When Does Sparsity Help for k-Independent Set in Hypergraphs and Other Boolean CSPs?

Timo Fritsch
Karlsruhe Institute of Technology, Germany

Marvin Künnemann
Karlsruhe Institute of Technology, Germany

Mirza Redzic
Karlsruhe Institute of Technology, Germany

Julian Stieß
Karlsruhe Institute of Technology, Germany

Abstract

Consider the fundamental task of finding independent sets of (constant) size k in a given n-node *hyper*graph. How much is the time complexity affected by the sparsity of the input, i.e., the number of hyperedges m? Turán's theorem implies that the problem is trivial if $m = O(n^{2-\epsilon})$ for some $\epsilon > 0$. Above that threshold (i.e., if $m = \Theta(n^\gamma)$ for some $\gamma \geq 2$), we give a perhaps surprising algorithm with running time $O\left(\min\left\{n^{\frac{\omega}{3}k} + m^{k/3}, n^k\right\}\right)$ (for k divisible by 3), which is essentially *conditionally optimal* for all $\gamma \geq 2$, assuming the k-clique and 3-uniform hyperclique hypotheses (here, $\omega \leq 2.372$ denotes the matrix multiplication exponent). In fact, we obtain a more detailed time complexity that is sensitive to the arity distribution of the hyperedges.

To study such phenomena in more generality, we study the time complexity of finding solutions of (constant) size k in sparse instances of Boolean constraint satisfaction problems, where n and m denote the number of variables and constraints, respectively. Our results include, among others:

- an essentially full classification of the influence of sparsity for Boolean constraint families of binary arity. Of particular technical interest is a conditionally tight algorithm for the family consisting of the binary NAND and the binary Implication constraints, with a running time of $\Theta(m^{\omega k/6 \pm c})$.
- the identification of a large class of constraint families $\mathcal{F}$ that exhibits a sharp phase transition: there is a threshold $\gamma_{\mathcal{F}}$ such that the problem is trivial for $m = O(n^{\gamma_{\mathcal{F}}-\epsilon})$, but requires essentially brute-force running time $\Theta(n^{k\pm c})$ for $m = \Omega(n^{\gamma_{\mathcal{F}}})$, assuming the 3-uniform hyperclique hypothesis.

In general, we observe a rich landscape of time complexities. Notably, in many cases the *combination* of constraints display higher time complexity than either constraint alone.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Problems, reductions and completeness

Keywords and phrases Multivariate algorithmics, fine-grained complexity theory, classification theorems, algorithmic hypergraph theory

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.94

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2605.10778`

Funding *Marvin Künnemann*: Research partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 462679611.
Mirza Redzic: Research partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 462679611.
Julian Stieß: Research partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 462679611.

53rd International Colloquium on Automata, Languages, and Programming (ICALP 2026).
Editors: Sayan Bhattacharya, Danupon Nanongkai, Michael Benedikt, and Gabriele Puppis;
Article No. 94; pp. 94:1–94:24

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

How does *sparsity* of the input influence a problem's time complexity? From an algorithmic perspective, such questions have been well studied particularly for graph-theoretic problems, as many real-world graphs are rather far from dense. Indeed, for a problem solvable in time $n^{c\pm o(1)}$ on n-node graphs, a natural target to shoot for is an $m^{c/2\pm o(1)}$-time algorithm, where m denotes the number of edges in the graph. Such an algorithm recovers the time bound of $n^{c\pm o(1)}$ in the dense case $m = \Theta(n^2)$, while significantly improving the running time if $m = O(n^{2-\epsilon})$ – we shall refer to such a situation as a *natural interpolation*. For many graph problems, natural interpolation is indeed achievable. Much less is known on the hardness side. Nevertheless, a growing body of work establishes conditional lower bounds ruling out natural interpolation for several fundamental problems. Notable examples include: approximate Diameter and Radius [26, 4], All-Edges Triangle Detection [25, 30], APSP and related problems [21], k-Dominating Set for $k \geq 3$ [15] and more; see also [5].

The goal of this work is to investigate the rich interplay of the input sparsity and the resulting time complexity. Particularly, we focus on cases in which instead of achieving a natural interpolation, the running time exhibits a complex relationship with the sparsity – possibly overshooting or undershooting natural interpolation, or both. Predominantly, we investigate problems beyond graphs, specifically, hypergraph problems as well as constraint satisfaction problems. In such problems, n objects (nodes or variables) interact in a combination of relationships (hyperedges of different arity or constraints of different types). If each relation or constraint has arity at most r, then the sparsity m (i.e., the total size of these relationships) is bounded by $O(n^r)$. In these cases, for an $n^{c\pm o(1)}$-time solvable problem, we denote the natural interpolation as a running time of the form $m^{c/r\pm o(1)}$. We shall see several cases in which this running time is *partially* achievable, using *conditionally optimal* algorithms that carefully consider the combination of relationships.

1.1 Result I: Conditionally Optimal k-Independent Set in Sparse Hypergraphs

As our first focus, we consider the k-Independent Set problem (k-IS) in hypergraphs: Given a (hyper)graph $G = (V, E)$, determine if there is a set $S \subseteq V$ of size k that contains no (hyper)edge of G, i.e., $e \not\subseteq S$ for all $e \in E$. Already in graphs, k-IS is central to algorithmic graph theory: by complementing the graph, we obtain the classic k-clique problem, perhaps the best known $W[1]$-complete problem.[1] Generalizing to hypergraphs, we obtain a significantly more expressive problem. Indeed, for 3-uniform hypergraphs, it is equivalent (up to sparsity) to the 3-uniform hyperclique problem. This problem is generally considered to be more difficult than k-clique; the corresponding 3-uniform hyperclique hypothesis has seen a surge of applications recently, see [21, 2, 7, 20, 8, 13, 23, 16, 17], among others (see below for further discussion). As a further case in point, note that the fairly recent hypergraph container method (see, e.g., [6]) gives a combinatorial tool for independent sets in well-behaved hypergraphs, which has found various applications, including in algorithm design [32].

We ask: *Let $k \geq 2$ and $\gamma \geq 0$ be constants. What is the time complexity of k-Independent Set in n-node hypergraphs with $m = \Theta(n^\gamma)$ hyperedges?*

[1] Note however, that complementing a sparse graph generally yields a dense graph, so that the influence of sparsity differs between k-clique and k-IS.

Let us first consider the case of graphs rather than hypergraphs. Here, $0 \leq \gamma \leq 2$, and without taking sparsity into account, k-IS is well known to be solvable in time $O(n^{(\omega/3)k})$ [24][2]; the k-clique hypothesis postulates that this running time is essentially optimal. However, taking sparsity into account, the problem becomes *trivial* when $\gamma < 2$, i.e., $m = O(n^{2-\epsilon})$: This already follows from Turán's Theorem, which implies that any n-node graph with at most $(\frac{1}{k-1} - o(1))\frac{n^2}{2}$ edges contains an independent set of size k.

The case for h-uniform hypergraphs with $r \geq 3$ can be resolved analogously: Without taking sparsity into account, no substantial improvement over brute-force running time $O(n^k)$ is known (see [3] for subpolynomial-factor improvements), and the 3-uniform hyperclique hypothesis postulates that this is best possible. While generalizing Turán's theorem to hypergraphs is challenging (see, e.g., [18]), a simple greedy argument establishes that k-IS in h-uniform hypergraphs with $\gamma < h$ (i.e., the hypergraph contains $m \leq O(n^{h-\epsilon})$ hyperedges) is again trivial. Thus, for any arity $2 \leq h \leq k-1$, the k-IS problem in h-uniform hypergraphs is non-trivial only for the hardest, dense case of $\gamma = h$.[3]

We thus turn to the general case of hypergraphs with possibly *mixed-arity* hyperedges. We may assume without loss of generality that $\gamma \leq k$, since we may simply ignore any edge of arity at least $k+1$. The problem is trivial only for $\gamma < 2$. How does the problem's complexity behave in between? E.g., how quickly can we solve k-IS when we have, say, $\Theta(n^2)$ edges (of binary arity) and $\Theta(n^{2.5})$ hyperedges of arity 3?

In this case, we determine a conditionally optimal running time of $O(n^{(5/6)k\pm c})$, which is intermediate between $O(n^{(\omega/3)k\pm c})$ and $O(n^{k\pm c})$. More generally, we obtain the following result.

▶ **Theorem 1.** *Assuming the Clique and the 3-uniform Hyperclique Hypothesis, the optimal running time for k-Independent Set in n-node hypergraphs with m hyperedges, each of arity at least 2, is*

$$\min\{n^{\frac{\omega}{3}k} + m^{k/3}, n^k\},$$

up to a factor of the form $O(n^c)$ for some c independent of k.

For an illustration, we refer to Figure 2(a) – note that for $\omega \leq \gamma \leq 3$, and only there, natural interpolation is obtained. In fact, we give a much more detailed statement that takes the arity distribution into account, see Section 1.3.

Technically, our result relies on a very careful combination of ideas, which overcomes the challenge of incorporating different ways to handle edges of different arity. The main technical obstacle here is that the well known k-clique algorithm in graphs due to Nešetřil and Poljak [24] is not well compatible with including *any* hyperedge of arity at least 3. However, exploiting that it can be used to *count* all k-cliques as well, we seek to count all k-cliques on the (binary-arity) edges, and subtract from this count the number of solutions including any of the $\leq m$ hyperedges of larger arity. To solve this task, we present a careful argument based on inclusion-exclusion and sparse witness listing in Section 1.3.[4]

[2] Whenever k is divisible by 3; in other cases, the running time is only slightly higher.

[3] The case of $h = k$ is trivial for all $\gamma < k$ and trivially solvable in linear time in the input if $\gamma = k$.

[4] We remark that the idea of using inclusion-exclusion in combination with fast clique/triangle counting has been used in prior work, e.g., to obtain a $O(m^{2\omega/(\omega+1)})$-time algorithm for counting 3-ISes [29, Footnote 2]. In our work, we exploit these ingredients further to overcome a different challenge, i.e., handling the presence of mixed-arity edges.

We now turn towards a vastly more general setting, specifically, the class of Boolean constraint satisfaction problems, which includes the k-IS problem in hypergraphs as just one of many interesting examples.

1.2 Result II: On the Influence of Sparsity for Boolean CSPs

Boolean constraint satisfaction has long served as a challenge for understanding the precise limits of our algorithmic and complexity-theoretic methods. The extensive list of such works includes classifications of tractability in terms of P vs NP-complete [27, 33, 10], counting complexity [9], parameterized complexity [22, 11], approximability [19] and many more. We wish to explore how sparsity affects the fine-grained time complexity investigated in [20].

Formally, we define, for any finite constraint family $\mathcal{F}$ and $\gamma > 0$, the algorithmic problem $\textsc{Csp}_k^\gamma(\mathcal{F})$: given a set of $m = \Theta(n^\gamma)$ constraints, each formed by applying some function $f : \{0,1\}^r \to \{0,1\} \in \mathcal{F}$ on a set of r pairwise distinct Boolean variables chosen from $x_1, \ldots, x_n$, determine whether there exists an assignment that sets precisely k variables to true and satisfies all constraints. For a singleton family $\mathcal{F} = \{f\}$, we also write $\textsc{Csp}_k^\gamma(f)$. Throughout the paper, we consider k as a constant independent of n.

Note that this class contains k-IS in h-uniform hypergraphs with $m = \Theta(n^\gamma)$ hyperedges as $\textsc{Csp}_k^\gamma(\mathbf{NAND}_h)$, where $\mathbf{NAND}_h(y_1, \ldots, y_h) = \overline{y_1 \wedge \cdots \wedge y_h}$. It also contains k-IS in mixed-arity hypergraphs as $\textsc{Csp}_k^\gamma(\{\mathbf{NAND}_2, \ldots, \mathbf{NAND}_k\})$.

We ask: *What determines the influence of sparsity for detecting size-k solutions of Boolean CSPs?*

Generally speaking, we obtain a full[5] classification for constraint families consisting exclusively of functions of binary arity. For higher-arity constraint functions, we give a large subclass of CSPs exhibiting a precise cutoff point: for sparser instances, the problem is trivial, for denser instances, it is essentially as hard as the dense case.

Classification for Binary Constraint Families

Without taking sparsity into account, previous works [22, 20] establish the following regimes within the Boolean constraint families: (1) problems fine-grained equivalent to $\textsc{Csp}_k(\mathbf{NAND})$, (2) problems fine-grained equivalent to $\textsc{Csp}_k(\mathbf{IMPL})$ where $\mathbf{IMPL}(x, y) = x \to y$, and (3) FPT problems.

For the two central hard CSPs, it is not too difficult to establish the following baselines (see Section 4):

1. $\textsc{Csp}_k^\gamma(\mathbf{NAND})$, i.e., k-IS in graphs, is trivial if $\gamma < 2$ and has complexity $O(n^{\omega k/3 \pm c})$ if $\gamma = 2$, assuming the clique hypothesis.
2. $\textsc{Csp}_k^\gamma(\mathbf{IMPL})$ is trivial if $\gamma < 1$ and has time complexity $n^{g(k)}$ uniformly for all $1 \leq \gamma \leq 2$, where $\Omega(\sqrt[3]{k}) \leq g(k) \leq O(\sqrt{k})$, assuming the clique hypothesis.

We find that surprisingly, the *combined* constraint family $\{\mathbf{IMPL}, \mathbf{NAND}\}$ has *higher* time complexity for all $1 \leq \gamma < 2$ than either **IMPL** or **NAND** individually.

▶ **Theorem 2.** *Assuming that the clique hypothesis holds. The optimal time complexity for $\textsc{Csp}_k^\gamma(\{\boldsymbol{NAND}, \boldsymbol{IMPL}\})$ is $\Theta(m^{\omega k/6 \pm c})$ for some c independent of k.*[6]

[5] Full with regard to the effects of sparsity, as the classification is not tight for $\textsc{Csp}_k^\gamma(\mathbf{IMPL})$, for which the precise complexity is unclear in the dense case already.

[6] By slight use of notation, in the introduction we write $\Theta(n^{f(k)\pm c})$ to express existence of an algorithm with running time $O(n^{f(k)+c})$ and a conditional lower bound of $\Omega(n^{f(k)-c})$.

Thus, for $1 \leq \gamma < 2$, while $\textsc{Csp}_k^\gamma(\mathbf{NAND})$ is trivial and $\textsc{Csp}_k^\gamma(\mathbf{IMPL})$ has a subexponential time complexity $n^{g(k)}$, the combination of both constraints has an exponential time complexity that is a natural interpolation of the k-IS running time.

The above theorem consists of two parts: (1) a conditional lower bound and (2) a matching algorithm. The conditional lower bound is strikingly simple: We can introduce $\approx n - n_0$ **IMPL**-constraints enforcing that any satisfying assignment with k nonzeroes chooses its nonzero variables from a set $V_0 \subseteq V$ of size n_0. On V_0, we embed an arbitrary, possibly dense instance as long as $n_0^2 = O(m)$. By choosing $n_0 \approx \sqrt{m}$, we can reduce from a k-clique instance in an $\sqrt{m}$-vertex graph, yielding a conditional lower bound of $m^{k\omega/6-o(1)}$, as desired.

The corresponding upper bound does not follow as easily. To beat time $O(n^{\omega k/3})$ when the input contains only few **NAND**- and **IMPL**-constraints, one might hope for a "reversal" of the reduction in the conditional lower bound: is it possible to quickly "peel off" most of the variables so that a small set of $O(\sqrt{m})$ variables survives? Unfortunately, possibly intricate interactions between **IMPL**- and **NAND**-constraints prevent simple preprocessing schemes. Instead, we perform a careful combination of arguments: We first preprocess the instance to obtain a structured setting in which crucially each variable implies at most one other variable. We use this structured setting to group the variables, yielding a win-win argument: Either (1) there exists a "large" group with few **NAND** constraints, and we can detect a solution within this single group greedily, or (2) all groups are "small". In this case, we carefully reduce to an almost-balanced Triangle Detection instance. For more details, we refer to Section 1.3 and the full proof in Section 4.

Interestingly, we observe that among the binary constraint families, the regime of $\textsc{Csp}_k^\gamma(\{\mathbf{NAND}, \mathbf{IMPL}\})$ is the only new regime that emerges. Specifically, we arrive at the following classification, illustrated in Figure 1.

▶ **Theorem 3** (Sparsity Classification for Binary Constraint Families). *Let $\mathcal{F}$ be a family consisting exclusively of binary constraint functions. For $0 \leq \gamma < 1$, $\textsc{Csp}_k^\gamma(\mathcal{F})$ can be solved in $\mathcal{O}(n)$. Furthermore,*

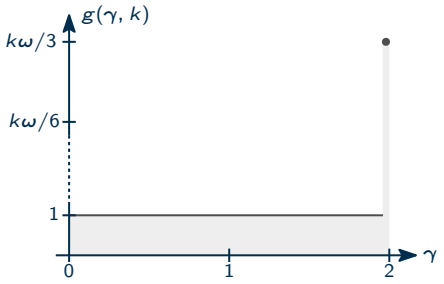
- *If $\mathcal{F} = \{\boldsymbol{NAND}\} \cup \mathcal{F}'$ for some non-empty family $\mathcal{F}'$, the optimal time complexity of $\textsc{Csp}_k^\gamma(\mathcal{F})$ for $1 \leq \gamma \leq 2$ is $\Theta(m^{\omega k/6 \pm c})$, assuming the clique hypothesis.*
- *If $\mathcal{F} = \{\boldsymbol{NAND}\}$, then $\textsc{Csp}_k^\gamma(\mathcal{F})$ is trivial for $\gamma < 2$. For $\gamma = 2$, the optimal time complexity is $\Theta(n^{\omega k/3 \pm c})$, assuming the clique hypothesis.*
- *If $\boldsymbol{NAND} \notin \mathcal{F}$, but $\boldsymbol{IMPL} \in \mathcal{F}$, then the optimal time complexity of $\textsc{Csp}_k^\gamma(\mathcal{F})$ for $1 \leq \gamma \leq 2$ is $n^{g(k)}$ with $\Omega(\sqrt[3]{k}) \leq g(k) \leq O(\sqrt{k})$, assuming the clique hypothesis.*
- *Finally, if $\boldsymbol{IMPL}, \boldsymbol{NAND} \notin \mathcal{F}$, then $\textsc{Csp}_k^\gamma(\mathcal{F})$ can be solved in time $O(n+m)$ for all $1 \leq \gamma \leq 2$.*

We prove the first part in Section 4, deferring details on the remaining parts to the full paper due to space constraints.

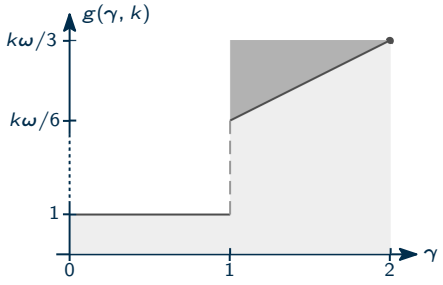
Towards Higher-Arity Constraint Families: Phase Transition At Triviality Cutoff

Classifying the influence of sparsity for higher-arity constraint families becomes significantly more challenging. Already the question how many constraints are minimally required to define a non-trivial instance is not obvious. We shall however see, that for a large class of constraint families, this quantity is decisive to understand the time complexity, as we observe a phase transition at this *triviality cutoff*.

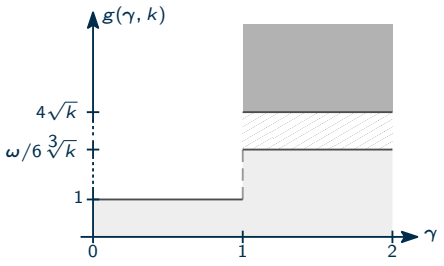
Specifically, we say that $\textsc{Csp}_k^\gamma(\mathcal{F})$ is *non-trivial* if there are infinitely many NO instances in $\textsc{Csp}_k^\gamma(\mathcal{F})$; conversely, it is *trivial*, if for all sufficiently large n, any instance of $\textsc{Csp}_k^\gamma(\mathcal{F})$ with n variables is a YES instance. For any constraint family $\mathcal{F}$, we define the *triviality*



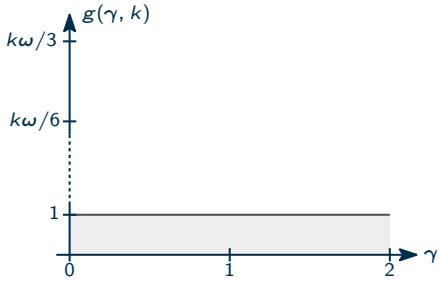
(a) $\textsc{Csp}_k^\gamma(\mathbf{NAND})$, equivalent to k-Independent Set.



(b) $\textsc{Csp}_k^\gamma(\{\mathbf{NAND}\} \cup \mathcal{F}')$ for any binary, non-trivial family $\mathcal{F}'$.



(c) $\textsc{Csp}_k^\gamma(\{\mathbf{IMPL}\} \cup \mathcal{F}')$ for any binary family with $\mathbf{NAND} \notin \mathcal{F}'$.



(d) $\textsc{Csp}_k^\gamma(\mathcal{F})$ for any binary family with $\mathbf{NAND}, \mathbf{IMPL} \notin \mathcal{F}$.

Figure 1 Relationship between runtime and density for our different regimes: Given some density γ and family $\mathcal{F}$ we obtain upper and conditional lower bounds of the form $n^{g(\gamma,k)}$. For improved readability, we state the exponent $g(\gamma, k)$ up to an additive constant independent of k, γ. Under this tolerance, these regimes are tight, except for case *(c)* where the knowledge gap between $n^{\Omega(\sqrt[3]{k})}$ and $n^{O(\sqrt{k})}$ from the dense case transfers.

cutoff γ_{triv} as the smallest $\gamma \geq 0$ such that $\textsc{Csp}_k^\gamma(\mathcal{F})$ is non-trivial. Thus, every n-variable instance with sufficiently large n and $O(n^{\gamma_{\text{triv}}-\epsilon})$ constraints from $\mathcal{F}$ contains a solution, while there exists infinitely many instances with $\Theta(n^{\gamma_{\text{triv}}})$ constraints from $\mathcal{F}$ that do not contain a solution with k nonzeroes.

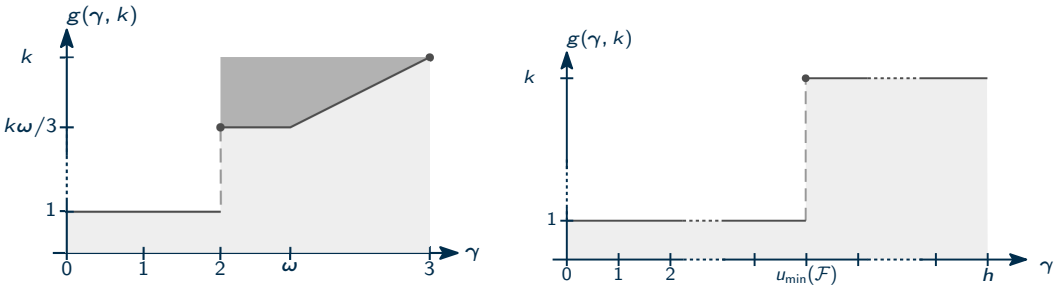
We first introduce a parameter $u_{\min}(\mathcal{F})$ that describes this triviality cutoff. Specifically, let $u_{\min}(f)$ denote the smallest weight $\|x\|_1$ of an assignment x violating f, i.e., $u_{\min}(f) = \min\{\|x\|_1 \mid f(x) = 0\}$. Defining $u_{\min}(\mathcal{F}) \coloneqq \min_{f \in \mathcal{F}} u_{\min}(f)$, we can establish the triviality cutoff γ_{triv} as $u_{\min}(\mathcal{F})$.

▶ **Theorem 4** (Triviality Cutoff). *Let $\mathcal{F}$ be any finite constraint family. The problem $\textsc{Csp}_k^\gamma(\mathcal{F})$ with $\gamma = u_{\min}(\mathcal{F})$ is non-trivial. Conversely, if $\gamma < u_{\min}(\mathcal{F})$, then there exists n_0 such that any instance of $\textsc{Csp}_k^\gamma(\mathcal{F})$ with at least n_0 variables admits a satisfying assignment of weight k.*

For a large class of constraint families, we obtain a sharp phase transition from trivial instances to requiring brute force running time, assuming the 3-uniform hyperclique hypothesis. For an illustration, see Figure 2.

▶ **Theorem 5** (Phase Transition). *Let $\mathcal{F}$ be a constraint family. If $u_{\min}(\mathcal{F}) \geq 3$, then $\textsc{Csp}_k^\gamma(\mathcal{F})$ is trivial for $\gamma < u_{\min}(\mathcal{F})$. For $\gamma \geq u_{\min}(\mathcal{F})$, it requires time $n^{k-o(1)}$ assuming the $u_{\min}(\mathcal{F})$-uniform hyperclique hypothesis.*

To future work, we leave the challenge of settling the influence of sparsity for constraint families $\mathcal{F}$ with $u_{\min}(\mathcal{F}) \in \{0, 1, 2\}$. As witnessed by the mixed-arity k-IS family $\mathcal{F} = \{\mathbf{NAND}_2, \mathbf{NAND}_3\}$ (which has $u_{\min}(\mathcal{F}) = 2$), settling such families can become technically quite demanding.



(a) $\textsc{Csp}_k^\gamma\big(\{\mathbf{NAND}_2, \mathbf{NAND}_3\}\big)$ equivalent to k-Independent Set in 3-hypergraphs.

(b) $\textsc{Csp}_k^\gamma(\mathcal{F})$ for any family $\mathcal{F}$ with $u_{\min}(\mathcal{F}) \geq 3$.

Figure 2 Relationship between runtime and sparsity for $\textsc{Csp}_k^\gamma(\mathcal{F})$ with higher-arity $\mathcal{F}$: (a) For $\mathcal{F} = \{\mathbf{NAND}_2, \mathbf{NAND}_3\}$, the complexity displays a nuanced relationship (a kinked slope) in the regime $2 \leq \gamma \leq 3$; note that $u_{\min}(\mathcal{F}) = 2$. (b) showcases the sharp phase transition at the triviality cutoff for or families $\mathcal{F}$ with $u_{\min}(\mathcal{F}) \geq 3$.

1.3 Detailed Results and Technical Overview

We highlight our main technical contributions, and provide a more detailed overview of our results. We begin by presenting our algorithmic contributions for k-Independent Set problem in 3-hypergraph and developing the core framework that will also be useful in extending to general hypergraphs of higher arity.

Algorithm for k-Independent Set in Sparse 3-Hypergraphs

Let H be a 3-Hypergraph[7] with n vertices and m hyperedges. Let G be the underlying graph of H, obtained by removing all hyperedges of arity 3 from H. Notice that any k-independent set in H is also a k-independent set in G. We refer to the k-independent sets in G as *potential solutions*. Formally, let $I(G,k)$ denote the set of all independent sets of size k in G and let $I_{\text{invalid}}(H,k)$ denote the set of all *false solutions*, i.e. the set of all potential solutions that contain a hyperedge in H. We begin with a simple observation that H has a k-independent set if and only if $|I(G,k)| - |I_{\text{invalid}}(H,k)| > 0$.

Counting the potential solutions, i.e., the value $|I(G,k)|$ can be done via the classical algorithm of Nešetřil and Poljak [24], yielding an $O(n^{\frac{k\omega}{3}})$ bound (if k is divisible by 3). Hence, the main algorithmic challenge is to efficiently count the invalid solutions, namely those potential solutions that violate at least one hyperedge constraint in H. We now turn our attention to this task. For any subset of hyperedges S, let $I_S(H,k)$ denote the set of all potential solutions that contain all vertices spanned by S; that is, for each $X \in I_S(H,k)$, the induced hypergraph $H[X]$ contains all hyperedges in S. Let $E_{\geq 3}$ denote the set of all hyperedges in H that contain at least three vertices. We can now observe that, by definition of I_{invalid}, we have

$$I_{\text{invalid}}(H,k) = \bigcup_{e \in E_{\geq 3}} I_{\{e\}}(H,k),$$

and for any non-empty set $S \subseteq E_{\geq 3}$

$$I_S(H,k) = \bigcap_{e \in S} I_{\{e\}}(H,k).$$

[7] That is, each hyperedge contains *at most* 3 vertices.

This gives us a natural way to compute the size of I_{invalid} via the inclusion-exclusion principle as follows.

$$|I_{\text{invalid}}(H,k)| = \sum_{i=1}^{\binom{k}{3}} \sum_{S \in \binom{E_{\geq 3}}{i}} (-1)^{i+1} |I_S(H,k)|.$$

However, naively enumerating all sets S and computing $|I_S(H,k)|$ fails to improve upon trivial $O(n^k)$ brute-force algorithm, even in the sparse regime $m = O(n)$. To circumvent this issue, we notice that a lot of the "higher-order" terms in the inclusion-exclusion formula are either irrelevant, or redundant. Specifically, such terms either (1) span more than k vertices (irrelevant), or (2) can be obtained by appropriately guessing a "lower-order" term (redundant). Case in point: consider a set S that contains $\binom{k}{3}$ many edges. The only way that S spans k vertices is if these vertices induce a clique of size k in H. However, in that case, for any set $S' \subset S$ that contains $k/3$ pairwise non-intersecting hyperedges will span the same vertex set and consequently $I_{S'}(H,k) = I_S(H,k)$

This observation will allow us to drastically reduce the number of inclusion–exclusion terms that must be considered, and form the basis of our improved counting algorithm. Intuitively, it brings us to the following win-win scenario: let $S \subset E_{\geq 3}$ of size $> k/3$. Then either (i) the hyperedges in S are "clustered together" and we can get away by guessing a smaller set $S' \subset S$ (S is redundant), or (ii) the hyperedges in S are not "clustered together", but they span more than k vertices, so we never have to consider S at all (S is irrelevant). However, the issue of "clustered edges" cannot be resolved by simply restricting our outer sum to stop at $k/3$, as that would generally lead to a lot of double counting. To bypass this problem, the idea is to impose a total ordering $\prec$ on the hyperedges of H. Intuitively, whenever a hyperedge $e \in S$ intersects another hyperedge $e' \in E_{\geq 3}$ with $e' \prec e$, the algorithm recognizes that all the independent sets containing $V(e) \cup V(e')$ have been accounted for in another iteration S^* that contains e', so it avoids counting them again. It remains to formalize and implement this idea efficiently, which we address next.

Avoiding double counting. Fix an arbitrary total order $\prec$ on the hyperedges of H. For any nonempty set of hyperedges S, let $I'_S(H,k)$ denote the set of all potential solutions X of size k that satisfy the following two conditions.

1. X contains all vertices spanned by the hyperedges in S.
2. For every $e \in S$ and every hyperedge $e' \in E_{\geq 3}$ with $e \cap e' \neq \emptyset$, if $e' \prec e$, then X does not contain e'.

We claim that replacing I_S by I'_S in the inclusion-exclusion formula preserves correctness. In particular, we show that

$$I_{\text{invalid}}(H,k) = \bigcup_{e \in E_{\geq 3}} I'_{\{e\}}(H,k)$$

The rough idea is as follows. Consider any $X \in I_{\text{invalid}}(H,k)$. By definition, the induced subhypergraph $H[X]$ contains at least one hyperedge (and possibly as many as $\binom{k}{3}$). Let e be the minimum hyperedge in $H[X]$ with respect to the ordering $\prec$; that is, for every edge $e' \in E(H[X]) \setminus \{e\}$, we have $e \prec e'$. By construction, X satisfies both defining conditions of $I'_{\{e\}}(H,k)$, and hence $X \in I'_{\{e\}}(H,k)$. Particularly, this shows that $I_{\text{invalid}}(H,k) \subseteq \bigcup_{e \in E_{\geq 3}} I'_{\{e\}}(H,k)$. The containment in the other direction follows immediately by noticing that for each $S \subseteq E_{\geq 3}$, we have $I'_S(H,k) \subseteq I_S$ and as argued previously, $I_{\text{invalid}}(H,k) = \bigcup_{e \in E_{\geq 3}} I_{\{e\}}(H,k)$. Together, these inclusions imply the claimed equality.

We further observe that any set $S \subseteq E_{\geq 3}$ containing more than $k/3$ hyperedges either: (1) spans more than k vertices, or (2) contains two hyperedges e, e' such that $e \cap e' \neq \emptyset$. In the first case, since $|V(S)| > k$, no independent set of size k can contain S. Also, in the second case either $e \prec e'$, or $e' \prec e$. In either case, any potential solution in $I_S(H,k)$ containing both e and e' violates the second condition in the definition of $I'_S(H,k)$. In particular, in both cases the set $I'_S(H,k)$ is empty. Consequently, all "higher-order" inclusion-exclusion terms corresponding to $|S| > k/3$ vanish. Combining this with the observations above, we have the following refinement of our inclusion-exclusion formula.

$$|I_{\text{invalid}}(H,k)| = \sum_{i=1}^{\lceil \frac{k}{3} \rceil} \sum_{S \in \binom{E_{\geq 3}}{i}} (-1)^{i+1} |I'_S(H,k)|.$$

The remaining challenge is that, unlike the original sets $I_S(H,k)$, the sets $I'_S(H,k)$ do not admit an obvious efficient counting algorithm. We are now going to prove that we can indeed compute their sizes efficiently.

Computing $|I'_S(H,k)|$. Fix a set $S \subset E_{\geq 3}$ and consider any hyperedge $e' \in E_{\geq 3}$. Suppose that there exists a hyperedge $e \in S$ such that (i) $e \cap e' \neq \emptyset$, and (ii) $e' \prec e$. In this case, any potential solution containing e' would violate the second condition in the definition of $I'_S(H,k)$, and we enforce this restriction via the following reduction.

If $|e' \setminus e| = 1$, we delete the unique vertex in $e' \setminus e$ from the hypergraph. If instead $|e' \setminus e| = 2$, we replace the hyperedge e' by an edge connecting the two vertices in $e' \setminus e$. Let H' be the hypergraph obtained from H by iterating this reduction over all hyperedges $e' \in E_{\geq 3}$. We show that this transformation preserves exactly the desired solutions, namely that $I'_S(H,k) = I_S(H',k)$. Recall that the set $I_S(H',k)$ can be computed by running the Nešetřil, Poljak [24] k-clique algorithm on an appropriate subgraph of the underlying graph of H'. This yields an efficient procedure for computing $|I'_S(H,k)|$ and consequently an efficient way to compute $|I_{\text{invalid}}(H,k)|$. By implementing this carefully, we can bound the running time for computing this value as follows (assuming k is divisible by 3, otherwise we get a small polynomial overhead).

$$T_k(n,m) \leq \mathcal{O}\left(m^{\frac{k}{3}} + \sum_{i=0}^{\frac{k}{3}-1} m^i \left(m + n^{(k-3i)\omega/3} \right) \right) \leq \mathcal{O}\left(m^{\frac{k}{3}} + n^{k\omega/3} \right).$$

For more details on the implementation, as well as the detailed computation on this bound, we refer the reader to Section 3.1.

We complement this algorithmic result with matching conditional lower bounds. Using a technique that we call *sparse embedding*, that maps a small dense instance of k-Hyperclique Detection in 3-uniform hypergraphs into a sparse instance of k-Independent Set Detection in 3-hypergraphs, we show that for any density $m = \Theta(n^\gamma)$, $2 \leq \gamma \leq 3$, no algorithm can run in time $\mathcal{O}(m^{k/3-\varepsilon})$, unless the 3-Uniform Hyperclique Hypothesis fails. Moreover, the lower bound of $n^{k\omega/3-o(1)}$ is inherited directly from hardness of k-Independent Set Detection in graphs. Combining these two lower bounds establishes the conditional optimality of our algorithm across the entire sparsity spectrum. [8]

[8] When $m = \Theta(n^\gamma)$ for some $\gamma < 2$, every such instance is a trivial yes-instance as a consequence of Turán's theorem.

Extending the Algorithm to General h-Hypergraphs

Let H be an h-Hypergraph for some $h \geq 3$, with n vertices and $m = \sum_{i=2}^{h} m_i$ hyperedges, where each m_i is the number of hyperedges of arity i. The goal is to extend the algorithmic framework of detecting k-Independent Sets in 3-Hypergraphs to this more general setting. We first show that a straightforward extension of the techniques developed for 3-hypergraphs yields an improvement over the brute-force $O(n^k)$ running time for general hypergraphs, provided that $m_i \leq O(n^{3-\varepsilon_i})$ for every arity i (and consequently $m = O(n^{3-\varepsilon})$). More precisely, we prove the following proposition, which generalizes the algorithm for 3-hypergraphs and provides a unified upper bound for k-Independent Set in general h-hypergraphs.

▶ **Proposition 6.** *Given any h-hypergraph H with n vertices and m edges, for every k divisible by 3, there is an algorithm deciding if H contains a k-independent set in time* $\mathcal{O}\left(\min\left\{n^k, n^{\frac{k\omega}{3}} + m^{\frac{k}{3}}\right\}\right)$.

On a high level, we are taking the same blueprint as in the 3-hypergraph case: we count the potential solutions and subtract the number of invalid solutions. Recall,

$$|I(H,k)| = |I(G,k)| - |I_{\text{invalid}}(H,k)|,$$

where G denotes the underlying graph of H. While the value $|I(G,k)|$ can still be computed efficiently, evaluating $|I_{\text{invalid}}(H,k)|$ becomes more complicated as we increase the arity. As before, we employ the inclusion-exclusion formula, and eliminate the higher order terms in the same way, by using the following formula.

$$|I_{\text{invalid}}(H,k)| = \sum_{i=1}^{\lceil \frac{k}{3} \rceil} \sum_{S \in \binom{E_{\geq 3}}{i}} (-1)^{i+1} |I'_S(H,k)|.$$

However, the main difficulty lies in computing $|I'_S(H,k)|$. Recall that for 3-hypergraphs, this was achieved by imposing an ordering on hyperedges, and applying the two simple reduction rules,thereby reducing the problem to computing a small independent set in an appropriate graph. For hypergraphs of higher arity, however, when two hyperedges e, e' intersect, the set $e' \setminus e$ might contain more than two vertices, so simply adding an edge of arity 2 no longer suffices. A natural generalization of the "type 2" reduction rule is to replace e' by a hyperedge spanning $e' \setminus e$. While this preserves correctness of the construction, crucially, it reduces the problem only to detecting an independent set in hypergraphs, and we can no longer apply the matrix-multiplication-based algorithm to improve upon trivial runtime (see [21] for a detailed discussion). Fortunately, we can circumvent this issue by observing that the new hypergraph has strictly smaller arity than H, allowing us to apply induction on the arity, with the 3-hypergraph algorithm as a base case.

In fact, we improve upon this by using a combination of the three techniques (1) partitioning the hyperedges of H into "sparse" and "dense" components (2) applying the inclusion-exclusion procedure described above to the sparse part (3) using sparse witness enumeration to handle the dense parts. With a careful implementation of these techniques, and by addressing several technical obstacles, we can finally prove the following upper bound for the k-Independent Set problem on general hypergraphs.

▶ **Theorem 7** (k-Independent Set Algorithm for Higher Arity Hypergraphs)**.** *Given any h-hypergraph H with n vertices and $m_i = \Theta(n^{\gamma_i})$ edges of arity i (for each $i \geq 2$), there is an algorithm deciding if H contains a k-independent set for any k divisible by 3 in time*

$$\mathcal{O}\left(n^{\frac{k\omega}{3}} + \sum_{i=3}^{h} \min\left\{m_i^{\lceil \frac{k-i+3}{3} \rceil}, m_i n^{k-i}\right\}\right).$$

Perhaps surprisingly, we show that this upper bound is essentially tight: unless either the k-Clique Hypothesis, or the 3-Uniform Hyperclique Hypothesis fails, no significantly faster algorithm is possible. More specifically, by employing appropriate notions of sparse and dense embeddings, we can prove the following theorem.

▶ **Theorem 8** (Arity-Sensitive Lower Bound). *Let $\varepsilon > 0$ be arbitrary and n be the number of vertices of any given h-hypergraph. For any $3 \leq i \leq h$, let $m_i = \Theta(n^{\gamma_i})$ for any $2 \leq \gamma_i \leq i$ be the number of hyperedges of arity i of the input hypergraph. Then:*

1. (Clique LB) *There is no algorithm solving k-Independent Set problem in h-hypergraphs in time $\mathcal{O}\left(n^{\frac{k\omega}{3}-\varepsilon}\right)$, assuming the k-Clique Hypothesis.*
2. (3-Uniform Hyperclique LB) *For no i such that $2 \leq \gamma_i \leq 3$ is there an algorithm solving the problem in time $\mathcal{O}\left(m_i^{\frac{k-i+3}{3}-\varepsilon}\right)$, assuming the 3-Uniform Hyperclique Hypothesis.*
3. (r-Uniform Hyperclique LB) *For no i such that $\gamma_i \geq 3$ is there an algorithm running in $\mathcal{O}\left(m_i n^{k-i-\varepsilon}\right)$, assuming the $(\lfloor \gamma_i \rfloor)$-Uniform Hyperclique Hypothesis.*

Beyond Independent Sets: Boolean CSP

To study the effect of sparsity systematically in a more general class of problems, we turn to the class of Boolean Constraint Satisfaction Problems (CSPs). Recall that k-Independent Set in h-hypergraphs can be viewed as a special case of Boolean CSP over the constraint family $\mathcal{F} = \{\mathbf{NAND}_2, \ldots, \mathbf{NAND}_h\}$. We formally define this class of problems as follows.

▶ **Definition 9** (Boolean Constraint Satisfaction Problem ($\textsc{Csp}_k^\gamma$)). *Let $\mathcal{F}$ be a finite Boolean constraint family (i.e. a set of functions $f : \{0,1\}^h \to \{0,1\}$). Given a set Φ of m Boolean constraints C on variables $x_1, \ldots, x_n$, each of the form $f(x_{i_1}, \ldots, x_{i_h})$, where $f \in \mathcal{F}$ and $m \in \Theta(n^\gamma)$, the problem $\textsc{Csp}_k^\gamma$ asks if there exists an assignment $a : \{x_1, \ldots, x_n\} \to \{0,1\}$ satisfying Φ, that sets precisely k variables to 1.*

We begin by considering the special case binary constraints families. Perhaps surprisingly, for every such family $\mathcal{F}$, and for all values of γ, we obtain a complete classification of the problem $\textsc{Csp}_k^\gamma(\mathcal{F})$: we design an algorithm running in time $T_k(m,n)$, and show its conditional optimality under the k-Clique Hypothesis. To obtain such a complete classification across the entire sparsity spectrum, we show that instances of $\textsc{Csp}_k^\gamma(\mathcal{F})$ for values $\gamma < 1$, admit a linear time algorithm: $\mathcal{O}(n+m)$. The classification becomes much more interesting when we consider the regime $\gamma \geq 1$. In particular we prove the following classification theorem, which is a more formal version of Theorem 3.

▶ **Theorem 10** (Classification of Boolean CSPs over Binary Families). *Let $\mathcal{F}$ be a family of binary Boolean constraints. Then, for any $1 \leq \gamma \leq 2$ we obtain the following classification:*

1. (Linear Regime) *If $\mathcal{F}$ contains neither **NAND** nor **IMPL**, then $\textsc{Csp}_k^\gamma(\mathcal{F})$ can be solved in linear time $\mathcal{O}(m+n)$.*
2. (Subexponential Regime, [20]) *If $\mathcal{F}$ contains **IMPL**, but not **NAND**, then $\textsc{Csp}_k^\gamma(\mathcal{F})$ can be solved in time $\mathcal{O}\left(n^{4\sqrt{k}}\right)$, furthermore, unless the k-clique hypothesis fails, there is no algorithm solving $\textsc{Csp}_k^\gamma(\mathcal{F})$ in time $\mathcal{O}\left(n^{\omega \sqrt[3]{k}/6+c-\varepsilon}\right)$, for any $\varepsilon > 0$ and some $c > 0$ independent of k.*
3. (k-IS Regime) *If $\mathcal{F} = \{\mathbf{NAND}\}$, then for any $\gamma < 2$, every $\textsc{Csp}_k^\gamma(\mathcal{F})$ instance is trivial. For $\gamma = 2$, there is an algorithm solving $\textsc{Csp}_k^\gamma(\mathcal{F})$ in time $\mathcal{O}\left(n^{\omega k/3}\right)$ (for any k divisible by 3). Furthermore, any algorithm running in $\mathcal{O}\left(n^{\omega k/3-\varepsilon}\right)$ would refute the k-Clique Hypothesis.*

4. (Clique Regime) *If for some nonempty family $\mathcal{F}'$, $\mathcal{F}$ can be written as $\mathcal{F} = \{\boldsymbol{NAND}\} \cup \mathcal{F}'$, then we can solve $\textsc{Csp}_k^\gamma(\mathcal{F})$ in time $\mathcal{O}\left(m^{\omega(k-c_{\mathcal{F}'})/6+1}\right)$ (for all sufficiently large k divisible by 3), where $c_{\mathcal{F}'} \in \{0,1,2\}$ depends only on $\mathcal{F}'$. Moreover, any algorithm running in $\mathcal{O}\left(m^{\omega(k-c_{\mathcal{F}'})/6-\varepsilon}\right)$ would refute the k-Clique Hypothesis.*

By a clever branch and bound approach given in [22], we can generally resolve 0-invalid constraint functions, essentially allowing to reduce constraint families in the linear regime to $\{\mathbf{EQ}\}$ which in turn can be reduced to a bounded subset-sum instance. For the Subexponential Regime, a fairly simple argument suffices to show that $\textsc{Csp}_k^\gamma(\mathbf{IMPL})$ can be reduced to $\textsc{Csp}_k^1(\mathbf{IMPL})$, thus it is unlikely to be improved by exploiting sparsity. Note that this regime is not fully resolved in the dense case, this gap therefore transfers to the sparse setting as well. For the k-IS Regime, triviality follows by Turán's Theorem. The precise details on Regime 1-3 can be found in the full version of the paper.

The Clique Regime turned out to be the most interesting. On the hardness side, we can reduce the dense case to the sparse case using only **NAND** and one further binary constraint. We reduce the density by blowing up the number of variables accordingly, carefully ensuring that the additional variables are not part of a solution. Here we once again refer to the full version. For the algorithm, we essentially argue that for each family $\mathcal{F}$ that belongs to the clique regime we can reduce any instance of $\textsc{Csp}_k^\gamma(\mathcal{F})$ efficiently to $\textsc{Csp}_k^\gamma(\{\mathbf{NAND}, \mathbf{IMPL}\})$, so it suffices to construct an efficient algorithm for this particular family. Our algorithm proceeds in a few steps that when implemented carefully, yield the desired running time.

(1) (*Reduction to Restricted Instance*) We first leverage the directed reachability structure induced by **IMPL** constraints: intuitively, vertices with many descendants are cheap to guess, since including such a vertex in a solution forces all descendants to be set as well. By systematically branching on these vertices and propagating their implications, we can reduce to a *restricted* instance, where each vertex has at most a single descendant (besides itself) in the directed graph induced by **IMPL** constraints.

(2) (*Grouping and Sparsity Cutoff*) For such restricted instances, we introduce a grouping technique to partition the variables according to their local **IMPL** structure. Each group corresponds to a small implication neighborhood that can be treated as a single unit. We then establish a *sparsity cutoff*: if any group is sufficiently large and therefore sufficiently sparse with respect to **NAND** constraints, we immediately find a solution using Turán-like arguments. As a consequence, we can bound the size of each group and the total number of groups, reducing the problem to a *core instance*.

(3) (*Removing the 2-Cycles and Reduction to Triangle Detection*) For our last step, we want our Implication-induced graph to be acyclic. We prove the remaining 2-cycles can only be contained within a small subset of vertices and thus eliminated efficiently. Then, we show how to distribute the groups in a balanced manner and encode valid group selections as vertices in an auxiliary graph. This construction reduces the problem to Triangle Detection, allowing us to leverage matrix multiplication to achieve the desired running time.

Triviality and Hardness beyond Binary Constraints

In the binary setting we provide an (almost) complete classification of all binary families, condensing the limited landscape into few interesting regimes. We tackle families of unbounded arity by observing a specific parameter that is exploitable both in terms of lower and upper bounds. Precisely, we consider the minimum weight $u_{\min}(f)$ required to violate a constraint function f, i.e. let $u_{\min}(f) \coloneqq \min\{\|x\|_1 \mid f(x) = 0\}$ and $u_{\min}(\mathcal{F}) \coloneqq \min_{f \in \mathcal{F}} u_{\min}(f)$.

Equipped with this parameter, we generalize Turan's theorem [28] to Boolean CSPs of any arity via a constructive algorithm and define a surprisingly simple gadget to proof hardness conditioned on the ($h-$uniform) $k-$(hyper)clique hypothesis.

Triviality. Turan's theorem states for constant k and large n, there always exists a k-independent set in a simple graph with $m \in \mathcal{O}(n^{2-\varepsilon})$ edges. Finding this independent consists of two steps: Since $m \in \mathcal{O}(n^{2-\varepsilon})$, there exists a vertex with $\mathcal{O}(n^{1-\varepsilon})$ neighbors. Adding this vertex and removing its neighborhood for correctness, we obtain a sparse graph again, allowing us to perform this procedure k times in total.

Despite the simplicity of this approach it is not trivial how this transfers to high arity Boolean CSPs: What is a *low degree* variable in this context and how do we handle its neighborhood? To answer this, we consider the density of each constraint function individually: We define γ_f for all $f \in \mathcal{F}$ such that $m_f \in \Theta(n^{\gamma_f})$ where m_f denotes the number of f-constraints. With this we can state the precise triviality cutoff:

▶ **Theorem 11** (Special Triviality). *Let $\mathcal{F}$ be any finite constraint family and Φ an instance of $\textsc{Csp}_k^\gamma(\mathcal{F})$ with $\gamma_f < u_{\min}(f)$ for all $f \in \mathcal{F}$. Then Φ is trivial and we can find a size-k solution in time $\mathcal{O}(n+m)$.*

The algorithm is a generalization of the one described above for k-independent set in sparse graphs. We include a low degree variable into the solution, adjust its neighborhood, and then repeat the process $k-1$ further times. The sparsity requirement implies a variable v that is contained in at most $\mathcal{O}(n^{\gamma_f - 1})$ many f-constraints for all f with $\gamma_f \geq 1$ and contained no g-constraints with $\gamma_g < 1$. To adjust the neighborhood we replace each constraint $c = f(\dots, v, \dots)$ with a new constraint $c' = f(\dots, 1, \dots)$. This maintains both correctness and our desired sparsity property. For the full proof we refer to the arXiv version.

Hardness. To prove hardness and non-triviality, a rather simple gadget suffices. For some function f of arity h we can can place f-constraints on all size-h subsets of $h+k$ variables. One can then prove this gadget resembles $\mathbf{NAND}_{u_{\min}(f)}$, as all satisfying assignments set any number of variables that $\leq u_{\min}(f)$ to true. Carefully applying this gadget then allows us to both prove non-triviality and also conditional lower bounds based on the k-(hyper)clique hypothesis.

▶ **Theorem 12** (Non-Triviality and Clique-Hardness). *Let $\mathcal{F}$ be any finite constraint family.*

- *$\textsc{Csp}_k^\gamma(\mathcal{F})$ is non-trivial for $\gamma = u_{\min}(\mathcal{F})$*
- *If $\gamma \geq u_{\min}(\mathcal{F}) = 2$ then $\textsc{Csp}_k^\gamma(\mathcal{F})$ requires time $n^{k\omega/3-o(1)}$ assuming the k-clique hypothesis.*
- *If $\gamma \geq u_{\min}(\mathcal{F}) \geq 3$ then $\textsc{Csp}_k^\gamma(\mathcal{F})$ requires time $n^{k-o(1)}$ assuming the $u_{\min}(\mathcal{F})$-uniform k-hyperclique hypothesis.*

We again refer to full version for the proof and combine the two above theorems to obtain Theorem 4 and Theorem 5.

2 Preliminaries

For any natural number $x \in \mathbb{N}$, we denote with $[x]$ the set of integers $\{1, \dots, x\}$. Further, for any set S and any number $d \in \mathbb{N} \cup \{0\}$ we denote with $\binom{S}{d}$ the set of all subsets of S of size d. With $\binom{S}{\leq d}$ we denote the set of all subsets of S of size $\leq d$. An *h-hypergraph* is any hypergraph $H = (V, E)$ such that $E \subseteq \binom{V}{\leq h} \setminus \binom{V}{\leq 1}$. An *$h$-uniform hypergraph* is any h-hypergraph with $E \subseteq \binom{V}{h}$.

Hardness Assumptions. The k-clique problem asks, given a graph $G = (V, E)$ on n vertices, to determine whether there exists a k-clique $C \subset V$ of size k such that $\binom{C}{2} \subseteq E$. Dating back to [24], one can detect k-cliques time $\mathcal{O}(n^{\frac{\omega k}{3}})$ for k divisible by 3. Here, $\omega < 2.372$ denotes the matrix multiplication exponent, such that we can multiply two $n \times n$ matrices in time $\mathcal{O}(n^{\omega})$.

This problem generalizes to r-uniform hypergraphs, asking to find a set of vertices such that every r tuple is contained in a hyperedge – but algebraic approaches such as fast matrix multiplication fail in this setting (for a discussion of this phenomenon, see [21]). As such, no algorithm with substantial improvements over brute-force time $n^{k-o(1)}$ is known (see [3] for subpolynomial-factor improvements). In fact, any polynomial improvement would result in improved algorithms for problems such as Max-h-SAT [31] that are widely believed to be hard.

In fine-grained complexity theory, the k-clique problem has been widely used to obtain conditional lower bounds, such as e.g. [1, 12, 14]. Over the recent years, the r-uniform k-hyperclique problem has seen increasing popularity as well, yielding applications for a variety of problems [21, 2, 7, 20, 8, 13, 23, 16, 17].

As such we use the following hypothesis concerning k-(hyper)clique detection:

▶ **Hypothesis 13** (r-Uniform d-Hyperclique Hypothesis). *Let $\epsilon > 0$ and $k > d$.*

1. *For $d = 2$ there is no $\mathcal{O}(n^{(\omega k/3)-\varepsilon})$-time algorithm detecting a k-clique in a graph (also referred to as k-clique hypothesis).*
2. *For $d \geq 3$, there is no $\mathcal{O}(n^{k-\varepsilon})$-time algorithm detecting a k-clique in a d-uniform hypergraph.*

3 k-Independent Set in Non-Uniform Hypergraphs

This is our main technical section. It is dedicated to constructing algorithms and lower bounds for detecting a k-independent set in (non-uniform) h-hypergraphs.[9] Note that throughout this section we treat edges as subsets of vertices and use standard set-theory notation (e.g. $e \cap e'$, $e \setminus e'$, $V \subseteq \bigcup_{e \in E} e$). We begin by considering the simplest family of h-hypergraphs, namely the 3-hypergraphs.

3.1 k-Independent Set in 3-Hypergraphs

In this section, we prove the following two main theorems.

▶ **Theorem 14** (Algorithm for k-Independent Set Problem in 3-Hypergraphs). *Given any 3-hypergraph H with n vertices and m edges, there is an algorithm deciding whether H contains a k-independent set in time $\mathcal{O}\left(n^{\frac{k\omega}{3}} + m^{\frac{k}{3}}\right)$ for all k divisible by 3.*

Moreover, we show that this running time is essentially optimal, unless at least one of the two established hypotheses fails.

▶ **Theorem 15** (Conditional Lower Bounds for k-Independent Set Problem in 3-Hypergraphs). *There is no algorithm solving k-Independent Set problem in 3-hypergraphs in time:*

1. *$\mathcal{O}\left(n^{\frac{k\omega}{3}-\varepsilon}\right)$, unless the k-Clique Hypothesis fails.*
2. *$\mathcal{O}\left(m^{\frac{k}{3}-\varepsilon}\right)$, unless the 3-Uniform Hyperclique Hypothesis fails. Moreover, this holds even when restricting $m = \Theta(n^{\gamma})$, for any $2 \leq \gamma \leq 3$.*[10]

[9] I.e., each hyperedge has arity *at most* h.

[10] Recall that Turan's theorem implies that if $\gamma < 2$, any instance is a trivial yes-instance. Hence, this theorem gives us a full landscape of the complexity of the problem in terms of both number of vertices and number of (hyper)edges.

We dedicate the first part of this section to constructing our algorithm and proving Theorem 14. Our approach consists of first counting the *potential solutions*, which are in principle all independent sets on the underlying (2-uniform) graph of H (intuitively, in the first step we ignore arity-3 hyperedges), and then using inclusion-exclusion to count all *false potential solutions*, by which we understand those independent sets of the underlying graph of H which contain an arity-3 hyperedge in H. While this approach yields a correct solution, a naive implementation is unfortunately too slow. Intuitively, if many arity 3-edges are clustered together, we would spend too much time counting the higher-order terms in the inclusion-exclusion formula. The last step of our algorithm takes care of this by making sure that we never double-count the solutions from such clustered hyperedges, assuring that we only need to compute the higher order terms of the "nicely structured" hyperedges, yielding the desired running time. We start with the following simple observation.

▶ **Observation 16.** *Let H be a 3-hypergraph and G be the underlying graph of H (obtained by removing all arity-3 hyperedges). Let $I(G,k)$ be the set of all k-independent sets in G, and let $I_{\text{invalid}}(H,k)$ be the set of all independent sets in G that contain a hyperedge in H. Then H contains an independent set of size k if and only if*

$$|I(G,k)| - |I_{\text{invalid}}(H,k)| > 0.$$

Recall that the classical clique counting approach computes $|I(G,k)|$ in time $\mathcal{O}(n^{k\omega/3})$. It remains to argue that we can also compute $|I_{\text{invalid}}(H,k)|$ efficiently. The following lemma gives us a way to compute $|I_{\text{invalid}}(H,k)|$ via the standard inclusion-exclusion-based approach. In particular, for a subset of hyperedges S, we can count how many independent sets in G contain S and then make sure we avoid double-counting. For any subset of hyperedges S, let $I_S(H,k)$ denote the set of all independent sets of size k in G that contain all vertices spanned by S.

▶ **Lemma 17** (Invalid Solutions via the Inclusion-Exclusion Principle)**.** *Let H be a 3-hypergraph and let E_3 denote the set of all arity-3 hyperedges in H. Then the following equality holds.*

$$|I_{\text{invalid}}(H,k)| = \sum_{i=1}^{\binom{k}{3}} \sum_{S \in \binom{E_3}{i}} (-1)^{i+1} |I_S(H,k)|.$$

Proof. The statement follows directly from the inclusion-exclusion principle, by observing that $I_{\text{invalid}}(H,k) = \bigcup_{e \in E_3} I_{\{e\}}(H,k)$, and that for any non-empty set S we have $I_S(H,k) = \bigcap_{e \in S} I_{\{e\}}(H,k)$. ◀

It is easy to see that naively enumerating all sets I_S and computing the term $|I_S(H,k)|$ is infeasible. However, we can observe that a lot of these sets can be seen as redundant by a more clever implementation. For instance, the only way that a set S that contains $\binom{k}{3}$ edges is a part of an independent set of size k in G is that the k vertices that it spans form a hyperclique in the underlying 3-uniform hypergraph of H. However, in that case, we can observe that there is a subset S' of S that is of a much smaller size, in particular contains only $\frac{k}{3}$ hyperedges, that spans the same vertex set as S. Intuitively, if a particular solution contains many clustered hyperedges, by a naive implementation of our inclusion-exclusion approach, we will enumerate many edge sets S that span the same set of vertices, hinting at the fact that we are doing a lot of redundant work. In particular, this means that we only need to look at the sets S that contain up to $\frac{k}{3}$ many edges, as long as we can guarantee that the independent sets spanned by these clustered hyperedges are never double counted. We dedicate the rest of this section to formally introducing the notion of the clustered edges and showing how to avoid this redundant work.

Handling intersecting edges

Let us first introduce some useful notation and terminology. We will introduce all of the concepts in a more general setting, namely for h-hypergraphs for $h \geq 3$, in order to be able to reuse it in the following sections, but note that in this section we will only focus on the special case of $h = 3$. Denote by E_i the set of all hyperedges of arity i, and analogously, denote by $E_{\geq i}$ the set of all hyperedges of arity at least i. Let $\phi : E(H) \to [m]$ be any fixed bijection. We use ϕ to obtain a total ordering of the hyperedges of H. That is, we write $e \prec e'$ if $\phi(e) < \phi(e')$. Let e, e' be edges that intersect (share common vertices) and assume that $e \prec e'$. Intuitively, our goal is to efficiently remove all independent sets X that contain e from $I_S(H, k)$ for any S that contains e' to avoid the double counting of the sets spanned by the overlapping edges. We now formally prove that this does not destroy any valid solutions. For any set of hyperedges S, let $I'_S(H, k)$ be the set of all independent sets X of the underlying graph G of size k satisfying the following two conditions.

1. X contains all the vertices spanned by the hyperedges in S.
2. *(Handling the overlapping hyperedges)* For any edge $e \in S$ and any edge $e' \in E_{\geq 3}$, such that $e \cap e' \neq \emptyset$, if $e' \prec e$, then X does not contain e'.

We now prove that we can safely replace the sets I_S in our inclusion-exclusion formula by the sets I'_S.

▶ **Lemma 18.** *Let H be an h-hypergraph and let $E_{\geq 3}$ denote the set of all arity-≥ 3 hyperedges in H. Then the following equality holds.*

$$I_{\text{invalid}}(H, k) = \bigcup_{e \in E_{\geq 3}} I'_{\{e\}}(H, k).$$

Proof. We prove this by demonstrating set containment in both sides. Notice that one side is straightforward, as for each e we have $I'_{\{e\}} \subseteq I_{\{e\}}$, and hence:

$$\bigcup_{e \in E_{\geq 3}} I'_{\{e\}}(H, k) \subseteq \bigcup_{e \in E_{\geq 3}} I_{\{e\}}(H, k) = I_{\text{invalid}}(H, k),$$

where the last equality follows directly from definition of the set $I_{\text{invalid}}(H, k)$. To show the other containment, let X be any set in $I_{\text{invalid}}(H, k)$. It suffices to show that there exists some e such that X is contained in $I'_{\{e\}}$. By definition, the subhypergraph $H[X]$ contains at least one hyperedge. Let e be the first hyperedge contained in $H[X]$ with respect to the ordering $\prec$, i.e. for any hyperedge e' contained in X, we have $e \prec e'$. Then clearly X is contained in $I'_{\{e\}}$. ◀

The idea of replacing I by I' in our inclusion-exclusion formula is to avoid double counting the clustered independent sets, hence we should expect that many higher order terms will vanish. Indeed, we now prove that this is indeed the case.

▶ **Lemma 19.** *Let H be an h-hypergraph and let $E_{\geq 3}$ denote the set of all arity-≥ 3 hyperedges in H. Let $S \subseteq E_{\geq 3}$ be any set of at least $\frac{k}{3} + 1$ hyperedges. Then $I'_S(H, k) = \emptyset$.*

Proof. First notice that if for every pair of hyperedges $e, e' \in S$ it holds that $e \cap e' = \emptyset$, then S spans at least $3(\frac{k}{3} + 1) > k$ many vertices, and vacuously cannot be contained in any independent set of size k in the underlying graph, hence $I'_S(H, k) \subseteq I_S(H, k) = \emptyset$. Now assume that S contains two edges e, e' that share at least one common vertex. Then each independent set in $I_S(H, k)$ violates Condition 2 (since either $e \prec e'$, or vice versa, in either case any independent set that contains both e, e' violates Condition 2) in the definition of $I'_S(H, k)$ and hence is not contained in $I'_S(H, k)$, implying that I'_S is empty. ◀

We can now formally rewrite our inclusion-exclusion formula in terms of sets I'_S.

▶ **Corollary 20.** *Let H be an h-hypergraph and let $E_{\geq 3}$ denote the set of all arity-3 hyperedges in H. Then the following equality holds.*

$$|I_{\text{invalid}}(H,k)| = \sum_{i=1}^{\lceil \frac{k}{3} \rceil} \sum_{S \in \binom{E_{\geq 3}}{i}} (-1)^{i+1} |I'_S(H,k)|.$$

We now proceed to argue that we can construct the sets I'_S efficiently.

Constructing the sets I'_S

Given two hyperedges e, e' such that $e \cap e' \neq \emptyset$, we distinguish between two types of intersections between them. We say that the intersection of two hyperedges e, e' is *type i* if (1) $e' \prec e$, and (2) $|e' \setminus e| = i$. The idea to construct the sets I'_S is to recursively guess an arity-3 edge e and then for all edges e' in $E_3(H)$ such that the intersection e, e' is type i, we span the vertex set $(e' \setminus e)$ by a hyperedge of arity i in all the descending branches. This ensures that we never consider the independent sets that violate Condition 2 in the definition of the set I'_S. More formally, we consider the following algorithm.

Algorithm 1 Subroutine that resolves intersecting hyperedges via replacement or removal. Given a hypergraph H and a subset of edges S, returns a new hypergraph H' such that $I'_S(H,k) = I_S(H',k)$.

```
1:  procedure RESOLVE-INTERSECTIONS(H, S ⊂ E(H))
2:      for e ∈ S do
3:          H' ← H
4:          for e' ∈ E_3(H) do
5:              if |e ∩ e'| = 0 or e ≺ e' then
6:                  continue
7:              else if |e' \ e| = 1 then
8:                  v ← the unique vertex in set (e' \ e)
9:                  H' ← H' − v
10:             else
11:                 e'' ← hyperedge spanning (e' \ e)
12:                 H' ← (H' − e') ∪ e''
        return H'
```

▶ **Lemma 21.** *Let H be a 3-hypergraph and let $S \subset E_3(H)$. Let H' be the hypergraph returned by the* RESOLVE-INTERSECTIONS(H,S) *function in Algorithm 1. Then $I'_S(H,k) = I_S(H',k)$.*

Proof sketch. Let $X \in I_S(H',k)$. We show that X satisfies Condition 2 in the definition of $I'_S(H,k)$. Consider any edge $e' \in E_3(H)$ such that for some $e \in S$, we have $e' \prec e$ and $e' \cap e \neq \emptyset$. If $|e' \setminus e| = 1$, then the procedure removes the unique vertex in $e' \setminus e$ from H', so e' cannot be contained in X. If $|e' \setminus e| = 2$, the procedure inserts an arity-2 edge between the two vertices of $e' \setminus e$, and thus no independent set in the underlying graph of H' can contain e'. The remaining cases are impossible: $|e' \setminus e| = 0$ implies $e' = e$, while $|e' \setminus e| \geq 3$ implies $e' \cap e = \emptyset$. Hence $X \in I'_S(H,k)$.

Conversely, let $X \in I'_S(H,k)$. Suppose that $X \notin I_S(H',k)$. Since X is independent in the underlying graph of H, this can only happen if $H[X]$ contains some edge e' such that $e' \prec e$ and $e' \cap e \neq \emptyset$ for some $e \in S$. But then X violates Condition 2, contradicting $X \in I'_S(H,k)$. ◀

We can now use Algorithm 1 as a subroutine in our main algorithm. Recall that in a hypergraph H, a *matching* M of size i is a set of i pairwise non-intersecting hyperedges. Also, for simplicity, assume that k is divisible by 3, and we will handle the remaining cases later in the analysis.

Algorithm 2 Algorithm for counting size-k sets that are independent with respect to arity-2 edges but contain at least one arity-3 hyperedge.

```
1:  procedure Invalid-Solutions(H, k)
2:      count ← 0
3:      G ← Underlying graph of H
4:      for i ∈ [k/3] do
5:          sign ← (−1)^{i+1}
6:          for matching S ⊂ E_{≥3} of size i do
7:              if |V(S)| = k then
8:                  if S ∈ I(G, k) then
9:                      count++
10:                 continue
11:             H′ ← resolve-intersections(H, S)
12:             count ← count + sign · |I_S(H′, k)|
        return count
```

It remains to prove the correctness of this algorithm and analyse the running time.

▶ **Lemma 22.** *Given a 3-Hypergraph H with n vertices and m hyperedges, Algorithm 2 returns the value $|I_{\text{invalid}}(H,k)|$ correctly in time $\mathcal{O}\left(n^{\frac{k\omega}{3}} + m^{\frac{k}{3}}\right)$.*

Proof. Correctness follows directly from Lemma 21 and Corollary 20. We now prove the upper bound on the running time. Recall that there are at most $\mathcal{O}(m^i)$ matchings of size i, and that each such matching S by definition spans precisely $3i$ many vertices in 3-hypergraphs. In particular, the set $I_S(H', k)$ consists of $3i$ many "guessed" matching vertices and $k - 3i$ many vertices outside S. If $3i < k$, we can find the remaining $k - 3i$ many vertices by running the standard matrix-multiplication algorithm on the underlying graph $G' - N_{G'}[S]$. Constructing H' (and consequently the underlying graph G') takes at most $\mathcal{O}(m)$ time. Note that if $3i = k$, we only need to check that S forms an independent set in the underlying graph G, which we can do in $\mathcal{O}(1)$ time (Line 8). This allows us to bound the total running time as follows.

$$T_k(n,m) = \mathcal{O}\left(m^{\frac{k}{3}} + \sum_{i=0}^{\frac{k}{3}-1} m^i \left(m + n^{(k-3i)\omega/3}\right)\right) = \mathcal{O}\left(m^{\frac{k}{3}} + \sum_{i=0}^{\frac{k}{3}-1} m^{i+1} + m^i n^{(k-3i)\omega/3}\right)$$
$$= \mathcal{O}\left(m^{\frac{k}{3}} + \sum_{i=0}^{\frac{k}{3}-1} m^i n^{(k-3i)\omega/3}\right) = \mathcal{O}\left(\sum_{i=0}^{\frac{k}{3}} m^i n^{(k-3i)\omega/3}\right) = \mathcal{O}\left(m^{\frac{k}{3}} + n^{k\omega/3}\right). \quad \blacktriangleleft$$

This Lemma, together with Observation 16 concludes the proof of Theorem 14. We prove the conditional optimality (Theorem 15) of this algorithm in the full version of the paper.

4 Binary Constraint Families and The Effects Of Sparsity

Due to space constraints, we dedicate this section to proving the upper bound for solving constraint families of the form $\mathcal{F} \cup \{\mathbf{NAND}_2\}$ in time $\mathcal{O}(n^{\gamma(k-s_{\min}(\mathcal{F}))\omega/6+1})$, i.e. the positive part of Regime 4 of Theorem 10. Here $s_{\min}(f)$ denotes the minimum number of ones required to satisfy a constraint function f. Formally we let $s_{\min}(f) \coloneqq \min\{\|x\|_1 \mid f(x) = 1\}$ and $s_{\min}(\mathcal{F}) \coloneqq \min_{f\in\mathcal{F}} s_{\min}(f)$.

▶ **Theorem 23.** *Let $\mathcal{F}$ be a non-empty binary constraint family. Then we can solve $\#\mathrm{CSP}^{\gamma}_k(\mathcal{F} \dot\cup \{\boldsymbol{NAND}_2\})$ in time $\mathcal{O}(m^{\frac{\omega(k-s_{\min}(\mathcal{F})}{6}+1})$.*

Preliminaries. Let us recall and introduce some #CSP specific notations and results. For a singleton constraint family $\mathcal{F} = \{f\}$, we also write $\#\mathrm{CSP}^{\gamma}_k(f)$. Denote by $\mathrm{vars}(\Phi)$ the variables of an instance Φ of $\#\mathrm{CSP}^{\gamma}_k(\mathcal{F})$, then we define its *primal graph* $G(V, E)$ where $V = \{x_i \mid x_i \in \mathrm{vars}(\Phi)\}$ and $E = \{(x_{i_1}, \ldots, x_{i_h}) \mid f(x_1, \ldots, x_{i_h}) \in \Phi\}$. Furthermore, we call a satisfying assignment a for $\#\mathrm{CSP}^{\gamma}_k$ a *solution* if it is of weight exactly k. For a Boolean constraint function f, we use $f(\mathbf{x})$ with $\mathbf{x} = (x_1, \ldots, x_h)$ as a shorthand for $f(x_1, \ldots, x_h)$.

A constraint function f is 0-*valid/invalid* if it is satisfied/violated by the all 0-assignment. Moreover, we say a constraint family is 0-valid if this holds for all its constraints and 0-invalid if it contains at least one 0-invalid constraint.

We generally assume our constraint families to be finite not containing trivial constraint functions, as they can be easily resolved. In the following, we will refer to Boolean constraint functions as known from propositional logic, i.e. $F(x) = \neg x$, $\mathbf{NAND}_2(x, y) = \neg(x \wedge y)$, $\mathbf{IMPL}(x, y) = x \to y$, $\mathbf{EQ}(x, y) = x \leftrightarrow y$ and so on.

We will only give the algorithm for the family of $\{\mathbf{NAND}_2, \mathbf{IMPL}\}$, as we can reduce all other binary constraint families to this case. For full details on the remaining cases, refer to the full version of the paper.

▶ **Theorem 24.** *There is an algorithm solving $\#\mathrm{CSP}^{\gamma}_k(\{\boldsymbol{NAND}_2, \boldsymbol{IMPL}\})$ in time $\mathcal{O}\left(m^{k\omega/6+1}\right)$.*

In the following, we will consider the primal graph G of the given instance, distinguishing between $\mathbf{NAND}_2$ and **IMPL** edges in G. With regard to the **IMPL**-edges, we define for a vertex v its descendants $D(v) = \{u \mid u \in V, \text{s.t. } u \text{ is reachable from } v\}$ and ancestors $A(v) = \{u \mid u \in V, \text{s.t. } v \text{ is reachable from } u\}$. In particular, we have that $v \in D(v)$ and $v \in A(v)$. We also extend this notation to sets of vertices s.t. $D(V) = \bigcup_{v\in V} D(v)$. We compute for every vertex $D(v)$ and $A(v)$. During the computation, if we find two vertices $u, w \in D(v)$ such that there exists a $\mathbf{NAND}_2$-edge on u, w, we can safely remove v and all its ancestors from the graph. Further, we can remove any vertex v such that $D(v) \geq k$.

After this preprocessing step, we observe the following: due to the nature of **IMPL** constraints, vertices that have many descendants are *cheap* to guess: Consider the vertices $V_{\geq 3} = \{|D(v)| \geq 3 \mid v \in V\}$. Setting some $v \in V_{\leq 3}$ to 1 yields at least 2 more variables $u \in D(v)$, which we need to set to 1. Find the complete proof in the full version of the paper.

Hence, let us only consider the hard case where for all $v \in V$ it holds that $|D(v)| \leq 2$. We call instances that satisfy this condition *restricted*.

▶ **Theorem 25** (Restricted $\#\mathrm{CSP}^{\gamma}_k(\{\mathbf{NAND}_2, \mathbf{IMPL}\})$). *Let Φ be an instance of $\#\mathrm{CSP}^{\gamma}_k(\{\boldsymbol{NAND}_2, \boldsymbol{IMPL}\})$ such that no two variables share both an **IMPL**- and **NAND**$_2$-constraint and $|D(v)| \leq 2$ for all $v \in \mathit{vars}(\Phi)$. Then there is an algorithm solving Φ in time $\mathcal{O}\left(m^{k\omega/6+1}\right)$.*

Proof. We devise this algorithm in 3 steps. First, we exploit the restriction on **IMPL**-constraints to partition the variables into groups. Each group induces a solution within itself, if it is sufficiently large and sparse w.r.t. $\mathbf{NAND}_2$-edges. Otherwise, we can assume that such a group must be dense which allows us to bound its size. Then,by grouping vertices cleverly (depending on **IMPL**-edges), we reduce to triangle detection.

The following observation regarding $\mathbf{NAND}_2$ sparsity of large vertex sets is crucial for our algorithm:

▷ Claim 26. Let $V' \subseteq V$ with $|V'| \geq \sqrt{m}^{1+\epsilon}$ for all $\epsilon > 0$, then there exists an independent set $\{v_1, \ldots, v_k\}$ w.r.t. $\mathbf{NAND}_2$-edges.

Proof. Assume V' were dense, i.e. the count of edges is

$$\mathcal{O}\left(|V'|^2\right) = \mathcal{O}\left(\left(\sqrt{m}^{1+\epsilon}\right)^2\right) = \mathcal{O}\left(m^{1+\epsilon}\right) = \mathcal{O}\left(n^{(\gamma+\gamma\epsilon)}\right)$$

where $\gamma\epsilon > 0$, which is a contradiction as the graph has only $\mathcal{O}(n^\gamma)$ edges in total. Thus we can find a k-independent set in V' in linear time. ◁

In particular, this bounds the size of any $V' \subseteq V$ where **IMPL** edges allow us to (almost) freely choose vertices from V' by $|V'| \geq \sqrt{m}^{1+\epsilon}$. Otherwise we can find a k-independent set directly in time $\mathcal{O}(n+m)$.

We note that we can efficiently remove all 2-cycle w.r.t. **IMPL** edges, as their number can be bounded using Claim 26.

Reducing to Triangle Detection. We partition V into sets

- $V_L = \{v \mid |A(v)| = 1 \text{ and } |D(v)| = 2\}$,
- $V_R = \{v \mid |A(v)| \geq 2\}$ and
- $V_0 = \{v \mid |D(v)| = |A(v)| = 1\}$.

For any vertex $u \in V_L$ to be part of a solution, there exists some $v \in V_R$ that is necessarily contained as well. In this regard, we call the set of ancestors $A(v)$ of $v \in V_R$ and v itself its *group*. Conversely, if we know that some $v \in V_R$ is part of a solution, then including any subset of its group is consistent with the **IMPL** constraints. Vertices in V_0 are isolated w.r.t. **IMPL**-edges.

Using Claim 26, we can bound the size of a group. If a group is too large, there is solution fully contained therein. We have the following properties for groups:

P1 $|V_R| \leq f(k)\sqrt{m}$, thus there are at most $\mathcal{O}(\sqrt{m})$ many groups.

P2 Each group has size at most $|A(v)| \leq f(k)\sqrt{m}$

To see this, observe that there are no **IMPL**-edges between vertices in V_L, and no **IMPL**-edges contained in V_R, as the restricted instance promises that $|D(V)| \leq 2$ for all $v \in V$. Lastly, we also consider V_0 as a group as well, where we can bound its size $|V_0| \leq f(k)\sqrt{m}$ by Claim 26.

We exploit these properties by guessing *how many groups* (at most $\mathcal{O}(\sqrt{m})$) a solution is composed of. Further, we guess how many vertices k_i of each group (containing at most $\mathcal{O}(\sqrt{m})$ vertices) are part of the solution, thus it suffices to consider at most $\mathcal{O}(\sqrt{m}^{k_i})$ subsets per group. To this end, the algorithm guesses a partition of k into ℓ groups of sizes $k = k_1 + \cdots + k_\ell$ with $k_1 \leq \cdots \leq k_\ell$, to eventually reduce to an instance of triangle detection. First, consider the case of at most two groups.

$\ell \leq 2$. We guess at most two groups in time $\mathcal{O}(\sqrt{m}^2) = \mathcal{O}(m)$. It remains to find a $(k-2)$-independent set among the $f(k)\sqrt{m}$ many vertices in the groups $A(v_1) \cup A(v_2)$. We distribute these $\mathcal{O}(\sqrt{m})$ vertices into three parts, where each part corresponds to a choice of $\lceil \frac{k-2}{3} \rceil \leq \frac{k}{3}$ many vertices and reduce to a 3-partite triangle instance of size $\mathcal{O}\left(\sqrt{m}^{\frac{k}{3}}\right)$. Two vertices in the triangle instance are adjacent, if the corresponding sets of vertices in the original instance form an independent set. The total time of guessing the groups, constructing and solving the triangle instance is bounded by $\mathcal{O}\left(m^{k\omega/6+1}\right)$.

$\ell \geq 3$. We first prove the following claim, stating that we can distribute the groups corresponding to $k_1, \ldots, k_{\ell-2}$ in three bins in such a way that the *imbalance* of any pair of bins is at most $k_{\ell-1}$; then we can use the remaining two groups to fix the imbalance:

▷ Claim 27. Let $k_1 \leq \cdots \leq k_\ell \in \mathbb{N}$. Then there exists a distribution of $k_1, \ldots, k_{\ell-2}$ into three bins S_1, S_2, S_3 such that for all pairs $p, q \in [3]$ it holds that

$$\left|\left|\left(\sum_{k_i \in S_p} k_i\right)\right| - \left|\left(\sum_{k_j \in S_q} k_j\right)\right|\right| \leq k_{\ell-1}.$$

We omit the proof due to spaces reasons, but this can be verified by a simple greedy argument.

Let s_1, s_2, s_3 be the sums of values from the bins S_1, S_2, S_3 respectively. We construct the triangle instance according to the claim. This yields an instance of triangle detection X_1, X_2, X_3 where each set X_i corresponds to all valid choices of s_i vertices that adhere to our group assignment choice. Again, two vertices are adjacent if their vertex choices form a valid independent set.

To fix a potentially large imbalance, we use the remaining $k_{\ell-1}+k_\ell$ vertices to balance these partitions. We guess the remaining two groups in $\mathcal{O}(m)$ (i.e. assume that the corresponding vertices in V_R are part of the solution) time and distribute the vertices of their group, where exactly $k_{\ell-1} + k_\ell - 2$ many of those vertices have to be part of a solution. The new triangle instance will have each node correspond to a subset of $\binom{V}{s_i+r_i}$ vertices of the original instance, where the s_i vertices are obtained by taking the union of groups $1, \ldots, \ell-2$ as in the claim, and the r_i vertices come from the remaining two groups, distributed to minimize the imbalance. Indeed, we can reduce the imbalance between buckets to $\max_{i,j\in[3]} |s_i + r_i - s_j - r_j| \leq 1$. See that the imbalance after distributing $k_1, \ldots, k_{\ell-2}$ is at most $k_{\ell-2}$. Thus, we distribute the contribution $k_{\ell-1} - 1$ and $k_\ell - 1$ of our guessed groups to the imbalanced buckets. As we have at least $2(k_{\ell-2})$ many vertices available and the imbalance is at most $k_{\ell-2}$, we can distribute the contributions up to an imbalance of 1. We obtain buckets of size at most $s_k \leq \lceil \frac{k-2}{3} \rceil \leq \frac{k}{3}$ and the triangle instance is of size $\mathcal{O}(\sqrt{m}^{\frac{k}{3}})$. The total time thus is bounded $\mathcal{O}\left(m^{k\omega/6+1}\right)$. Performing this for every partition of k, we arrive at a total runtime of $\mathcal{O}\left(m^{k\omega/6+1}\right)$ ◀

References

1 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.

2 Amir Abboud, Karl Bringmann, Holger Dell, and Jesper Nederlof. More consequences of falsifying SETH and the orthogonal vectors conjecture. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 253–266. ACM, 2018. doi:10.1145/3188745.3188938.

3 Amir Abboud, Nick Fischer, and Yarin Shechter. Faster combinatorial k-clique algorithms. In José A. Soto and Andreas Wiese, editors, *LATIN 2024: Theoretical Informatics - 16th Latin American Symposium, Puerto Varas, Chile, March 18-22, 2024, Proceedings, Part I*, volume 14578 of *Lecture Notes in Computer Science*, pages 193–206. Springer, 2024. doi:10.1007/978-3-031-55598-5_13.

4 Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 377–391. SIAM, 2016. doi:10.1137/1.9781611974331.CH28.

5 Udit Agarwal and Vijaya Ramachandran. Fine-grained complexity for sparse graphs. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 239–252. ACM, 2018. doi:10.1145/3188745.3188888.

6 József Balogh, Robert Morris, and Wojciech Samotij. The method of hypergraph containers. In *Proceedings of the International Congress of Mathematicians (ICM 2018)*, pages 3059–3092, 2018. doi:10.1142/9789813272880_0172.

7 Karl Bringmann, Nick Fischer, and Marvin Künnemann. A fine-grained analogue of Schaefer's theorem in P: dichotomy of existsˆk-forall-quantified first-order graph properties. In Amir Shpilka, editor, *34th Computational Complexity Conference, CCC 2019, New Brunswick, NJ, USA, July 18-20, 2019*, volume 137 of *LIPIcs*, pages 31:1–31:27. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.CCC.2019.31.

8 Karl Bringmann and Jasper Slusallek. Current algorithms for detecting subgraphs of bounded treewidth are probably optimal. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, Glasgow, Scotland (Virtual Conference), July 12-16, 2021*, volume 198 of *LIPIcs*, pages 40:1–40:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.40.

9 Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. *J. ACM*, 60(5):34:1–34:41, 2013. doi:10.1145/2528400.

10 Andrei A. Bulatov. A dichotomy theorem for nonuniform csps. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 319–330. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.37.

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

12 Yi-Jun Chang. Hardness of RNA Folding Problem With Four Symbols. In Roberto Grossi and Moshe Lewenstein, editors, *27th Annual Symposium on Combinatorial Pattern Matching (CPM 2016)*, volume 54 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:12, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CPM.2016.13.

13 Mina Dalirrooyfard, Ce Jin, Virginia Vassilevska Williams, and Nicole Wein. Approximation algorithms and hardness for n-pairs shortest paths and all-nodes shortest cycles. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 290–300. IEEE, 2022. doi:10.1109/FOCS54457.2022.00034.

14 Mina Dalirrooyfard and Virginia Vassilevska Williams. Induced cycles and paths are harder than you think. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 531–542. IEEE, 2022. doi:10.1109/FOCS54457.2022.00057.

15 Nick Fischer, Marvin Künnemann, and Mirza Redzic. The effect of sparsity on k-dominating set and related first-order graph properties. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4704–4727. SIAM, 2024. doi:10.1137/1.9781611977912.168.

16 Nick Fischer, Marvin Künnemann, Mirza Redzic, and Julian Stieß. The role of regularity in (hyper-)clique detection and implications for optimizing boolean csps. In Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis, editors, *52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025, Aarhus, Denmark, July 8-11, 2025*, volume 334 of *LIPIcs*, pages 78:1–78:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ICALP.2025.78.

17 Cheng-Hao Fu, Andrea Lincoln, and Rene Reyes. Worst-case and average-case hardness of hypercycle and database problems. In Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis, editors, *52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025, Aarhus, Denmark, July 8-11, 2025*, volume 334 of *LIPIcs*, pages 81:1–81:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ICALP.2025.81.

18 Peter Keevash. Hypergraph turan problems. *Surveys in combinatorics*, 392:83–140, 2011.

19 Sanjeev Khanna, Madhu Sudan, and David P. Williamson. A complete classification of the approximability of maximization problems derived from boolean constraint satisfaction. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 11–20. ACM, 1997. doi:10.1145/258533.258538.

20 Marvin Künnemann and Dániel Marx. Finding small satisfying assignments faster than brute force: A fine-grained perspective into boolean constraint satisfaction. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 27:1–27:28. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.27.

21 Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1236–1252. SIAM, 2018. doi:10.1137/1.9781611975031.80.

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

23 Surya Mathialagan, Virginia Vassilevska Williams, and Yinzhan Xu. Listing, verifying and counting lowest common ancestors in dags: Algorithms and fine-grained lower bounds. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, Paris, France, July 4-8, 2022*, volume 229 of *LIPIcs*, pages 94:1–94:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.94.

24 Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 026(2):415–419, 1985. URL: http://eudml.org/doc/17394.

25 Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 603–610. ACM, 2010. doi:10.1145/1806689.1806772.

26 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 515–524. ACM, 2013. doi:10.1145/2488608.2488673.

27 Thomas J. Schaefer. The complexity of satisfiability problems. In Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, editors, *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 216–226. ACM, 1978. doi:10.1145/800133.804350.

28 P Turán. On an extremal problem in graph theory. mat. fiz. lapok 48 436–452, 1941.

29 Virginia Vassilevska Williams, Joshua R. Wang, Richard Ryan Williams, and Huacheng Yu. Finding four-node subgraphs in triangle time. In Piotr Indyk, editor, *Proc. Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015)*, pages 1671–1680. SIAM, 2015. doi:10.1137/1.9781611973730.111.

30 Virginia Vassilevska Williams and Yinzhan Xu. Monochromatic triangles, triangle listing and APSP. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 786–797. IEEE, 2020. doi:10.1109/FOCS46700.2020.00078.

31 R. Ryan Williams. *Algorithms and resource requirements for fundamental problems*. PhD thesis, Carnegie Mellon University, USA, 2007. AAI3274191.

32 Or Zamir. Algorithmic applications of hypergraph and partition containers. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 985–998. ACM, 2023. doi:10.1145/3564246.3585163.

33 Dmitriy Zhuk. A proof of CSP dichotomy conjecture. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 331–342. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.38.