend on k (unless $\mathsf{FPT} = \mathsf{W}[1]$). Can we obtain tight bounds on $g(k)$ when it must depend on k? In particular, can we determine for which $\mathcal{F}$ the brute-force $\mathcal{O}(n^{k+c})$-time solution is essentially optimal?

1.1 Our Results

Let us formally state our problems and results.

▶ **Problem 1.1.** *Let $\mathcal{F}$ be a finite* constraint family *of Boolean functions. The problem* $\mathsf{SAT}(\mathcal{F})$ *asks to determine whether a given formula ϕ on Boolean variables $x_1, \dots, x_n$ is satisfiable by an assignment with k ones, where ϕ is a conjunction of m constraints C of the form $f(\mathbf{x})$, where $f : \{0,1\}^r \to \{0,1\}$ is a constraint function in $\mathcal{F}$ and $\mathbf{x}$ is an r-tuple of variables among $x_1, \dots, x_n$.*

Note that if all $f \in \mathcal{F}$ have arity bounded by r, then there are at most $\mathcal{O}(n^r)$ possible constraints, and exhaustive search solves $\mathsf{SAT}(\mathcal{F})$ in time $\mathcal{O}(n^{k+r})$.

We will show that the complexity of $\mathsf{SAT}(\mathcal{F})$ is tightly characterized by the set of functions *expressible as restrictions of constraint functions $f \in \mathcal{F}$*. To formally introduce this concept, let $f : \{0,1\}^r \to \{0,1\}$ be an arbitrary Boolean function. We say that $g : \{0,1\}^s \to \{0,1\}$ is a *restriction of f* if it is obtained from g by replacing each argument of f by either the constant 0, the constant 1, or an argument of g, i.e., we can partition $[r]$ into $X_1, \dots, X_s, Z_0, Z_1$ such that

$$g(x_1, \dots, x_s) = f(\overbrace{x_1 \dots x_1}^{X_1}, \dots, \overbrace{x_s \dots x_s}^{X_s}, \overbrace{0 \dots 0}^{Z_0}, \overbrace{1 \dots 1}^{Z_1}).$$

Here, $\overbrace{y \dots y}^{Y}$ denotes plugging in y for all (not necessarily contiguous) positions $Y \subseteq [r]$, see Section 2.

▶ **Definition 1.2.** *Let $g : \{0,1\}^d \to \{0,1\}$ be an arbitrary Boolean function. A constraint family $\mathcal{F}$* represents *g if there is some $f \in \mathcal{F}$ such that g is a restriction of f. If $\mathcal{F}$ does not represent g, we say that $\mathcal{F}$* avoids *g.*

Let $\mathrm{IMPL} : \{0,1\}^2 \to \{0,1\}$ and $\mathrm{NAND}_d : \{0,1\}^d \to \{0,1\}$ be the binary implication and d-ary NAND function, respectively, i.e.,

$$\mathrm{IMPL}(y_1, y_2) \coloneqq \overline{y_1} \vee y_2,$$
$$\mathrm{NAND}_d(y_1, \dots, y_d) \coloneqq \overline{\bigwedge\nolimits_{i=1}^{d} y_i}.$$

In [28], it is shown that assuming $\mathsf{FPT} \neq \mathsf{W}[1]$, $\mathsf{SAT}(\mathcal{F})$ is solvable in FPT time $f(k)n^c$ if and only if $\mathcal{F}$ is *weakly separable*, which is a condition equivalent to $\mathcal{F}$ avoiding NAND_2 and IMPL. We show an almost tight characterization of $g(k)$ (under plausible assumptions from fine-grained complexity theory) that depends only on whether or not $\mathcal{F}$ represents IMPL, NAND_2 or NAND_d for higher order $d \geq 3$. Specifically, we obtain the following main theorem, illustrated in Figure 1.

▶ **Theorem 1.3.** *Let $\mathcal{F}$ be a finite constraint family.*

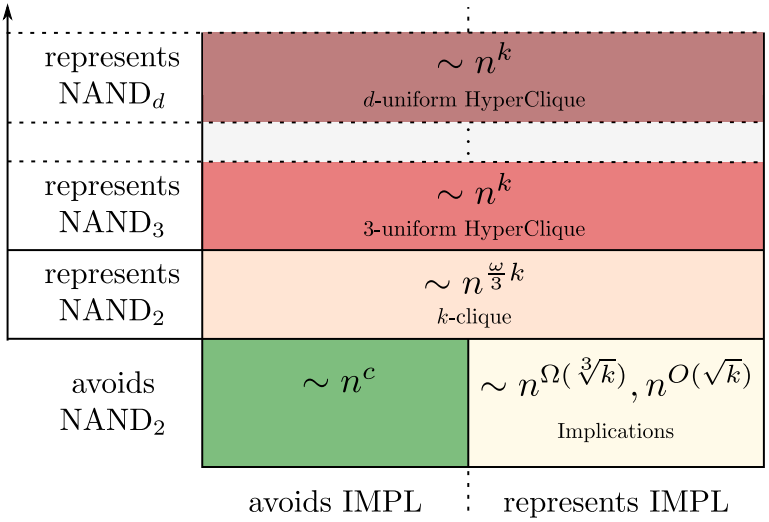

Figure 1 Overview over our main results. The parts of the diagram to the right of the vertical IMPL line depict $\mathcal{F}$ representing IMPL, while the parts to the left avoid IMPL. Analogously, the parts of the diagram above a NAND$_d$ line depict NAND$_d$-representing $\mathcal{F}$, while those below avoid NAND$_d$. For each cell, we illustrate our (typically matching) algorithmic and hardness results, together with a problem that is complete for this cell (in a certain sense). For clarity of presentation, we drop additional $f(k)n^c$-factors of stated running times.

1. **[FPT regime]**
   If $\mathcal{F}$ avoids both NAND$_2$ *and* IMPL*, then there is a computable $f(k)$ and constant $c_\mathcal{F}$ such that* SAT$(\mathcal{F})$ *can be solved in time $f(k)n^{c_\mathcal{F}}$.*
2. **[Subexponential regime]**
   If $\mathcal{F}$ represents IMPL*, but avoids* NAND$_2$*, then there is a computable $f(k)$ and constant $c_\mathcal{F}$ such that* SAT$(\mathcal{F})$ *can be solved in time $f(k)n^{4\sqrt{k}+c_\mathcal{F}}$;*
   furthermore, for no computable $f(k)$ and constants $c_\mathcal{F}, \epsilon > 0$, SAT$(\mathcal{F})$ *can be solved in time $f(k)n^{(\omega/6-\epsilon)\sqrt[3]{k}+c_\mathcal{F}}$, unless the k-clique conjecture fails.*
3. **[Clique regime]**
   If $\mathcal{F}$ represents NAND$_2$*, but avoids* NAND$_3$*, then there is a computable $f(k)$ and constant $c_\mathcal{F}$ such that* SAT$(\mathcal{F})$ *can be solved in time $f(k)n^{(\omega/3)k+c_\mathcal{F}}$;*
   furthermore for no computable $f(k)$ and constants $c_\mathcal{F}, \epsilon > 0$, SAT$(\mathcal{F})$ *can be solved in time $f(k)n^{(\omega/3-\epsilon)k+c_\mathcal{F}}$, unless the k-clique conjecture fails.*
4. **[Brute-force regime]**
   If $\mathcal{F}$ represents NAND$_3$*, then for no computable $f(k)$ and constants $c_\mathcal{F}, \epsilon > 0$,* SAT$(\mathcal{F})$ *can be solved in time $f(k)n^{(1-\epsilon)k+c_\mathcal{F}}$, unless the 3-uniform k-HyperClique conjecture fails.*

That is, we only have four regimes: $g(k)$ is either constant, sublinear in k with a value between essentially $(\omega/6)\sqrt[3]{k}$ and $4\sqrt{k}$, the clique detection bound of essentially $(\omega/3)k$, or the brute force bound of essentially k. Note that we do not try to optimize the bounds on $f(k)$, which generally are bounded by $r^{\mathcal{O}(k^3)}$, where r is the arity of $\mathcal{F}$.

Let us briefly discuss our hardness assumptions and their plausibility (for a detailed discussion, we refer to Section 2.1): The k-clique conjecture postulates that there is no $\mathcal{O}(n^{(\omega/3-\epsilon)k+c})$ time algorithm for detecting a k-clique in a given graph, with a matching upper bound of $\mathcal{O}(n^{(\omega/3)k+1})$ known since 1985 [29]. By now, it has been used, e.g., to justify (conditional) optimality of Valiant's parser for context free grammars [2] and to give

conditional lower bounds for string problems [10, 1], average-case settings [5], and more. Notably, the only k-clique algorithm known to break brute force by a polynomial factor makes crucial use of fast matrix multiplication techniques – unfortunately, these techniques do not extend to finding cliques in *hypergraphs*. This has led to the d-uniform HyperClique conjecture (for arbitrary $d \geq 3$): This conjecture states that there is no algorithm beating brute force, i.e., no $\mathcal{O}(n^{(1-\epsilon)k+c})$-time algorithm, for detecting a k-clique in a given d-uniform hypergraph. It has been used to expose hardness of problems in sparse graphs [27], for first-order queries to relational databases (specifically, in model-checking [9] and enumeration contexts [13]), and for the orthogonal vectors problem [3]; furthermore, it is known that its refutation requires giving a $\mathcal{O}((2-\epsilon)^n)$-time algorithm for Max-3SAT – we refer to [2, 27] for more detailed discussions of the plausibility of the (d-uniform Hyper-)Clique conjecture.

Interestingly, our classification does not fundamentally rely on the validity of the d-uniform HyperClique conjecture: If, for some $d \geq 3$, the d-uniform HyperClique conjecture is eventually refuted, we obtain faster-than-brute-force algorithms for all NAND_{d+1}-avoiding families!

Coarser Classification. While we state our results under very fine-grained hardness assumptions on clique and hyperclique detection, we may also state a coarser classification assuming only the assumption that k-clique cannot be solved in time $f(k)n^{o(k)}$. Already under this assumption, which is implied by the Exponential Time Hypothesis (see [14, 15]), our reductions and algorithms show that there exists an FPT regime where $g(k)$ is a constant, a subexponential regime where $g(k)$ is between $\Omega(\sqrt[3]{k})$ and $\mathcal{O}(\sqrt{k})$, and a linear regime where $g(k) = \Theta(k)$. However, based on the Exponential Time Hypothesis only, we cannot distinguish problems solvable in time $f(k)n^{(1\pm o(1))k}$ and $f(k)n^{(\omega/3\pm o(1))k}$, and thus cannot differentiate in the linear regime.

Examples. From our general classification, we can draw some interesting specific corollaries (assume here that k is a large constant):

- **3-SAT:** Finding satisfying assignments with k ones for 3-CNF formulas ($\mathcal{F}$ consists of all ternary functions with a single falsifying assignment) requires brute force time $n^{(1-o(1))k}$ under the 3-uniform HyperClique conjecture. However, if we drop a single function from $\mathcal{F}$ (specifically NAND_3, i.e., each constraint must have at most two negative literals), the problem can be solved in time $\mathcal{O}(n^{(\omega/3)k+c})$, which is essentially optimal under the k-Clique conjecture.
- **Subexponential cases:** We obtain $n^{\mathcal{O}(\sqrt{k})}$-time algorithms for interesting special cases: Beyond precedence-constrained SubsetSum with target k (i.e, $\mathsf{SAT}(\{\mathrm{IMPL}\})$), this includes $\mathsf{SAT}(\{f\})$ with $f(y_1, y_2, y_3) \coloneqq y_1 \Rightarrow (y_2 \vee y_3)$, and, more generally, every finite set of *dual-Horn constraints* (i.e., constraints that can be represented by clauses with at most a single negative literal)[4]. This also includes examples beyond dual-Horn constraints such as $\mathsf{SAT}(\{\mathrm{IMPL}, f'\})$ with f' being defined by $f'(y_1, y_2, y_3) = 1$ iff $(y_1, y_2, y_3) \in \{(0,0,0), (1,0,1), (1,1,0)\}$. Interestingly, all of these problems have the same (conditionally optimal) time complexity of $f(k)n^{\Theta(k^\alpha)}$ with $1/3 \leq \alpha \leq 1/2$; determining the precise value of α remains a challenge for future work.

[4] It is known that a constraint is dual-Horn if and only if it its satisfying assignments are closed under union, which immediately implies that it cannot contain NAND_2 as a restriction.

1.2 Technical Overview

We give an overview of the technical challenges that are handled in our work, from the highest running time regime to the lowest running time regime:

Brute-force regime. It is straightforward to obtain hardness for NAND_3-representing families by the following intuitive approach: To reduce from k-clique in a 3-uniform hypergraph G, we let x_i denote whether we include vertex v_i in our k-clique. By the standard observation that a clique in a hypergraph G is an independent set of its complement graph $\overline{G}$, we only need to ensure that for each edge $e = (v_a, v_b, v_c)$ of $\overline{G}$, not all vertices are included in our clique, i.e., $\mathrm{NAND}_3(x_a, x_b, x_c)$ holds. Since $\mathcal{F}$ represents NAND_3, we can express this constraint using an appropriate restriction of some $f \in \mathcal{F}$. Here, there is a technical issue of how we can generate the constants 0 or 1 to obtain the desired restrictions – using not particularly difficult, but careful constructions, we show that we can always simulate these constants as needed (Section 6).

Moderately hard regime. While the hardness of NAND_3-representing families is straightforward, it is surprising that this condition is in fact necessary for the brute-force approach to be (conditionally) optimal: If NAND_3 is *not* representable, we give a $f(k)n^{(\omega/3)k+c_{\mathcal{F}}}$-time algorithm via reduction to k-Clique.

The essential idea for this reduction is the following win-win argument. Let us denote by $a_{x,y}$ the weight-2 assignment setting only x and y to 1. Fix any weight-k satisfying assignment a. If there are two variables $x_i = x_{i'} = 1$ in a such that $a_{x_i,x_{i'}}$ is not satisfying, then we can use this pair of variables to "guide" our search towards a. We guess $x_i, x_{i'}$, identify a falsified constraint (of arity r) and guess an additional third variable from the at most r unguessed variables in this constraint. This means that by guessing two variables (n^2 possibilities), we obtain an additional variable almost for free (guessing r possibilities). That is, in the considered case we can identify 3 variables of a with a guess of rn^2 possibilities, which is a significant gain compared to the n^3 possibilities of brute force. Otherwise, if a has no such pair of variables, we observe that a satisfies already a simpler formula that uses only NAND_2's: specifically, the conjunction of $\mathrm{NAND}(x_i, x_{i'})$ for all i, i' such that assignment $a_{x_i,x_{i'}}$ violates the original formula. Furthermore, we show that since NAND_3 is not representable, any solution of the simpler formula indeed remains a solution of the original formula.

Interestingly, this reduction generalizes also to hypergraphs so that a refutation of the d-uniform HyperClique conjecture would give a $f(k)n^{(1-\epsilon)k+c_{\mathcal{F}}}$-time algorithm for NAND_{d+1}-avoiding families.

On the hardness side, analogously to the brute-force regime, it is rather straightforward to show that k-clique running time is indeed necessary for NAND_2-representing constraint families (see Section 6), which thus concludes a tight bound on $g(k)$ of essentially $(\omega/3)k$ in this regime.

Mildly hard regime. This is the technically most interesting regime. If NAND_2 is not representable, then $\mathsf{SAT}(\mathcal{F})$ might still not have an FPT algorithm, specifically, if it represents IMPL. Implicit in the $\mathsf{W}[1]$-hardness proof in [28] is a fine-grained lower bound of $n^{\Omega(\log k)}$ under the k-clique conjecture. By giving a careful adaptation of the lower bound of [28], we can strengthen this lower bound to $n^{\Omega(\sqrt[3]{k})}$. While it is conceivable that this lower bound can be strengthened to $n^{\Omega(\sqrt{k})}$, the structure of the construction suffers from a fundamental obstacle that makes a lower bound beyond $n^{\Omega(\sqrt{k})}$ seem unlikely. This raises the suspicion that

a $n^{o(k)}$-time algorithm for NAND_2-avoiding families could exist – and indeed, we manage to develop a $n^{\mathcal{O}(\sqrt{k})}$-time algorithm, which is perhaps the most interesting technical contribution of our paper.

To illustrate our approach, consider the problem Weighted DAG Implications: Given a DAG $G = (V, E)$ with node weights $w : V \to \mathbb{N}$ and a parameter $k \in \mathbb{N}$, the task is to find a set $S \subseteq V$ such that (1) $u \in S$ and $(u, v) \in E$ implies $v \in S$ and (2) S has total weight $\sum_{s \in S} w(s) = k$. Without edges, this problem simplifies to SubsetSum which we could solve in $\mathsf{poly}(k)$ time [23, 8]. However, to enable a generalization to our precedence setting, we describe a different approach based on a combinatorial property inspired by the famous Frobenius coin problem: Given coins of denominations $2 \leq d_1 < d_2 < \cdots < d_\ell$ with $\gcd(d_1, \ldots, d_\ell) = 1$, what is the largest number x not representable as $x = \sum_{i=1}^{\ell} \alpha_i d_i$ for some non-negative values $\alpha_i \geq 0$? A proof attributed to Schur (see [7, 30, 21]) yields an upper bound of $x \leq (d_1 - 1)(d_\ell - 1)$. Consequently, if $w_1 \leq \cdots \leq w_\ell$ with $\gcd(w_1, \ldots, w_\ell) \mid k$ are the weights occurring in an edgeless G, and $w_\ell \leq \sqrt{k}$, then there always exists a set S of total weight k, provided each weight occurs sufficiently often (say, at least k times). Thus, if we can preprocess the instance such that each weight is bounded by $\sqrt{k}$ and occurs sufficiently often, we can determine the answer to the instance by simply computing the gcd of the weights. Intuitively, this is possible in time $n^{\mathcal{O}(\sqrt{k})}$ by guessing the $\mathcal{O}(\sqrt{k})$ vertices of weight larger than $\sqrt{k}$, as well as brute-forcing vertices of each weight class containing only few vertices.

Interestingly, this approach can be lifted to the setting with precedence constraints. To this end, assume that the graph consists of layers $V_1, \ldots, V_\ell$ such that each V_i consists of a sufficiently large number of vertices of weight w_i and that all edges respect the layering (i.e., an edge between a vertex in V_i and a vertex in V_j implies $i > j$). We show the following property, which gives a generalization of Schur's bound to the precedence setting:

If for each vertex v, the total weight of its descendants (including v itself) is at most $\sqrt{k/2}$, then there exists a solution of total size k *if and only if* $\gcd(w_1, \ldots, w_\ell) \mid k$.

By an $n^{\mathcal{O}(\sqrt{k})}$-time preprocessing analogous to the intuitive arguments for the edge-less case, we can ensure that the preconditions are satisfied. We give the details of this approach in Section 3.

The above algorithmic insight solves the Weighted DAG Implications problem in time $\mathcal{O}(n^{4\sqrt{k}})$. To obtain such a bound for all NAND_2-avoiding families, we use a randomized reduction to Weighted DAG Implications. On a very high level, the approach is to create a Weighted DAG Implications instance G that contains only solutions that satisfy the given formula ϕ by iteratively choosing random implications consistent with certain solutions of ϕ. Doing this in an appropriate manner, a fixed feasible solution survives this process with $1/f(k)$ probability, which gives an algorithm running in time essentially $\mathcal{O}(f(k)n^{4\sqrt{k}})$. We give the details in Section 4.

Fast regime. For the remaining regime of families avoiding both IMPL and NAND_2, an $f(k)n^c$-time algorithm follows from [28], concluding the characterization.

1.3 Related work

Dichotomy theorems for constraint satisfaction have a rich history, starting with Schaefer's Theorem classifying Boolean Constraint Satisfaction Problems (CSPs) into either polynomial-time solvable or NP-complete [31]. The subsequent *Dichotomy conjecture* [20], which postulated that Schaefer's Theorem can be extended to any constant domain size beyond

Boolean, was resolved positively only recently by Bulatov [11] and Zhuk [34]. Further classifications have been investigated in a number of related settings, including quantified CSP (see, e.g., [18, 35]) and optimization variants (see, e.g. [17, 22]). Parameterizing by the solution size (as we do here), corresponding dichotomies have been obtained for Boolean [28] and larger domain sizes [12, 26], with a characterization of kernelization for Boolean domain given in [24] and a study of parameterized approximability given in [6]. A parameterized dichotomy for related local search tasks has been given in [25].

On a conceptual level, our work is related to a fine-grained classification result for model-checking first-order properties with a bounded number of quantifiers [9], where a fine-grained dichotomy under the 3-uniform HyperClique conjecture is given. Note, however, that the hardness criterion and techniques developed there are substantially different due to the different nature of the problem settings.

1.4 Open Problems

The main open problem raised by our work is to close the gap in the subexponential regime: Can we solve Implications $=$ $\mathsf{SAT}(\{\text{IMPL}\})$ already in $f(k)n^{\mathcal{O}(\sqrt[3]{k})}$ or can we improve our lower bound to $f(k)n^{\Omega(\sqrt{k})}$? Note that by our reductions, improved bounds directly transfer to all NAND_2-avoiding families.

Second, a natural direction is to extend our classification beyond the Boolean domain, i.e., give a fine-grained perspective building on [12, 26].

Finally, interesting related settings include natural problem variants with different size restrictions (*at most* k or *at least* k), local search tasks as well as optimization settings with weights on the variables or on the constraints.

2 Preliminaries

We write $[n] \coloneqq \{1, \dots, n\}$ and for any set S and integer d, let $\binom{S}{d}$ denote the set of d-element subsets of S.

For a finite constraint family $\mathcal{F}$, we say its *arity* r is the maximum arity of a function $f \in \mathcal{F}$. Since in the constraints of $\mathsf{SAT}(\mathcal{F})$, we may use variables in arbitrary order, we use the following notation for convenience: For any $f : \{0,1\}^r \to \{0,1\}$ and partition $X_1, \dots, X_s$ of $[r]$, we write

$$f(\overbrace{x_1 \dots x_1}^{X_1}, \dots, \overbrace{x_s \dots x_s}^{X_s})$$

to denote the value of $f(u_1, \dots, u_r)$ where we plug in x_j for each u_i with $i \in X_j$. Correspondingly $g : \{0,1\}^d \to \{0,1\}$ can be obtained as a restriction of f if and only if there is a partition $X_1, \dots, X_d, Z_0, Z_1$ of $[r]$ such that

$$g(x_1, \dots, x_d) = f(\overbrace{x_1 \dots x_1}^{X_1}, \dots, \overbrace{x_d \dots x_d}^{X_d}, \overbrace{0 \dots 0}^{Z_0}, \overbrace{1 \dots 1}^{Z_1}).$$

We say that an assignment $a : [n] \to \{0,1\}$ has *weight* k if $\sum_{i=1}^{n} a(i) = k$. Furthermore, we say that a is dominated by an assignment $a' : [n] \to \{0,1\}$, written $a \leq a'$, if for all $i \in [n]$, we have $a(i) \leq a'(i)$. For a subset $S \subseteq [n]$, we let a_S denote the assignment that sets $a(i) = 1$ if and only if $i \in S$. We let $\text{ones}(a) \coloneqq \{x_i \mid a(i) = 1\}$ denote the set of 1-variables of a. For any constraint $C = f(\mathbf{x})$ where $\mathbf{x} = (x_{i_1}, \dots, x_{i_r})$ with $i_1, \dots, i_r \in [n]$, we let $\text{vars}(C) = \{x_{i_1}, \dots, x_{i_r}\}$ denote the variable set involved in C.

All graphs considered in this paper are simple, i.e., we disallow multiple edges and self-loops. If $G = (V, E)$ is a directed graph, we call $S \subseteq V$ a *closed* set if for all $(u, v) \in E$, we have that $u \in S$ implies that $v \in S$. We say that v is a *descendant* of u if v is reachable by a path from u and let $D(u)$ denote the set of descendants of u (including u itself). Analogously, if v is a descendant of u, we call u an *ancestor* of v. We extend the notation naturally to sets $S \subseteq V$ by defining $D(S) \coloneqq \bigcup_{u \in S} D(u)$. For a graph $G = (V, E)$ with node weights $w : V \to \mathbb{N}$ and $S \subseteq V$, we write $w(S) \coloneqq \sum_{v \in S} w(v)$. For any $S \subseteq V$, we let $G[S]$ denote the subgraph of G induced by S, i.e., the subgraph obtained by deleting all vertices in $V \setminus S$ and adjacent edges.

2.1 Hardness Assumptions

Let k-clique denote the following problem: Given a (simple) undirected graph $G = (V, E)$, determine whether there is a *clique* of size k, i.e., $S \subseteq V, |S| = k$ such that for all $\{u, v\} \in \binom{S}{2}$ we have $\{u, v\} \in E$. A simple algorithm [29] solves k-clique in time $\mathcal{O}(n^{(\omega/3)k})$ when k is divisible by 3, which extends to time $\mathcal{O}(n^{\lfloor k/3 \rfloor \omega + (k \bmod 3)})$ for arbitrary k (for more precise bounds, see [19]). This running time is conjectured to be best possible, in the following sense.

▶ **Hypothesis 2.1** (k-Clique Conjecture). *For no $c, \epsilon > 0$ and $f(k)$, there is an $f(k)n^{(\omega/3-\epsilon)k+c}$-time algorithm for k-Clique.*[5]

As without the use of matrix multiplication, no $\mathcal{O}(n^{(1-\epsilon)k+c})$-time algorithms are known, a variant of the conjecture postulates that there are even no $\mathcal{O}(n^{(1-\epsilon)k+c})$-time *combinatorial* algorithms, i.e., algorithms avoiding the sophisticated algebraic techniques underlying current matrix multiplication algorithms.

By now, the k-clique conjecture has been used to explain hardness barriers in various contexts, such as the optimality of Valiant's parser for context-free grammar recognition [2], pattern matching in uncompressed and compressed strings [10, 1], average-case hardness [5] and more. For a more detailed discussion of this hardness assumption, we refer to [2].

The k-clique problem naturally extends to hypergraphs: Given a d-uniform hypergraph $G = (V, E)$, the d-uniform k-HyperClique problem asks to determine whether there is a *(hyper-)clique* of size k, i.e., $S \subseteq V, |S| = k$ such that for all subsets $S' \in \binom{S}{d}$, we have $S' \in E$.

▶ **Hypothesis 2.2** (d-Uniform k-HyperClique Conjecture). *Let $d \geq 3$. For no $c, \epsilon > 0$ and $f(k)$, there is an $f(k)n^{(1-\epsilon)k+c}$-time algorithm for d-uniform k-HyperClique.*

Similarly to the k-Clique conjecture, this hardness conjecture reveals hardness barriers in a number of contexts, such as hardness for problems on sparse graphs [27], for deciding or enumerating answers to first-order queries [9, 13] and for the study of fine-grained average-case complexity [5]. It is known that it implies the Orthogonal Vectors conjecture [3], however, refuting this conjecture requires (at least) to give an $\mathcal{O}((2-\epsilon)^n)$-time exact algorithm for Max3SAT; for details and further discussion of the plausibility of this conjecture, see [27].

[5] Note: sometimes, the k-clique conjecture is stated as

$$\inf\{F \mid 3k\text{-clique can be solved in time } n^{Fk+o(1)} \text{ for all (sufficiently large) constant } k\} = \omega,$$

which can be seen to be equivalent to the above formulation via a standard self-reduction for k-clique.

3 Algorithm for Implications

In this section, we give an algorithm for the problem Implications $=$ SAT($\{$IMPL$\}$) that is much faster than brute force and achieves $O(\sqrt{k})$ dependence of k in the exponent n. For convenience, we reduce Implications to the following problem. (Recall that for any graph $G = (V, E)$, we say that $S \subseteq V$ is *closed*, if for all $(u, v) \in E$, we have $u \in S$ implies $v \in S$.)

▶ **Problem 3.1** (Weighted DAG Implications)**.** *Given a DAG $G = (V, E)$ with node weights $w : V \to \mathbb{N}$ and parameter $k \in \mathbb{N}$, determine whether there is a closed set $S \subseteq V$ of weight exactly k, i.e., $w(S) = k$.*

The easy reduction works as follows. For each variable x_i, we introduce a corresponding vertex x_i of weight 1 and introduce an edge (x_i, x_j) for every implication constraint $x_i \Rightarrow x_j$ of ϕ. We contract each strongly connected component $C = \{v_1, v_2, \ldots, v_\ell\}$ in G to a single vertex v_C of weight $\sum_{i=1}^{\ell} w(v_i)$ in time $\mathcal{O}(n + m) = \mathcal{O}(n^2)$ [33]. Observe that the resulting graph is a DAG which has a closed set of weight k if and only if ϕ has satisfying assignment of weight k.

Recall that for any $v \in V$, we let $D(v)$ denote the set of descendants of v, i.e., the set of nodes reachable from v (including v).

As we will formally argue later, by a $f(k)n^{\mathcal{O}(\sqrt{k})}$-time preprocessing it is not difficult to preprocess a Weighted DAG Implications instance into the following form, which we call *Frobenius instance*, as it admits a combinatorial characterization of solvability that is analogous to Schur's bound for the Frobenius coin problem.

▶ **Definition 3.2.** *A* Frobenius instance with parameter k *is a weighted directed graph $G = (V, E, w)$ with ℓ parts $V = V_1 \cup V_2 \cup \cdots \cup V_\ell$ and weight function $w : V \to \mathbb{N}$ such that the following properties hold:*

(P1) *there are weights $w_1, \ldots, w_\ell$ such that $w(v) = w_i$ for all $v \in V_i$ and $i \in [\ell]$.*
(P2) *for any edge $(u, v) \in E$, we have $u \in V_i$ and $v \in V_j$ for some $\ell \geq i > j \geq 1$,*
(P3) *for all $i \in [\ell]$, we have $|V_i| \geq k$,*
(P4) *for all $v \in V$, we have $w(D(v)) \leq \sqrt{k/2}$.*

Intuitively, the necessary preprocessing follows from the following arguments: To ensure (P4), note that any weight-k closed set S has at most $\sqrt{2k}$ many vertices $v \in S$ with $w(D(v)) > \sqrt{k/2}$, which we can exhaustively enumerate with $n^{\mathcal{O}(\sqrt{k})}$-time overhead. By suitably arranging remaining nodes among the layers, it is straightforward to ensure (P1), (P2) and additionally that $\ell \leq f(k)$, since by (P4), each node has at most $\mathcal{O}(\sqrt{k})$ descendants. Finally, to ensure (P3), if any part V_i is small (i.e., $|V_i| < k$), we can exhaustively try out including any subset of V_i, introducing an overhead of only $2^{\mathcal{O}(k)}$ per V_i; since $\ell \leq f(k)$, this additional overhead is bounded by $f(k)2^{\mathcal{O}(k)}$.

If a Frobenius instance had no edges, then Schur's bound on the Frobenius coin problem implies that it has a solution if and only if $\gcd(w_1, \ldots, w_\ell) \mid k$. We prove that this criterion holds even in the setting of precedence constraints.

▶ **Lemma 3.3.** *Let G be a Frobenius instance with parameter k. Then G has a closed set of weight k if and only if $\gcd(w_1, \ldots, w_\ell) \mid k$.*

Proof. Since $\gcd(w_1, \ldots, w_\ell) \mid w(S)$ for any $S \subseteq V$, the condition $\gcd(w_1, \ldots, w_\ell) \mid k$ is necessary for G to have a closed set of weight k.

We show that this condition is also sufficient via induction on ℓ. In the base case $\ell = 1$, let $S \subseteq V_1 = V$ be an arbitrary subset of k/w_1 vertices (note that by $k/w_1 \leq k \leq |V_1|$, such a set indeed exists). By construction, S has weight $|S|w_1 = k$ and is closed, as G cannot contain any edges.

Thus let us assume that the claim holds for all $\ell' \leq \ell - 1$ and consider a Frobenius instance with $d' := \gcd(w_1, \ldots, w_\ell) \mid k$. Let $d := \gcd(w_1, \ldots, w_{\ell-1})$. We may assume that $d \nmid k$; otherwise, already the Frobenius instance $G[V_1 \cup \cdots \cup V_{\ell-1}]$ satisfies the assumption $\gcd(w_1, \ldots, w_{\ell-1}) \mid k$ and we obtain a closed set by inductive hypothesis.

Intuitively, we want to use the variables in V_ℓ to reach the target weight k modulo d; then we can reduce to a simpler instance where every weight (including the target weight) is divided by d. Note that we may assume

$$2 \leq d \leq \sqrt{k/2}, \tag{1}$$

where the lower bound follows from $d \nmid k$ and the upper bound follows from $d \leq \min_{i \in [\ell-1]} w_i \leq \sqrt{k/2}$, as $w(v) \leq w(D(v)) \leq \sqrt{k/2}$ for any $v \in V$.

Let b be the smallest non-negative integer such that $b \cdot w_\ell \equiv k \pmod{d}$. Such an integer exists and satisfies $b < d$: By Bézout's identity, since $\gcd(w_\ell, d) = d' \mid k$, there are coefficients β, γ such that $\beta w_\ell + \gamma d = k$, and thus any b with $b \equiv \beta \pmod{d}$ achieves the desired congruence.

Let $S \subseteq V_\ell$ be an arbitrary subset of size $b < d$; such a set indeed exists as $d \leq \sqrt{k/2} \leq k \leq |V_\ell|$. We observe that S satisfies

$$w(D(S)) \leq \sum_{s \in S} w(D(s)) \leq |S|\sqrt{k/2} \leq d\sqrt{k/2} \leq \frac{k}{2}, \tag{2}$$

where we used (P4) for the second inequality, and (1) for the last inequality. Consider the graph $G' = (V', E')$ obtained as a copy from G from which we delete $V_\ell \cup D(S)$ and define the node weights $w'(v') = w(v)/d$ for any $v \in V \setminus (V_\ell \cup D(S))$. We claim that G' is a Frobenius instance with parameter $k' := (k - w(D(S)))/d$ (observe that k' is indeed integer, as $w(D(S)) \equiv bw_\ell \equiv k \pmod{d}$, and that $k' \geq 0$ by (2)). If this is indeed the case, then by inductive hypothesis G' has a closed set S' with $w'(S') = k'$, since the gcd of the weights w' is 1. Observe that by construction, $D(S) \cup S'$ is a closed set in G of weight $w(D(S)) + d \cdot w'(S') = w(D(S)) + (k - w(D(S))) = k$, as desired.

It remains to prove that G' is indeed a Frobenius instance with parameter k'. First, observe G' has $\ell - 1$ layers $V'_i := V_i \setminus D(S), i \in [\ell - 1]$ and that w' is well defined, as $d \mid w_i$ for all $i \in [\ell - 1]$. Conditions (P1) and (P2) of being Frobenius are fulfilled as G' is a subgraph of G. To see (P3), note that

$$|V'_i| \geq |V_i| - |D(S)| \geq |V_i| - w(D(S)) \geq k - w(D(S)) \geq k'.$$

To see (P4), we observe that by (2) and (1), we have

$$k' = \frac{k - w(D(S))}{d} \geq \frac{k - k/2}{d} = \frac{k}{2d} \geq \frac{k}{d^2}.$$

Thus, for any $v' \in V'$, we obtain

$$w'(D(v')) \leq \frac{w(D(v))}{d} \leq \frac{\sqrt{k/2}}{d} = \sqrt{\frac{k}{2d^2}} \leq \sqrt{k'/2},$$

where we used condition (P4) of G in the second inequality. Thus, G' is indeed a Frobenius instance with parameter k', concluding the claim and thus the proof of our lemma. ◀

The above criterion is the main technical tool in the algorithmic result of the session. What remains is to show that the instance can be preprocessed in a way that it becomes a Frobenius instance.

▶ **Theorem 3.4.** *We can solve* Weighted DAG Implications *in time* $f(k)n^{4\sqrt{k}}$.

Proof. Consider the following recursive algorithm, which proceeds in 4 steps:

Step 1: For every $v \in V$ with $w(D(v)) \geq \sqrt{k/2}$, we return YES if a recursive call determines that $G[V \setminus D(v)]$ has a closed set of weight $k - w(D(v))$; otherwise, we delete v and all its ancestors from G. From now on, G satisfies $w(D(v)) \leq \sqrt{k/2}$ for all $v \in V$.

Step 2: We construct layers $L_1, \ldots, L_{\sqrt{k/2}}$ by the following iterative process: for every $i = 1, \ldots, \sqrt{k/2}$, we let L_i consists of all vertices in $V \setminus (L_1 \cup \cdots \cup L_{i-1})$ whose outgoing edges end in $L_1 \cup \cdots \cup L_{i-1}$. Note that $L_1, \ldots, L_{\sqrt{k/2}}$ partitions V; in particular, every vertex is included in some L_i, since if there was a vertex $v \in V \setminus (L_1 \cup \cdots \cup L_{\sqrt{k/2}})$, then by construction there exists a path from v containing strictly more than $\sqrt{k/2}$ vertices, leading to the contradiction $w(D(v)) \geq |D(v)| > \sqrt{k/2}$.

We observe that each layer L_i can be partitioned into sublayers $L_{i,j}, j \in \{1, \ldots, \sqrt{k/2}\}$ such that each $v \in L_{i,j}$ has weight $w(v) = j$: there can be no vertex of larger weight, as otherwise $w(D(v)) \geq w(v) > \sqrt{k/2}$ yields a contradiction. We consider layers $L_{i,j}$ in increasing lexicographic order of (i, j): If $|L_{i,j}| < k$, then for every $v \in L_{i,j}$, we return YES if a recursive call determines that $G[V \setminus D(v)]$ contains a closed set of size $k - w(D(v))$, and otherwise we delete v and all its ancestors from G. Observe that by the lexicographic ordering, we never delete vertices from already processed layers, so that at the end of the process, each $L_{i,j}$ is either empty or contains at least k vertices.

Step 3: We let $V_1, \ldots, V_\ell$ be an enumeration of all non-empty sublayers $L_{i,j}$ by the lexicographic order on (i, j) so that any vertex $v \in V_i$ has only edges to vertices in $V_1 \cup \cdots \cup V_{i-1}$. Observe that by construction, this yields a Frobenius instance. Let $w_1, \ldots, w_\ell$ be the weights of the Frobenius instance. We return YES if $\gcd(w_1, \ldots, w_\ell) \mid k$ and NO otherwise.

Using Lemma 3.3, the correctness of the algorithm is easy to see.

▷ Claim 3.5. The above algorithm is correct.

Proof. If the algorithm returns YES, indeed there is a closed set of size k: If we return YES in Steps 1 or 2, we have found a vertex v and a closed set S' in $G[V \setminus D(v)]$ of size $k - w(D(v))$, which yields a closed set $S' \cup D(v)$ in G of size k, as desired. Otherwise, we have arrived at a Frobenius instance and returned YES since $\gcd(w_1, \ldots, w_\ell) \mid k$, which implies that G has a closed set of size k by Lemma 3.3.

Conversely, fix a closed set S of size k, and we show that the algorithm returns YES: If S contains a vertex v investigated in Steps 1 or 2, then the recursive call to $G[V \setminus D(v)]$ (for the first such vertex v) will find a solution of size $|S| - w(D(v))$ (note that $D(v) \subseteq S$ if $v \in S$). Otherwise, we have arrived at a Frobenius instance which must satisfy $\gcd(w_1, \ldots, w_\ell)$ by Lemma 3.3, and we return YES. ◁

Finally, we need to bound the running time of the recursive algorithm. The analysis relies on the observation that the algorithm makes at most n recursive calls with a parameter decrease of at least $\sqrt{k/2}$, and at most $O(k^2)$ recursive calls with a parameter decrease of one.

▷ Claim 3.6. The above algorithm can be implemented in time $f(k)n^{4\sqrt{k}}$.

Proof. Let U be the set of vertices of small layers ($|L_{i,j}| < k$) considered in Step 2. We observe that the above algorithm can be implemented recursively with the following recurrence on its running time $T(n, k)$ on instances with n vertices and parameter k.

$$T(n,k) \leq \sum_{v \in V, w(D(v)) \geq \sqrt{k/2}} T(n, k - w(D(v))) + \sum_{u \in U} T(n, k - w(D(u))) + \mathcal{O}(n^2)$$

We claim by induction on k that this yields a bound of $T(n,k) \leq f(k)n^{4\sqrt{k}}$ for some $f(k) = k^{\mathcal{O}(k)}$. It is not difficult to see that for $k \leq 2$, we can solve the problem in time $\mathcal{O}(n^2) = \mathcal{O}(n^{4\sqrt{k}})$, yielding the base case. For $k \geq 3$, we thus obtain the following bound, using that in Step 2, we process less than k vertices for each "small" sublayer $L_{i,j}, 1 \leq i,j \leq \sqrt{k/2}$, i.e., $|U| \leq k(k/2) = k^2/2$,

$$\begin{aligned} T(n,k) &\leq \mathcal{O}(n \cdot f(k-\sqrt{k/2})n^{4\sqrt{k-\sqrt{k/2}}} + k^2 f(k-1)n^{4\sqrt{k-1}} + n^2) \\ &\leq (f(k)/2)(n^{4\sqrt{k-\sqrt{k/2}}+1} + n^{4\sqrt{k}}) \leq f(k)n^{4\sqrt{k}}, \end{aligned}$$

where the second bound follows from choosing $f(k) = k^{\mathcal{O}(k)}$ large enough to ensure $k^2 f(k-1) \leq f(k)/2$ and the last bound follows from the observation that $4\sqrt{k-\sqrt{k/2}}+1 \leq 4\sqrt{k}$ if and only if

$$\begin{aligned} & \left(4\sqrt{k-\sqrt{k/2}}+1\right)^2 \leq 16k \\ \iff \quad & 16(k-\sqrt{k/2}) + 8\sqrt{k-\sqrt{k/2}} + 1 \leq 16k \\ \iff \quad & 8\sqrt{k-\sqrt{k/2}} + 1 \leq 16\sqrt{k/2}, \end{aligned}$$

where the last inequality holds since $8\sqrt{k}+1 \leq 16\sqrt{k/2}$ as $k \geq 3$. ◁

Claims 3.5 and 3.6 show the correctness of our algorithm for Weighted DAG Implications. By the reduction described at the beginning of the section, a similar algorithm follows for Implications. ◀

4 Algorithms for NAND_2-avoiding $\mathcal{F}$: Reduction to Implications

In this section, we show that for any NAND_2-avoiding constraint family $\mathcal{F}$, we can reduce $\mathsf{SAT}(\mathcal{F})$ to Implications. Specifically, we obtain the following theorem.

▶ **Theorem 4.1.** *Let $\mathcal{F}$ be a NAND_2-avoiding constraint family and let $T_{\mathrm{IMPL}}(n,k)$ denote the optimal running time to solve Implications. There is a constant $c_{\mathcal{F}}$ and computable $f(k)$ such that we can solve $\mathsf{SAT}(\mathcal{F})$ in time $f(k)(T_{\mathrm{IMPL}}(n,k) + n^{c_{\mathcal{F}}})\log n$.*

Together with Theorem 3.4, this gives an $f(k)n^{4\sqrt{k}+c_{\mathcal{F}}}$-time algorithm for any NAND_2-avoiding constraint family $\mathcal{F}$.

To prove the above theorem, we prepare some notation and helpful facts. Let ϕ be an arbitrary formula. For any assignment a, we call a' a *minimal satisfying extension* of a, if a' satisfies ϕ, $a \leq a'$, and no other satisfying assignment $a'' \notin \{a, a'\}$ fulfills $a \leq a'' \leq a'$. The following lemma shows that there are only $f(k)$ many minimal extensions of weight at most k, and these minimal extensions can be computed in time $f(k)n^c$ for some constant c independent of k. Intuitively, this follows by using the bounded search tree technique over violated constraints, where the depth of the search tree is bounded by k and each branching step has at most r possibilities.

▶ **Lemma 4.2** ([12, Lemma 2.3])**.** *Let $\mathcal{F}$ be a finite constraint family of bounded arity r. There is a constant $c'_{\mathcal{F}}$ such that given any instance ϕ of $\mathsf{SAT}(\mathcal{F})$ and assignment a, there are at most $\mathcal{O}(r^k)$ minimal extensions of a of weight k, and we can compute these extensions in time $\mathcal{O}(r^k n^{c'_{\mathcal{F}}})$.*

As an immediate useful consequence, we obtain that for our algorithmic results, we may assume without loss of generality that $\mathcal{F}$ is 0-valid, i.e., each $f \in \mathcal{F}$ is satisfied by the all-zeroes assignment.

▶ **Corollary 4.3** (see also [28, Lemma 4.1])**.** *We can reduce any instance of* $\mathsf{SAT}(\mathcal{F})$ *with parameter* k *to* $\mathcal{O}(r^k)$ *many instances of* $\mathsf{SAT}(\mathcal{F}')$ *with a parameter bounded by* k*, where* $\mathcal{F}'$ *is the set of all* 0*-valid* f' *that are represented by* $\mathcal{F}$*.*

By definition, if $\mathcal{F}$ does not represent NAND_2, then also $\mathcal{F}'$ does not represent NAND_2, and it remains to give an $f(k)(T_{\mathrm{IMPL}}(n,k) + n^{c_{\mathcal{F}}}) \log n$-time algorithm for 0-*valid* NAND_2-avoiding $\mathcal{F}$.

In the remainder of this section, we will use the graph formulation of the Implications problem: We are given a directed graph $G = (V, E)$ and the task is to find a closed set S (recall that S is closed, if for all $(u, v) \in E$ we have that $u \in S$ implies $v \in S$) of size k. Recall that for any vertex set $S \subseteq V$, $D(S)$ denotes the set of descendants of any vertex $s \in S$ (including the vertices in S).

Our aim is the following: Given a formula ϕ of $\mathsf{SAT}(\mathcal{F})$, we give a randomized construction of an Implications instance G such that

- **(i)** any closed set S in G corresponds to a satisfying assignment of ϕ, and
- **(ii)** with large enough probability, G contains a closed set of size k if ϕ has a weight-k solution.

To this end, we let $V = \{x_1, \ldots, x_n\}$ and recall that, for any set $S \subseteq V$, we let $a_S : [n] \to \{0, 1\}$ denote a corresponding assignment with $a_S(i) = 1$ iff $x_i \in S$. From now on, we often synonymously speak of closed sets $S \subseteq V$ in G and the corresponding assignment a_S for ϕ.

The rough outline is as follows: we start with the graph $G = (V, \emptyset)$, and try to repeatedly "fix" some closed set S that violates ϕ, by determining a (random) implication consistent with a minimal satisfying extension of S. The main insight is that if $\mathcal{F}$ avoids NAND_2, then it suffices to make sure that all sets $D(v)$ for $v \in V$ are satisfying and this will automatically ensure that every closed set is satisfying.

Let us formally describe the algorithm:

1. Given ϕ, initialize $G = (V, E)$ with $V = \{x_1, \ldots, x_n\}$ and $E = \emptyset$.
2. While there exists some $v \in V$ such that $a_{D(v)}$ violates ϕ, do the following:
   - **a.** Compute the set A_v of minimal satisfying extensions of $a_{D(v)}$ of weight at most k.
   - **b.** Let X consist of all $x_i \in V \setminus D(v)$ such that there is some $a \in A_v$ with $a(i) = 1$.
   - **c.** If $X = \emptyset$, delete all ancestors of v (including v) from G. Otherwise, pick x uniformly at random from X and add the edge (v, x) to E.

The important properties of the algorithm are captured in the following lemma.

▶ **Lemma 4.4.** *Let* $\mathcal{F}$ *be a finite* 0*-valid constraint family. There is a constant* $c_{\mathcal{F}}$ *and a function* $g(k)$ *such that the following properties hold.*

- **(P1)** *During the process, each vertex* v *is considered at most* k *times in the while loop. Thus, the algorithm can be implemented to run in time* $\mathcal{O}(g(k)n^{c_{\mathcal{F}}})$*.*
- **(P2)** *If* ϕ *has a satisfying assignment of weight* k*, then with probability at least* $g(k)^{-1}$*, there is a closed set* S *in* G *of size* k*.*
- **(P3)** *If* $\mathcal{F}$ *avoids* NAND_2*, any closed set* $S \subseteq V$ *in the constructed graph yields a satisfying assignment* a_S *for* ϕ*.*

Proof. For (P1), note that whenever $v \in V$ is considered in the while loop, it is either deleted, or an edge (v, x) with $x \notin D(v)$ is added to the graph. Thus, when v is considered for the k-th time, we have $|D(v)| \geq k$, and thus there can be no satisfying extension of $a_{D(v)}$

of weight at most k. Consequently, we must have $A_v = \emptyset$, and thus $X = \emptyset$, which forces v to be deleted. Thus, we have at most kn iterations of the while loop, where each iteration can be implemented in time $\mathcal{O}(r^k n^{c'_{\mathcal{F}}})$ by Lemma 4.2.

For (P2), assume that there is a set S of size k such that a_S satisfies ϕ. We show that with large enough probability, we will maintain as invariant that $D(v) \subseteq S$ for every $v \in S$, and thus S will be a closed set in G. To this end, we first observe that for $D(v) \subseteq S$ to hold for all $v \in S$, it suffices that the following property holds:

In each iteration that considers a vertex $v \in S$, the selected vertex x is in S. (3)

Indeed, if this is the case, then no $v \in S$ is ever deleted. Furthermore, we have that $D(v) \subseteq S$ for all $v \in S$, and thus S is a closed set in G. It remains to give a lower bound on the probability that (3) holds throughout the process.

To this end, consider the event that some $v \in V$ is considered in the while loop, conditioned that (3) has not been violated in a previous iteration. Under this event, $D(v) \subseteq S$, and thus there is a minimal satisfying extension $D(v) \subsetneq S' \subseteq S$ such that $a_{S'}$ satisfies ϕ and thus $a_{S'} \in A_v$. Let $s \in S' \setminus D(v)$ be arbitrary, then $s \in X$ by construction (note that s has not been deleted). By Lemma 4.2, we have that $|A_v| \leq \mathcal{O}(r^k)$. Since each $a \in A_v$ has weight at most k, this yields $|X| \leq k|A_v| \leq \mathcal{O}(kr^k)$. Thus, the probability that the random choice is $x = s$ is at least $1/|X| \geq \Omega(1/(kr^k))$. Finally, we observe that by (P1), for each $v \in S$, there are at most k iterations considering v, where each iteration has a probability of at least $\Omega(1/(kr^k))$ of not violating (3). Thus, we obtain that (3) holds with probability at least $\Omega(1/(kr^k)^{k|S|}) = \Omega(1/(kr^k)^{k^2})$, and the claim follows by setting $g(k) \coloneqq (kr^k)^{-k^2}$.

Finally, for (P3), note that at the end of the process, the property holds that

For all (remaining) $v \in V, a_{D(v)}$ satisfies ϕ. (4)

We will leverage this fact to show that a_S satisfies ϕ for *all* closed sets $S = D(v_1) \cup ... \cup D(v_\ell)$ for $v_1, \ldots, v_\ell \in V$. We first transform the graph G to a DAG by contracting all strongly connected components $C = \{v_1, \ldots, v_c\}$ to a single vertex v_C representing the set C. Note that the closed sets in the DAG remain in a one-to-one correspondence to the closed sets of the original graph (and the corresponding assignments to ϕ), thus this transformation is without loss of generality. Thus, we may assume that G has a topological ordering $v_1, \ldots, v_{n'}$ of its vertices ($n' \leq n$). We will prove by induction on $i = n', ..., 1$ that for all closed sets $S \subseteq \{v_i, ..., v_{n'}\}$, a_S satisfies ϕ.

For the base case $i = n'$, we only need to verify that (i) the all-0 assignment satisfies ϕ, which holds by 0-validity of $\mathcal{F}$, and (ii) that $a_{v_{n'}}$ satisfies ϕ, which holds by (4) (as $D(v_{n'}) = \{v_{n'}\}$). Thus, for $i < n'$, let us assume that the claim holds for $i+1$. Consider any closed set $U \subseteq \{v_i, \ldots, v_{n'}\}$. If U does not contain v_i, the claim follows by inductive assumption, thus let us assume that $v_i \in U$ and thus $U \supseteq D(v_i)$, as U is closed. If $U = D(v_i)$, a_U satisfies ϕ by (4). Thus, it remains to consider $U \supsetneq D(v_i)$, for which we assume for contradiction that a_U violates ϕ. Let $W \coloneqq U \setminus D(v_i)$, and note that $D(W) \subseteq U$ is a closed set in $\{v_{i+1}, \ldots, v_{n'}\}$. Thus, by inductive assumption, $a_{D(W)}$ satisfies ϕ. Furthermore, observe that $Z \coloneqq D(v_i) \cap D(W)$ is a closed set in $\{v_{i+1}, \ldots, v_{n'}\}$ (since the intersection of any two closed sets yields a closed set). Thus, a_Z satisfies ϕ by inductive assumption. It remains to show that the fact that $a_{D(v_i)}$, $a_{D(W)}$ and $a_Z = a_{D(v_i) \cap D(W)}$ all satisfy ϕ, while $a_U = a_{D(v_i) \cup D(W)}$ violates ϕ, gives a contradiction to $\mathcal{F}$ avoiding NAND_2.

To this end, let C be a constraint violated by a_U and note that $C = f(x_{i_1}, \ldots, x_{i_r})$ for some $f \in \mathcal{F}$ and $i_1, \ldots, i_r \in [n]$. Note that we can view f as $f : \{0,1\}^{V_c} \to \{0,1\}$ for some appropriate variable set V_C. We show how to obtain NAND_2 as a restriction of f by partitioning V_C into $X' \coloneqq (D(v_i) \setminus Z) \cap V_C, Y' \coloneqq (D(W) \setminus Z) \cap V_C, Z_1 \coloneqq Z \cap V_C, Z_0 \coloneqq V_C \setminus (X' \cup Y' \cup Z_1)$ and observing that

$$
\begin{array}{llllcll}
 & \overbrace{}^{X'} & \overbrace{}^{Y'} & \overbrace{}^{Z_0} & \overbrace{}^{Z_1} & & \\
f(0\ldots0, & 0\ldots0, & 0\ldots0, & 1\ldots1) & = & 1, & \text{[since } a_Z \text{ satisfies } C\text{]} \\
f(1\ldots1, & 0\ldots0, & 0\ldots0, & 1\ldots1) & = & 1, & \text{[since } a_{D(v_i)} \text{ satisfies } C\text{]} \\
f(0\ldots0, & 1\ldots1, & 0\ldots0, & 1\ldots1) & = & 1, & \text{[since } a_{D(W)} \text{ satisfies } C\text{]} \\
\hline
f(1\ldots1, & 1\ldots1, & 0\ldots0, & 1\ldots1) & = & 0. & \text{[since } a_{D(W)\cup D(v_i)} \text{ violates } C\text{]}
\end{array}
$$

◀

It remains to give the proof of Theorem 4.1.

Proof of Theorem 4.1. By Corollary 4.3, we may assume without loss of generality that $\mathcal{F}$ is 0-valid. We repeat the following process $g(k)$ many times: We use the above algorithm to generate an Implications instance G, and return YES if G contains a closed set of size k, which we determine using an optimal Implications algorithm. If none of the $g(k)$ iterations were successful, we return NO. Note that this approach can be implemented in time $g(k)\mathcal{O}(g(k)n^{c_{\mathcal{F}}} + T_{\text{IMPL}}(n,k))$ by (P1), and correctly decides the instance with probability at least $1-(1-1/g(k))^{g(k)} \geq 1-1/e$ by (P2) and (P3).

The algorithm described above can be derandomized using the standard technique of Color Coding [4]. In each iteration when vertex v is considered, a random vertex x is selected from a set X of at most $K = \mathcal{O}(kr^k)$ vertices. As each vertex is considered at most k times, we can represent the random choices by a function $r : V \to [K]^k$, with the meaning that $r(v)$ is the vector of choices made when considering vertex v. As discussed in the proof of Lemma 4.4, when considering vertices $v \in S$, these random choices need to be consistent with S to ensure that S is a closed set in the resulting graph. That is, for each $v \in S$ there is a vector $c(v) \in [K]^k$ such that if the random choice satisfies $r(v) = c(v)$ for every $v \in S$, then S is a closed set.

We say that a family $\mathcal{H}$ of functions $h : [n] \to [k]$ is a (n,k)-*perfect family of hash functions* if for every $S \subseteq V$ of size k, there is an $h \in \mathcal{H}$ that is injective on S, i.e., assigns different values to different elements of S. It is known that a (n,k)-perfect family of size $2^{O(k)}\log n$ can be computed in time $2^{O(k)}n\log n$ [4]. The derandomized algorithm would first compute such a family $\mathcal{H}$ over V and would iteratively go through every $h \in \mathcal{H}$ and function $q : [k] \to [K]^k$. For a given choice of h and q, we define the function $r(v) = q(h(v))$ and run the randomized algorithm using this function r instead of the random choices. It is easy to see that the definition of (n,k)-perfect hash functions implies that there is at least one choice of h and q where $r(v)$ is exactly the prescribed value $c(v)$ for every $v \in S$ and therefore the randomized algorithm correctly finds the solution S. As we are considering at most $|\mathcal{H}| = 2^{O(k)}\log n$ functions h and K^{k^2} different functions q, there is a function $f(k)$ such that the total running time is at most $f(k)\log n$ times a single run of the randomized algorithm. ◀

5 Algorithms for NAND-representing $\mathcal{F}$: Reduction to Clique

In this section, we develop algorithm for constraint families that might represent NAND_2, but avoid NAND_d for some $d \geq 3$. To this end, we give a reduction to $(d-1)$-uniform Hyper-Clique for NAND_d-avoiding families, giving in particular a $f(k)n^{(\omega/3)k+c_{\mathcal{F}}}$-time algorithm for NAND_3-avoiding families.

We first start with a natural reduction of $\mathsf{SAT}(\mathcal{F})$ for any $\mathcal{F}$ with arity bounded by r to r-uniform HyperClique, based on color-coding. To this end, let $T_{d\text{-HC}}(n,k)$ denote the optimal running time of finding a k-clique in a d-uniform hypergraph.

▶ **Proposition 5.1.** *Let $\mathcal{F}$ be a constraint family of arity at most r. Then $\mathsf{SAT}(\mathcal{F})$ can be solved in time $f(k)(n^{2r} + T_{r\text{-}HC}(n,k))\log n$.*

Proof. Let ϕ be an arbitrary $\mathsf{SAT}(\mathcal{F})$ formula. Observe that any constraint C of ϕ depends only on a set $\mathrm{vars}(C) \subseteq \{x_1, \ldots, x_n\}$ of at most r variables. For an assignment a, we let $C(a) \in \{0,1\}$ denote whether C is satisfied by a.

We first show how to determine, given a partition of $x_1, \ldots, x_n$ into k sets $X_1, \ldots, X_k$, whether there is a solution that sets precisely one variable in each X_i to true. To this end, we construct a hypergraph G with vertex set $X_1 \cup X_2 \cup \cdots \cup X_k$ and the following set of hyperedges: we include each possible hyperedge $e = \{x_{j_1}, \ldots, x_{j_r}\}$ with $x_{j_1} \in X_{j_1}, \ldots, x_{j_r} \in X_{j_r}$ and distinct $j_1, \ldots, j_r \in [k]$ *unless* there exists a clause C with $\mathrm{vars}(C) \subseteq X_{j_1} \cup \cdots \cup X_{j_r}$ which is violated by the assignment that sets precisely the variables $e = \{x_{j_1}, \ldots, x_{j_r}\}$ to 1, i.e., $C(a_e) = 0$.

We claim that $H := \{x_{i_1}, \ldots, x_{i_k}\}$ with $x_{i_1} \in X_1, \ldots, x_{i_k} \in X_k$ yields a k-clique in G if and only if the assignment a_H satisfies ϕ. Indeed, assume that there is a clause C violated by a_H. Note that as C has arity at most r, we have $\mathrm{vars}(C) \subseteq X_{j_1} \cup \cdots \cup X_{j_r}$ for some distinct $i_1, \ldots, i_r \in [k]$ (if C involves variables of less than r sets, we may use arbitrary additional sets). Thus, $e := \{x_{j_1}, \ldots, x_{j_r}\}$ cannot be an edge in G, since a_H violates C, a_e and a_H agree on $\mathrm{vars}(C)$, and thus also a_e violates C. Conversely, if there is some $e := \{x_{i_1}, \ldots, x_{i_r}\}$ with distinct $i_1, \ldots, i_r \in [k]$ such that e is not an edge in G, then there exists some clause C with $\mathrm{vars}(C) \subseteq X_{i_1} \cup \cdots \cup X_{i_r}$ which is violated by a_e. Since a_H and a_e agree on $\mathrm{vars}(C)$, we conclude that also a_H violates C and thus ϕ.

To create the desired k-partition of variables, we use a (deterministic) color-coding scheme: Let $\mathcal{H}$ be a (n,k)-*perfect family of hash functions* $h : [n] \to [k]$ – recall that this means that for any $S = \{s_1, \ldots, s_k\} \subseteq [n]$, there exists some $h \in \mathcal{H}$ such that $\{h(s_1), \ldots, h(s_k)\} = \{1, \ldots, k\}$. Known efficient constructions [32, 4] produce such assignments with $\ell = 2^{\mathcal{O}(k)} \log(n)$ in time $2^{\mathcal{O}(k)} n \log n$. Given this family, we create for each $h \in \mathcal{H}$ the k-partition $X_1^{(h)}, \ldots, X_k^{(h)}$ with $X_j^{(h)} = \{x_s \mid h(s) = j\}$ and solve the corresponding r-uniform HyperClique instance in time $T_{r\text{-HC}}(n,k)$. If any of these instances returns a solution, then indeed ϕ has a satisfiable assignment of weight k. Conversely, if a_S is a weight-k satisfying assignment for ϕ, then by construction, there exists a hash function $h \in \mathcal{H}$ such that $|S \cap X_j^{(h)}| = 1$ for $j = 1, \ldots, k$, and thus the corresponding r-uniform HyperClique instance indeed contains a solution. For each of the $2^{\mathcal{O}(k)} \log(n)$ hash functions, the time to construct and solve the d-uniform HyperClique instance is bounded by $\mathcal{O}(n^{2r} + T_{r\text{-HC}}(n,k))$, concluding the claim. ◀

The main result in this section is the following reduction from NAND_{d+1}-avoiding constraint families to d-uniform HyperClique.

▶ **Theorem 5.2.** *Let $d \geq 2$ and $\mathcal{F}$ be an NAND_{d+1}-avoiding constraint family. If there are constants $\gamma \geq d/(d+1)$ and c, and a computable $g(k)$ such that d-uniform HyperClique can be solved in time $g(k)n^{\gamma k + c}$, then there is a constant c' and computable $g'(k)$ such that $\mathsf{SAT}(\mathcal{F})$ can be solved in time $g'(k)n^{\gamma k + c'}$.*

In particular, since we can find k-cliques in graphs in time $\mathcal{O}(n^{\frac{\omega}{3}k+1})$, we obtain an $g(k)n^{\frac{\omega}{3}k+c'}$-time algorithm for solving $\mathsf{SAT}(\mathcal{F})$ for all NAND_3-avoiding constraint families. Similarly, if for $d \geq 3$ the d-uniform HyperClique conjecture is refuted by exhibiting a $g(k)n^{(1-\epsilon)k+c}$-time algorithm for some constants $0 < \epsilon < 1/(d+1)$ and c, we would obtain a $g'(k)n^{(1-\epsilon)k+c'}$-time algorithm for $\mathsf{SAT}(\mathcal{F})$ for NAND_{d+1}-avoiding families $\mathcal{F}$.

In the remainder of the section, we give the proof of Theorem 5.2. The main task of the algorithm is to detect *robust* assignments, defined as follows.

▶ **Definition 5.3.** *Let $a : [n] \to \{0,1\}$ be a weight-k assignment that satisfies ϕ. We say that a is* d-robust *if there is no assignment $a' \leq a$ of weight at most d that violates ϕ.*

The first step of the algorithm is the easier task of detecting satisfying assignments that are *not* d-robust (if there exists any): Intuitively, an assignment that is not d-robust offers an advantage to find it: Assume we correctly guess an assignment $a' \leq a$ of weight $w \leq d$ such that some clause C is violated by a', then to extend a' to the satisfying assignment a, we know that at least one additional variable in C must be set to true. By bruteforcing over the at most $r - w \leq r$ many possibilities, we gain an advantage. Specifically, by enumerating $O(n^w r) = \mathcal{O}(n^w)$ many possibilities, we can fix $w + 1$ true variables in our solution.

Let $T(n, k)$ denote the time our algorithms takes to solve an arbitrary $\mathsf{SAT}(\mathcal{F})$ instance for a NAND_{d+1}-avoiding family $\mathcal{F}$. In a preprocessing step, we first enumerate all assignments a' of weight at most d. If there exists a clause $C_{a'}$ that is violated by a', then we enumerate all variables $x \in \mathrm{vars}(C_{a'}) \setminus \mathrm{ones}(a')$ (recall that $\mathrm{vars}(C)$ is the set of variables involved in C and $\mathrm{ones}(a)$ denotes the set of variables set to 1 under a). We recursively determine satisfiability of the formula $\phi_{a',x}$ obtained by restricting all variables in $\mathrm{ones}(a') \cup \{x\}$ to true. Disregarding the time to determine existence of violated clauses $C_{a'}$, this step takes time

$$\sum_{w=0}^{d} \sum_{\substack{\text{weight-}w \\ \text{assignment } a'}} \sum_{x \in \mathrm{vars}(C_{a'}) \setminus \mathrm{ones}(a')} T(n, k-(w+1)) \leq \sum_{w=0}^{d} O(n^w) T(n, k-(w+1)). \quad (5)$$

To determine a violated clause $C_{a'}$ (if it exists) for all weight-($\leq d$) assignments a', we simply traverse each clause C, determine the at most $\sum_{w=0}^{d} \binom{r}{w} = O(1)$ weight-($\leq d$) assignments violating C and store C as violated for each of these assignments (if no other clause is already stored). This step takes time $O(m) = O(n^r)$ in the beginning.

After this preprocessing, it remains to consider d-robust assignments. To determine whether a d-robust assignment satisfies ϕ, we define a formula ϕ_d that is satisfied only by satisfying assignments of ϕ, and particularly by all d-robust satisfying assignments of ϕ. To this end, let F_d contain all assignments of weight at most d that violate some clause C of ϕ, and define

$$\phi_d := \bigwedge_{a \in F_d} \mathrm{NAND}(\mathrm{ones}(a)).$$

▶ **Lemma 5.4.** *The constructed formula ϕ_d has the following properties:*

(P1) *If $\mathcal{F}$ is NAND_{d+1}-avoiding, then any satisfying assignment a of ϕ_d is a satisfying assignment of ϕ.*

(P2) *If a is a d-robust satisfying assignment of ϕ, then a satisfies ϕ_d.*

Proof. To prove (P1), we will make use of the following property.

▷ Proposition 5.5. Let $\mathcal{F}$ be a NAND_{d+1}-avoiding family. Then if an assignment a violates some clause C (chosen from $\mathcal{F}$), there is an assignment $a' \leq a$ of weight at most d that violates C.

Proof. We prove the claim via induction on the weight w of the clause C under a. If $w \leq d$, the claim trivially holds. To prove the inductive step, we may assume for contradiction that there is an assignment a of weight $w \geq d+1$ violating some clause $C = f(\bar{x})$, but no assignment $a' \leq a$ of weight at most $w - 1$ violates C. We will show that NAND_{d+1} can be obtained as a restriction of f. To this end, choose some set $S \subseteq \mathrm{ones}(a) \cap \mathrm{vars}(C)$ of size $d+1$ (which is possible as $w \geq d+1$), and partition $\mathrm{vars}(C)$ into S, $Z_1 := (\mathrm{ones}(a) \cap \mathrm{vars}(C)) \setminus S$ and $Z_0 := \mathrm{vars}(C) \setminus (S \cup Z_1)$. Observe that we have

$$\begin{array}{lllll} f(\overbrace{y_1 \dots y_{d+1}}^{S}, & \overbrace{0 \dots 0}^{Z_0}, & \overbrace{1 \dots 1}^{Z_1}) & = 1, & \text{if } (y_1, \dots, y_{d+1}) \neq (1, \dots, 1) \\ \hline f(\ 1 \dots 1\ \ , & 0 \dots 0, & 1 \dots 1) & = 0. & [\text{since } a \text{ violates } C] \end{array}$$

where the first line follows since no assignment $a' \leq a$ of weight at most $w-1$ violates C, yielding a contradiction. ◁

To prove (P1), assume that an assignment a violates some clause C of ϕ. Since $\mathcal{F}$ is NAND_{d+1}-avoiding, by Proposition 5.5 there exists an assignment $a' \leq a$ of weight at most d such that a' violates C. Thus, ϕ_d contains a clause $\text{NAND}(\text{ones}(a'))$, which is violated by a, as $a' \leq a$.

To prove (P2), assume for contradiction that a d-robust assignment a satisfies ϕ but not ϕ_d. Then there is some $a' \in F_d$ such that $\text{NAND}(\text{ones}(a'))$ is violated by a, i.e., $a' \leq a$. As $a' \in F_d$, there must be a clause C of ϕ that is violated by $a' \leq a$, which proves that a is not d-robust and thus yields a contradiction. ◀

Note that ϕ_d is a $\mathsf{SAT}(\mathcal{F}')$ formula with constraint family $\mathcal{F}' = \{\text{NAND}_j \mid 2 \leq j \leq d\}$ of arity d. Thus, by Proposition 5.1, we can determine satisfiability of ϕ_d in time $f(k)(n^{2d} + T_{d\text{-HC}}(n,k)) \log n$. We obtain the following recurrence by combining (5), the $O(m)$-time preprocessing to determine violated classes $C_{a'}$, and $f(k)(n^{2d} + T_{d\text{-HC}}(n,k)) \log n$ to solve ϕ_d:

$$T(n,k) = \mathcal{O}(m) + f(k)(n^{2d} + T_{d\text{-HC}}(n,k)) \log n + \sum_{w=0}^{d} O(n^w) T(n, k-(w+1)) \tag{6}$$

Assume that there are $\gamma \geq d/(d+1)$ and c such that $T_{d\text{-HC}}(n,k) \leq g(k) n^{\gamma k + c}$. We will show that $T(n,k) = g'(k)\mathcal{O}(n^{\gamma k + c'})$ for any $c' > \max\{c, 2r\}$ and $g'(k) = f(k)g(k)$.

We prove the claim via induction on k. The base case is $k < c'$, in which case we can solve $\mathsf{SAT}(\mathcal{F})$ in time $f(k)(n^{2r} + T_{r\text{-HC}}(n,k)) \log n = f(k)\mathcal{O}((n^{2r} + n^k) \log n) \leq \mathcal{O}(n^{c'})$, satisfying the claim. Thus, let us assume that $k \geq c'$ and that the claim holds for all $k' \leq k-1$. Using (6), we obtain

$$\begin{aligned} T(n,k) &\leq \mathcal{O}(m) + f(k)(n^{2d} + g(k) n^{\gamma k + c}) \log n + g'(k) \left(\sum_{w=0}^{d} O(n^w) n^{\gamma(k-w+1)+c'} \right) \\ &\leq g'(k) \log n \cdot \mathcal{O}\left(n^{2r} + n^{\gamma k + c} + \sum_{w=0}^{d} n^{w + \gamma(k-(w+1))+c'} \right) \\ &\leq g'(k) \log n \cdot \mathcal{O}\left(n^{2r} + n^{\gamma k + c} + n^{\gamma k + c'} \right) = g'(k)\mathcal{O}(n^{\gamma k + c'}), \end{aligned}$$

where in the second line, we used that $g'(k) = f(k)g(k)$, and in the last line we used that $\gamma(w+1) \geq w$ as $\gamma \geq d/(d+1) \geq w/(w+1)$ for $w \leq d$, as well as our choice of c' which satisfies $c' > c$ and $\gamma k + c' \geq c' > 2r$.

6 Hardness Results

In this section, we give our hardness results. To this end, we first consider Implications = $\mathsf{SAT}(\text{IMPL})$ and give a $f(k) n^{(\omega/6 - o(1))\sqrt[3]{k}}$-lower bound under the k-clique conjecture. Afterwards, we handle the case of NAND_d- or IMPL-representing families, by reducing from d-uniform (Hyper)Clique or Implications, respectively.

6.1 Hardness for Implications

▶ **Theorem 6.1.** *If* Implications *can be solved in time* $f(k)n^{(\omega/6-\epsilon)\sqrt[3]{k}+c}$ *for some* $\epsilon > 0, c$ *and* $f(k)$*, then the* k*-Clique conjecture fails.*

Proof. Let $G = (V, E)$ be an undirected graph. We construct an Weighted DAG Implications instance $G' = (V', E', w')$ with parameter $k' = k \cdot K + \binom{k}{2}$ with $K \coloneqq \binom{k}{2} + 1$ as follows. The vertex set V' is the disjoint union of *vertex nodes* $V'_V \coloneqq \{v_u \mid u \in V\}$ and *edge nodes* $V'_E \coloneqq \{v_e \mid e \in E\}$. For every $e = \{u, w\} \in E$, we introduce the edges $(v_e, v_u), (v_e, v_w)$ to E'. Furthermore, we set the weights of vertex nodes to K, and the weights of edge nodes to 1.

▷ Claim 6.2. There is a closed set X of weight k' in G' if and only if there is a k-clique in G.

Proof. Let $C = \{v_1, \ldots, v_k\}$ be a k-clique in G. Observe that $X = \{v_u \mid u \in C\} \cup \{v_e \mid e \in \binom{C}{2}\}$ is a closed set in G' of weight $|C|K + \binom{|C|}{2} = k \cdot K + \binom{k}{2} = k'$.

For the converse, assume that X is a closed set in G' of weight k'. Setting $X_V \coloneqq X \cap V'_V$ and $X_E \coloneqq X \cap V'_E$, we show the following sequence of facts:

1) $X_E \subseteq \binom{X_V}{2}$: note that X is only closed if for all $v_{\{u,w\}} \in X_E$, we have $v_u, v_w \in X_V$.
2) $|X_V| = k$ and $|X_E| = \binom{k}{2}$: note that if $|X_V| < k$, then $|X_E| \leq \binom{k-1}{2}$ by 1) and thus the weight of X is $|X_V|K + |X_E| \leq (k-1)K + \binom{k-1}{2} < kK + \binom{k}{2} = k'$. Furthermore, if $|X_V| > k$, then the weight of X is at least $|X_V|K \geq (k+1)K = kK + \binom{k}{2} + 1 > k'$. Thus, we have $|X_V| = k$, and hence we must have $|X_E| = \binom{k}{2}$ for $|X_V|K + |X_E| = k'$ to hold.
3) X_V forms a k-clique in G: Facts 1) and 2) require that $X_E = \binom{X_V}{2}$, which implies that E contains all edges between vertices of X_V.

The last statement concludes the proof of the claim. ◁

Assume that for some c and $\epsilon > 0$, there is an Implications algorithm running in time $f(k)n^{(\omega/6-\epsilon)\sqrt[3]{k}+c}$. Given a k-clique instance G, we run the above reduction to create a Weighted DAG Implications instance G' with parameter $k' \leq (k+1)(\binom{k}{2}+1) = (k^3+k+2)/2 \leq k^3$ for $k \geq 2$. Observe that G' has $\mathcal{O}(n^2)$ nodes and can be converted to an equivalent Implications instance G'' with the same parameter k' and $\mathcal{O}(k^2n^2)$ nodes by simulating each node weight w by a cycle of w nodes. Now, we determine whether G'' has a closed set of weight $k' \leq k^3$ using the Implications algorithm and thus decide k-clique in time $f(k^3)\mathcal{O}((k^2n^2)^{(\omega/6-\epsilon)k+c}) = f(k^3)k^{\mathcal{O}(k)}n^{(\omega/3-2\epsilon)k+2c}$, refuting the k-Clique conjecture. ◀

6.2 Hardness for SAT($\mathcal{F}$)

In this section, we give our hardness results for general constraint families $\mathcal{F}$ by reducing from (d-uniform Hyper-)Clique either via the independent set problem or via Implications.

To obtain these results, we frequently have to plug-in constant 0s or 1s to obtain our desired constraints. Technically, this is a non-trivial step, as we need to enforce some variables to be assigned fixed values without blowing up the number of variables or the weight of the desired solution. To facilitate our proofs, we first formalize the problem variant that allows us to plug-in constants freely.

▶ **Definition 6.3.** *Let* $\mathcal{F}$ *be an arbitrary constraint family and* $\Sigma \subseteq \{0, 1\}$*. The problem* $\mathsf{SAT}_\Sigma(\mathcal{F})$ *asks to determine whether a given formula* ϕ *with Boolean variables* $x_1, \ldots, x_n$ *has a satisfying assignment of weight* k*, where* ϕ *is a conjunction of* m *constraints of the form* $f(\mathbf{x})$*, where* $f : \{0,1\}^r \to \{0,1\}$ *is a constraint function in* $\mathcal{F}$ *and* $\mathbf{x}$ *is an* r*-tuple over* $\{x_1, \ldots, x_n\} \cup \Sigma$ *(any variable or constant* $c \in \Sigma$ *may be used repeatedly). Note that* $\mathsf{SAT}_\emptyset(\mathcal{F}) = \mathsf{SAT}(\mathcal{F})$*.*

Ideally, we would like to show that $\mathsf{SAT}_{\{0,1\}}(\mathcal{F})$ is equivalent to $\mathsf{SAT}(\mathcal{F})$. More specifically, we would like to employ reductions of the following form.

▶ **Definition 6.4.** *Let $\mathcal{F}$ be an arbitrary constraint family, and $\Sigma, \Sigma' \subseteq \{0,1\}$ be disjoint. We say that* $\mathsf{SAT}_\Sigma(\mathcal{F})$ expresses Σ'*, if there is a constant c such that the following holds: For any formula ϕ of $\mathsf{SAT}_{\Sigma\cup\Sigma'}(\mathcal{F})$ and parameter k, we can compute, in linear time, a formula ϕ' of $\mathsf{SAT}_\Sigma(\mathcal{F})$ with parameter $k' := k + c$ such that ϕ has a satisfying assignment of weight k if and only if ϕ' has a satisfying assignment of weight k'.*

Indeed, for 0-invalid $\mathcal{F}$, we can show that $\mathsf{SAT}(\mathcal{F})$ expresses $\{0,1\}$ (this is straightforward and was already shown in [28]). For 0-valid $\mathcal{F}$, however, expressing the constant 1 in general appears impossible. To still give tight hardness results for $\mathcal{F}$ whenever it represents a hard function g, we make use of a stronger notion that captures whether we can obtain g already as a restriction that avoids the constant 1. Formally, let $f : \{0,1\}^r \to \{0,1\}, g : \{0,1\}^s \to \{0,1\}$ be arbitrary Boolean functions. We say that a function f *contains g as a 0-restriction* if g is obtained from f by replacing each argument of f either by an argument of g or the constant 0, i.e., we can partition $[r]$ into $X_1, \ldots, X_s, Z_0$ such that

$$g(x_1, \ldots, x_s) = f(\overbrace{x_1 \ldots x_1}^{X_1}, \ldots, \overbrace{x_s \ldots x_s}^{X_s}, \overbrace{0 \ldots 0}^{Z_0}).$$

Using careful constructions, we can prove the following central technical lemma.

▶ **Lemma 6.5.** *Let $\mathcal{F}$ be an arbitrary constraint family and let g be* IMPL *or* NAND_d *for some $d \geq 2$. If some $f \in \mathcal{F}$ contains g as a restriction, then* $\mathsf{SAT}(\mathcal{F})$ *expresses* $\{0,1\}$*, or* $\mathsf{SAT}(\mathcal{F})$ *expresses* 0 *and f contains g already as a 0-restriction.*

Postponing the proof of the above lemma to the Sections 6.3 and 6.4, we can give the proof of our hardness results.

▶ **Theorem 6.6** (Hardness for $\mathsf{SAT}(\mathcal{F})$)**.** *Let $\mathcal{F}$ be a constraint family.*

1. *If $\mathcal{F}$ represents* IMPL*, then* $\mathsf{SAT}(\mathcal{F})$ *cannot be solved in time $f(k)n^{(\omega/6-\epsilon)\sqrt[3]{k}+c}$ for any computable $f(k)$ and constants $c, \epsilon > 0$, unless the k-Clique conjecture fails.*
2. *If $\mathcal{F}$ represents* NAND_2*, then* $\mathsf{SAT}(\mathcal{F})$ *cannot be solved in time $f(k)n^{(\omega/3-\epsilon)k+c}$ for any computable $f(k)$ and constants $c, \epsilon > 0$, unless the k-Clique conjecture fails.*
3. *If $\mathcal{F}$ represents* NAND_d *with $d \geq 3$, then* $\mathsf{SAT}(\mathcal{F})$ *cannot be solved in time $f(k)n^{(1-\epsilon)k+c}$ for any computable $f(k)$ and constants $c, \epsilon > 0$, unless the d-uniform HyperClique conjecture fails.*

Proof. First, we observe that Implications reduces to SAT(IMPL) such that

$$T_{\text{Implications}}(n,k) \leq O(T_{\mathsf{SAT}(\text{IMPL})}(n,k)). \tag{7}$$

Indeed, given any directed graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$, we define the formula ϕ with variables $x_1, \ldots, x_n$ and the set of constraints obtained by including $x_i \Rightarrow x_j$ for all $(v_i, v_j) \in E$. Note that for any $S \subseteq [n]$, $\{v_i\}_{i\in S}$ is a valid set in G iff a_S is a satisfying assignment of ϕ, yielding (7).

Similarly, we observe that the d-uniform HyperClique problem reduces to $\mathsf{SAT}(\mathrm{NAND}_d)$ such that

$$T_{d\text{-HC}}(n,k) \leq O(T_{\mathsf{SAT}(\mathrm{NAND}_d)}(n,k)). \tag{8}$$

Indeed, given any d-uniform hypergraph $G = (V, E)$ with $|V| = n$, we define the formula ϕ with variables $x_1, \ldots, x_n$ and the constraints obtained by including, for each distinct $v_{i_1}, \ldots, v_{i_d} \in V$ such that $(v_{i_1}, \ldots, v_{i_d}) \notin E$, the constraint $\mathrm{NAND}_d(x_{i_1}, \ldots, x_{i_d})$. Observe that $(v_{i_1}, \ldots, v_{i_k}) \in V^k$ is a hyperclique in G iff the weight-k assignment with $x_{i_\ell} = 1$ for all $\ell \in [k]$ satisfies ϕ, yielding (8).

It remains to show that whenever some $f \in \mathcal{F}$ contains $g \in \{\mathrm{IMPL}\} \cup \{\mathrm{NAND}_d \mid d \geq 2\}$ as a restriction, then there is a computable $f'(k)$ and constant c' such that

$$T_{\mathsf{SAT}(g)}(n, k) \leq f'(k) \cdot T_{SAT(\mathcal{F})}(n, k + c'). \tag{9}$$

Indeed, if $\mathsf{SAT}(\mathcal{F})$ expresses $\{0, 1\}$, then

$$T_{\mathsf{SAT}(g)}(n, k) \leq \mathcal{O}(T_{\mathsf{SAT}_{\{0,1\}}(\mathcal{F})}(n, k)) \leq f'(k)\mathcal{O}(T_{\mathsf{SAT}(\mathcal{F})}(n, k + c')).$$

Here the first inequality follows by replacing each occurrence of a constraint $g(x_{i_1}, \ldots, x_{i_d})$ of $\mathsf{SAT}(g)$ by the corresponding restriction $f(g_1(x_{i_1}, \ldots, x_{i_d}), \ldots, g_r(x_{i_1}, \ldots, x_{i_d}))$ of $\mathsf{SAT}_{\{0,1\}}(\mathcal{F})$. The second inequality follows from the definition of $\mathsf{SAT}(\mathcal{F})$ expressing $\{0, 1\}$.

In the other case, $\mathsf{SAT}(\mathcal{F})$ expresses only 0, but f contains g already as a 0-restriction. Then we have

$$T_{\mathsf{SAT}(g)}(n, k) \leq \mathcal{O}(T_{\mathsf{SAT}_{\{0\}}(\mathcal{F})}(n, k)) \leq f'(k)\mathcal{O}(T_{\mathsf{SAT}(\mathcal{F})}(n, k + c')),$$

as replacing each occurrence of a constraint $g(x_{i_1}, \ldots, x_{i_d})$ of $\mathsf{SAT}(g)$ by the corresponding restriction $f(g_1(x_{i_1}, \ldots, x_{i_d}), \ldots, g_r(x_{i_1}, \ldots, x_{i_d}))$ does not require the use of the constant 1. The second inequality again follows from the definition of $\mathsf{SAT}(\mathcal{F})$ expressing 0.

As a consequence, by (7) and (9), a $f(k) \cdot O(n^{(\omega/6-\epsilon)\sqrt[3]{k}+c})$ $\mathsf{SAT}(\mathcal{F})$ algorithm for an IMPL-representing family $\mathcal{F}$ would then give an Implications algorithm running in time

$$f(k)f'(k)\mathcal{O}(n^{(\omega/6-\epsilon)\sqrt[3]{k+c'}+c}) = f''(k)\mathcal{O}(n^{(\omega/6-\epsilon)\sqrt[3]{k}+c''}),$$

where $f''(k) = f(k)f'(k)$ and $c'' \leq c + \sqrt[3]{c'}$. This would refute the k-Clique conjecture by Theorem 6.1, concluding 1.

Similarly, a $f(k) \cdot O(n^{\gamma k+c})$ $\mathsf{SAT}(\mathcal{F})$ algorithm for an NAND_d-representing family $\mathcal{F}$ would give a d-uniform HyperClique algorithm running in time

$$f(k)f'(k)\mathcal{O}(n^{\gamma k+c+c'}) = f''(k)\mathcal{O}(n^{\gamma k+c''}),$$

where $f''(k) = f(k)f'(k)$ and $c'' = c + c'$. This yields 2. and 3. by the k-Clique or d-uniform HyperClique conjecture, respectively. ◀

In the remainder of the section, we prove Lemma 6.5. We split the proof in two cases, depending on whether f is 0-invalid (Lemma 6.7) or 0-valid (Corollary 6.14).

6.3 Proof of Lemma 6.5: 0-invalid case

Let f be such that we can obtain IMPL or NAND_d for $d \geq 2$ as a restriction. Note that if it contains $\mathrm{NAND}_d, d > 2$ then it also must contain NAND_2 as a restriction.

In this section, we consider the case that $f(y_1, \ldots, y_r)$ is not 0-valid, i.e., the all-zeroes assignment $u_1 = \cdots = u_r = 0$ does not satisfy f.

▶ **Lemma 6.7.** *If f contains* IMPL *or* NAND_2 *as a restriction and f is 0-invalid, then* $\mathsf{SAT}(\mathcal{F})$ *expresses* $\{0, 1\}$.

The above result in fact follows from the following claim.

▷ Claim 6.8. Let f be as above. Given a parameter k', we can compute, in time $\mathcal{O}(k')$, a formula $\phi_{0,1}$ of $\mathsf{SAT}(\mathcal{F})$ with variables $y, z_1, \ldots, z_{k'+1}$ such that the only satisfying assignment of weight at most k' is $y = 1, z_1 = \cdots = z_{k'+1} = 0$.

Indeed, let us assume the above claim, and take any formula ϕ of $\mathsf{SAT}_{\{0,1\}}(\mathcal{F})$ with parameter k. We construct $\phi_{0,1}$ with parameter $k' := k + 1$ and define the formula ϕ' on variable set $x_1, \ldots, x_n, y, z_1, \ldots, z_{k'+1}$ where we include all constraints of $\phi_{0,1}$ and all constraints of ϕ, *replacing each use of the constant* 0 *by* z_1 *and each use of the constant* 1 *by* y. This yields a formula of $\mathsf{SAT}(\mathcal{F})$ with the property that for any weight-k solution $x_1, \ldots, x_n$ of ϕ, the corresponding assignment that sets $y = 1$ and $z_1 = \cdots = z_{k'+1} = 0$ is a weight-$(k+1)$ solution of ϕ'. Conversely, any $(k+1)$-weight solution of ϕ' must set $y = 1$ and $z_1 = 0$ by the above claim, and hence the assignment to $x_1, \ldots, x_n$ must also satisfy ϕ. Observe that this proves Lemma 6.7.

Proof of Claim 6.8. We first give a set of constraints that enforces $y = 1$. To this end, let $S \subseteq [r]$ be such that a_S satisfies f; observe that S exists and is non-empty (otherwise f contains neither IMPL nor NAND_2 as a restriction). For each $j = 1, \ldots, k' + 1$, define the constraint C_j obtained by plugging in y for each u_i with $i \in S$ (i.e., all arguments set to 1 under a_S), and z_j for all other values. We claim that any weight-$(\leq k')$ assignment satisfying $\bigwedge_{j=1}^{k'+1} C_j$ sets $y = 1$: by the weight restriction, at least one of $z_1, \ldots, z_{k'+1}$ must be equal to 0, say z_{j^*}. Then setting $y = 0$ would falsify C_{j^*}, as then all its arguments are 0. Note, however, that the desired assignment $y = 1, z_1 = \cdots = z_{k'+1} = 0$ satisfies $\bigwedge_{j=1}^{k'+1} C_j$.

It remains to give additional constraints enforcing that $z_j = 0$ for *all* $j \in [k' + 1]$. As a first step, we find $S \subsetneq T$ such that $f(a_S) = 1$ but $f(a_T) = 0$: Since f represents IMPL or NAND_2, there is a partition of $[r]$ into X, Y, Z_0, Z_1 such that one of the following set of equalities hold:

	X	Y	Z_0	Z_1				X	Y	Z_0	Z_1		
$f($	$\overbrace{0\ldots0}$,	$\overbrace{0\ldots0}$,	$\overbrace{0\ldots0}$,	$\overbrace{1\ldots1}$)	$=$	1,	$f($	$\overbrace{0\ldots0}$,	$\overbrace{0\ldots0}$,	$\overbrace{0\ldots0}$,	$\overbrace{1\ldots1}$)	$=$	1,
$f($	$0\ldots0$,	$1\ldots1$,	$0\ldots0$,	$1\ldots1$)	$=$	1,	$f($	$1\ldots1$,	$0\ldots0$,	$0\ldots0$,	$1\ldots1$)	$=$	1,
$f($	$1\ldots1$,	$1\ldots1$,	$0\ldots0$,	$1\ldots1$)	$=$	1,	$f($	$0\ldots0$,	$1\ldots1$,	$0\ldots0$,	$1\ldots1$)	$=$	1,
$f($	$1\ldots1$,	$0\ldots0$,	$0\ldots0$,	$1\ldots1$)	$=$	0.	$f($	$1\ldots1$,	$1\ldots1$,	$0\ldots0$,	$1\ldots1$)	$=$	0.

In both cases, the first and fourth line yield sets $S \subsetneq T$ with $f(a_S) = 1$ and $f(a_T) = 0$ (specifically, for $S = Z_1$ and $T = X \cup Z_1$ or for $S = Z_1$ and $T = X \cup Y \cup Z_1$).

Given such S, T, for each $j, j' \in \binom{[r]}{2}$, we define the constraint $C'_{j,j'}$ obtained from $f(u_1, \ldots, u_r)$ by plugging-in y for all u_i with $i \in S$, z_j for all $i \in T \setminus S$ and $z_{j'}$ for all other i. Note that any satisfying assignment of weight at most k sets at least one of $z_1, \ldots, z_{k'+1}$ to 0, say z_{j^*}. Observe that the constraint C'_{j,j^*} is satisfied iff $z_j = 0$, as setting z_j to 0 or 1 corresponds to the assignments a_S (satisfying) or a_T (unsatisfying), respectively. Furthermore, observe that setting $y = 1$ and $z_1 = \cdots = z_{k'+1} = 0$ indeed satisfies all $C'_{j,j'}$. This concludes the claim that the only satisfying assignment of weight at most k' is $y = 1, z_1 = \cdots = z_{k'+1} = 0$. ◁

6.4 Proof of Lemma 6.5: 0-valid case

In this section, we consider the case that $f(y_1, \ldots, y_r)$ is 0-valid, i.e., the all-zeroes assignment $u_1 = \cdots = u_r = 0$ satisfies f. We first observe that we can still express at least the constant 0.

▶ **Lemma 6.9.** *If some* $f \in \mathcal{F}$ *contains* IMPL *or* NAND_2 *as a restriction and* f *is 0-valid, then* $\mathsf{SAT}(\mathcal{F})$ *expresses* 0.

Proof. Observe that it suffices to show how to construct, given a parameter k, a formula on variables $z_1, \dots, z_{k+1}$ such that the only satisfying assignment of weight at most k sets $z_1 = \cdots = z_{k+1} = 0$.

To this end, assume first that f is not satisfied by the all-ones assignment. Then, the formula $\bigwedge_{i=1}^{k+1} f(z_i, \dots, z_i)$ is trivially only satisfied by the assignment $z_1 = \cdots = z_{k+1} = 0$.

Otherwise, observe that there must be a non-empty set $S \subsetneq [r]$ such that a_S does not satisfy f (otherwise f would be a trivial constraint and could contain neither of IMPL and NAND_2). For each $i, i' \in [k+1]$, we define the constraint $C_{i,i'}$ obtained by using z_i for all arguments in S, and $z_{i'}$ for all arguments not in S. Observe that $C_{i,i'} \wedge C_{i',i}$ forces $z_i = z_{i'}$, and thus $z_1 = \cdots = z_{k+1}$, which is satisfied by an assignment of weight at most k if and only if the common value is 0. ◀

Interestingly, for 0-valid f, containing IMPL as a restriction is equivalent to containing IMPL already as a 0-restriction.

▶ **Lemma 6.10.** *If f contains* IMPL *as a restriction and is* 0*-valid, then f contains* IMPL *already as a* 0*-restriction.*

Proof. Since $f : \{0,1\}^r \to \{0,1\}$ contains IMPL as a restriction, we can partition $[r]$ into sets X, Y, Z_0, Z_1 and write

$$\begin{array}{llllll} f(\overbrace{0\dots0}^{X}, & \overbrace{0\dots0}^{Y}, & \overbrace{0\dots0}^{Z_0}, & \overbrace{1\dots1}^{Z_1}) & = & 1, \\ f(0\dots0, & 1\dots1, & 0\dots0, & 1\dots1) & = & 1, \\ f(1\dots1, & 1\dots1, & 0\dots0, & 1\dots1) & = & 1, \\ \hline f(1\dots1, & 0\dots0, & 0\dots0, & 1\dots1) & = & 0. \end{array} \tag{10}$$

Assume first that

$$f(\overbrace{0\dots0}^{X}, \overbrace{1\dots1}^{Y}, \overbrace{0\dots0}^{Z_0}, \overbrace{0\dots0}^{Z_1}) = 0. \tag{11}$$

Then, we obtain IMPL as a 0-restriction by setting $X' \coloneqq Y, Y' \coloneqq Z_1, Z' \coloneqq X \cup Z_0$ and observing that

$$\begin{array}{llllll} f(\overbrace{0\dots0}^{X'=Y}, & \overbrace{0\dots0}^{Y'=Z_1}, & \overbrace{0\dots0}^{Z'=X\cup Z_0}) & = & 1, & [f \text{ is 0-valid}] \\ f(0\dots0, & 1\dots1, & 0\dots0) & = & 1, & [\text{by (10)}] \\ f(1\dots1, & 1\dots1, & 0\dots0) & = & 1, & [\text{by (10)}] \\ \hline f(1\dots1, & 0\dots0, & 0\dots0) & = & 0. & [\text{by (11)}] \end{array}$$

Otherwise, if (11) does not hold, then we obtain IMPL as a 0-restriction by setting $X' \coloneqq X \cup Z_1, Y' \coloneqq Y, Z' \coloneqq Z_0$ and observing that

$$\begin{array}{llllll} f(\overbrace{0\dots0}^{X'=X\cup Z_1}, & \overbrace{0\dots0}^{Y'=Y}, & \overbrace{0\dots0}^{Z'=Z_0}) & = & 1, & [f \text{ is 0-valid}] \\ f(0\dots0, & 1\dots1, & 0\dots0) & = & 1, & [\text{by } \neg(11)] \\ f(1\dots1, & 1\dots1, & 0\dots0) & = & 1, & [\text{by (10)}] \\ \hline f(1\dots1, & 0\dots0, & 0\dots0) & = & 0. & [\text{by (10)}] \end{array}$$

◀

It remains to handle the case that f contains NAND_d as a restriction. We first observe that if f contains IMPL as a 0-restriction, then $\mathsf{SAT}_0(\mathcal{F})$ even expresses the constant 1. (Thus, afterwards, we may assume that f does not contain IMPL as a 0-restriction.)

▶ **Lemma 6.11.** *If some $f \in \mathcal{F}$ contains* IMPL *as a 0-restriction, then* $\mathsf{SAT}_0(\mathcal{F})$ *expresses* 1.

Proof. Given any formula ϕ of $\mathsf{SAT}_{\{0,1\}}(\mathcal{F})$ on variables $x_1, \dots, x_n$, we construct a formula ϕ' on variables $x_1, \dots, x_n, y$ as follows: Since some $f \in \mathcal{F}$ contains IMPL as a 0-restriction, we can express, for any variables v, v', the implication $v \Rightarrow v'$ by a corresponding constraint of $\mathsf{SAT}_0(\mathcal{F})$. We construct n such constraints to enforce $\bigwedge_{j=1}^{n}(x_j \Rightarrow y)$. Subsequently, we may use y to replace any use of the constant 1 to convert the constraints of ϕ to constraints of the $\mathsf{SAT}_0(\mathcal{F})$-formula ϕ'.

To argue correctness, note that any satisfying weight-k assignment of ϕ yields a satisfying weight-$(k+1)$ assignment of ϕ' by setting $y = 1$. Conversely, note that any weight-$(k+1)$-assignment of ϕ' must set $y = 1$ (since $k \geq 1$ implies that at least one variable x_i is set to one, which enforces $y = 1$ by the corresponding implication $x_i \Rightarrow y$) and thus corresponds to a weight-k assignment to $x_1, \dots, x_n$ satisfying ϕ. ◀

In the remainder of this section, we assume that f contains NAND_d as a restriction, but does not contain IMPL as a 0-restriction, and the aim is to find NAND_d already as a 0-restriction.

▶ **Lemma 6.12.** *For any 0-valid f, if f does not contain* IMPL *as a 0-restriction, then whenever $f(a_S) = f(a_T) = 1$ with $S \subseteq T$, then $f(a_{T \setminus S}) = 1$.*

Proof. If $S = T$, there is nothing to show, so let $S \subsetneq T$ and assume for contradiction that $f(a_{T \setminus S}) = 0$. We obtain IMPL as a 0-restriction as follows:

$f(\overbrace{0\dots0}^{X=T\setminus S},$	$\overbrace{0\dots0}^{Y=S},$	$\overbrace{0\dots0}^{Z=[r]\setminus T})$	$=$ $1,$	$[f$ is 0-valid$]$
$f(0\dots0,$	$1\dots1,$	$0\dots0)$	$=$ $1,$	$[f(a_S) = 1]$
$f(1\dots1,$	$1\dots1,$	$0\dots0)$	$=$ $1,$	$[f(a_T) = 1]$
$f(1\dots1,$	$0\dots0,$	$0\dots0)$	$=$ $0.$	[by assumption]

This yields the claim. ◀

We can finally obtain NAND_d as a 0-restriction.

▶ **Lemma 6.13.** *If f contains NAND_d as a restriction, does not contain* IMPL *as a 0-restriction and is 0-valid, then f contains NAND_d already as a 0-restriction.*

Proof. Since $f : \{0,1\}^r \to \{0,1\}$ contains NAND_d as a restriction, we can partition $[r]$ into sets $X_1, \dots, X_d, Z_0, Z_1$ such that $X_I \coloneqq \bigcup_{i \in I} X_i$ with $I \subseteq [d]$ satisfies:

$$f(a_{X_I \cup Z_1}) = \begin{cases} 0 & \text{if } I = [d], \\ 1 & \text{if } I \subsetneq [d]. \end{cases} \tag{12}$$

We claim that the partition $X'_i \coloneqq X_i$ for $i < d$, $X'_d \coloneqq X_d \cup Z_1$, $Z' \coloneqq Z_0$ provides NAND_d as a 0-restriction: Letting $X'_I \coloneqq \bigcup_{i \in I} X'_i$, this follows from

$$f(a_{X'_I}) = \begin{cases} 0 & \text{if } I = [d], \\ 1 & \text{if } I \subsetneq [d]. \end{cases} \tag{13}$$

To verify (13), note first that $f(a_{X'_{[d]}}) = f(a_{X_{[d]} \cup Z_1}) = 0$ by (12). Second, let $I \subsetneq [d]$. If $d \in I$, then $f(a_{X'_I}) = f(a_{X_I \cup Z_1}) = 1$ by (12). Otherwise, if $d \notin I$, then we have $f(a_{X'_I}) = f(a_{X_I}) = 1$ by Lemma 6.12 (for this, note that f does not contain IMPL as 0-restriction and that $f(a_{X_I \cup Z_1}) = f(a_{Z_1}) = 1$). This concludes the claim. ◀

The proof of this section is summarized in the following corollary.

▶ **Corollary 6.14.** *If f contains $g \in \{\text{IMPL}\} \cup \bigcup_{d\geq 2}\{\text{NAND}_d\}$ and f is 0-valid, then $\mathsf{SAT}(f)$ expresses $\{0,1\}$, or $\mathsf{SAT}(f)$ expresses 0 and contains g as a 0-restriction.*

Proof. If $g = \text{IMPL}$, then f contains g already as a 0-restriction by Lemma 6.10 and $\mathsf{SAT}(f)$ expresses $\{0,1\}$ by Lemmas 6.9 and 6.11.

If $g = \text{NAND}_d$, then either f also contains IMPL as a 0-restriction, in which case $\mathsf{SAT}(f)$ expresses $\{0,1\}$ by Lemmas 6.9 and 6.11, or it does not contain IMPL as a 0-restriction, and thus f contains g as a 0-restriction by Lemma 6.13 and $\mathsf{SAT}(f)$ expresses 0 by Lemma 6.9. ◀

References

1 Amir Abboud, Arturs Backurs, Karl Bringmann, and Marvin Künnemann. Fine-grained complexity of analyzing compressed data: Quantifying improvements over Decompress-and-Solve. In *Proc. 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2017)*, pages 192–203, 2017. doi:10.1109/FOCS.2017.26.

2 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is valiant's parser. *SIAM J. Comput.*, 47(6):2527–2555, 2018. doi:10.1137/16M1061771.

3 Amir Abboud, Karl Bringmann, Holger Dell, and Jesper Nederlof. More consequences of falsifying SETH and the Orthogonal Vectors conjecture. In *Proc. 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2018)*, pages 253–266, New York, NY, USA, 2018. ACM. doi:10.1145/3188745.3188938.

4 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.

5 Enric Boix-Adserà, Matthew Brennan, and Guy Bresler. The average-case complexity of counting cliques in Erdős-Rényi hypergraphs. In *Proc. 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2019)*, pages 1256–1280, 2019. doi:10.1109/FOCS.2019.00078.

6 Édouard Bonnet, László Egri, and Dániel Marx. Fixed-Parameter Approximability of Boolean MinCSPs. In Piotr Sankowski and Christos Zaroliagis, editors, *Proc. 24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:18, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ESA.2016.18.

7 Alfred Brauer. On a problem of partitions. *American Journal of Mathematics*, 64(1):299–312, 1942.

8 Karl Bringmann. A near-linear pseudopolynomial time algorithm for subset sum. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 1073–1084, 2017. doi:10.1137/1.9781611974782.69.

9 Karl Bringmann, Nick Fischer, and Marvin Künnemann. A fine-grained analogue of Schaefer's theorem in P: dichotomy of $\exists^k\forall$-quantified first-order graph properties. In *Proc. 34th Computational Complexity Conference (CCC 2019)*, pages 31:1–31:27, 2019. doi:10.4230/LIPIcs.CCC.2019.31.

10 Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. A dichotomy for regular expression membership testing. In Chris Umans, editor, *Proc. 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2017)*, pages 307–318. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.36.

11 Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *Proc. 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2017)*, pages 319–330, 2017. doi:10.1109/FOCS.2017.37.

12 Andrei A. Bulatov and Dániel Marx. Constraint satisfaction parameterized by solution size. *SIAM J. Comput.*, 43(2):573–616, 2014. doi:10.1137/120882160.

13 Nofar Carmeli and Markus Kröll. On the enumeration complexity of unions of conjunctive queries. In Dan Suciu, Sebastian Skritek, and Christoph Koch, editors, *Proc. 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS 2019)*, pages 134–148. ACM, 2019. doi:10.1145/3294052.3319700.

14 Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David W. Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized NP-hard problems. *Inf. Comput.*, 201(2):216–231, 2005. doi:10.1016/j.ic.2005.05.001.

15 Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006. doi:10.1016/j.jcss.2006.04.007.

16 Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010. doi:10.1016/j.tcs.2010.06.026.

17 Nadia Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *J. Comput. Syst. Sci.*, 51(3):511–522, 1995. doi:10.1006/jcss.1995.1087.

18 Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of boolean constraint satisfaction problems*. SIAM, 2001. doi:10.1137/1.9780898718546.

19 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theor. Comput. Sci.*, 326(1-3):57–67, 2004. doi:10.1016/j.tcs.2004.05.009.

20 Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998. doi:10.1137/S0097539794266766.

21 Jean Gallier. The Frobenius coin problem. Upper bounds on the Frobenius number, 2014.

22 Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000. doi:10.1137/S0097539799349948.

23 Konstantinos Koiliaris and Chao Xu. Faster pseudopolynomial time algorithms for subset sum. *ACM Trans. Algorithms*, 15(3):40:1–40:20, 2019. doi:10.1145/3329863.

24 Stefan Kratsch, Dániel Marx, and Magnus Wahlström. Parameterized complexity and kernelizability of max ones and exact ones problems. *TOCT*, 8(1):1:1–1:28, 2016. doi:10.1145/2858787.

25 Andrei A. Krokhin and Dániel Marx. On the hardness of losing weight. *ACM Trans. Algorithms*, 8(2):19:1–19:18, 2012. doi:10.1145/2151171.2151182.

26 Bingkai Lin. The parameterized complexity of the k-biclique problem. *J. ACM*, 65(5):34:1–34:23, 2018. doi:10.1145/3212622.

27 Andrea Lincoln, Virginia Vassilevska Williams, and Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In *Proc. 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*, pages 1236–1252, Philadelphia, PA, USA, 2018. Society for Industrial and Applied Mathematics.

28 Dániel Marx. Parameterized complexity of constraint satisfaction problems. *Computational Complexity*, 14(2):153–183, 2005. doi:10.1007/s00037-005-0195-9.

29 Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 026(2):415–419, 1985.

30 Jorge L. Ramirez Alfonsin. *The diophantine Frobenius problem*. Oxford University Press, Oxford, 2005.

31 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th Annual ACM Symposium on Theory of Computing (STOC 1978)*, pages 216–226, 1978. doi:10.1145/800133.804350.

32 Jeanette P. Schmidt and Alan Siegel. The spatial complexity of oblivious k-probe hash functions. *SIAM J. Comput.*, 19(5):775–786, 1990. doi:10.1137/0219054.

33 Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972. doi:10.1137/0201010.

34 Dmitriy Zhuk. A proof of CSP dichotomy conjecture. In *Proc. 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2017)*, pages 331–342, 2017. doi:10.1109/FOCS.2017.38.

35 Dmitriy Zhuk and Barnaby Martin. QCSP monsters and the demise of the Chen conjecture. *CoRR*, abs/1907.00239, 2019. arXiv:1907.00239.