A Fine-Grained Analogue of Schaefer's Theorem in P: Dichotomy of $\exists^k\forall$-Quantified First-Order Graph Properties

Karl Bringmann
Max Planck Institute for Informatics, Saarland Informatics Campus (SIC), Saarbrücken, Germany

Nick Fischer
Max Planck Institute for Informatics, Saarbrücken Graduate School of Computer Science, Saarbrücken, Germany

Marvin Künnemann
Max Planck Institute for Informatics, Saarland Informatics Campus (SIC), Saarbrücken, Germany

Abstract

An important class of problems in logics and database theory is given by fixing a first-order property ψ over a relational structure, and considering the model-checking problem for ψ. Recently, Gao, Impagliazzo, Kolokolova, and Williams (SODA 2017) identified this class as fundamental for the theory of fine-grained complexity in P, by showing that the *(Sparse) Orthogonal Vectors* problem is complete for this class under fine-grained reductions. This raises the question whether fine-grained complexity can yield a precise understanding of all first-order model-checking problems. Specifically, can we determine, for any fixed first-order property ψ, the exponent of the optimal running time $O(m^{c_\psi})$, where m denotes the number of tuples in the relational structure?

Towards answering this question, in this work we give a dichotomy for the class of $\exists^k\forall$-quantified graph properties. For every such property ψ, we either give a polynomial-time improvement over the baseline $O(m^k)$-time algorithm or show that it requires time $m^{k-o(1)}$ under the hypothesis that MAX-3-SAT has no $O((2-\varepsilon)^n)$-time algorithm. More precisely, we define a hardness parameter $h = H(\psi)$ such that ψ can be decided in time $O(m^{k-\varepsilon})$ if $h \leq 2$ and requires time $m^{k-o(1)}$ for $h \geq 3$ unless the h-uniform HyperClique hypothesis fails. This unveils a natural hardness hierarchy within first-order properties: for any $h \geq 3$, we show that there exists a $\exists^k\forall$-quantified graph property ψ with hardness $H(\psi) = h$ that is solvable in time $O(m^{k-\varepsilon})$ *if and only if* the h-uniform HyperClique hypothesis fails. Finally, we give more precise upper and lower bounds for an exemplary class of formulas with $k = 3$ and extend our classification to a counting dichotomy.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Fine-grained Complexity, Hardness in P, Hyperclique Conjecture, Constrained Triangle Detection

Digital Object Identifier 10.4230/LIPIcs.CCC.2019.31

1 Introduction

One of the most expressive problems in complexity theory is the model-checking problem for first-order definable properties over relational structures. Any such property can be written as a formula of the form

$$(Q_1 x_1)\ \ldots\ (Q_k x_k)\, \phi(x_1, \ldots, x_k),$$

where $Q_i \in \{\exists, \forall\}$, ϕ is an arbitrary Boolean formula defined over an arbitrary set of predicates and the relational structure is given by explicitly listing all tuples defining the predicates. This problem encompasses, e.g., the hugely diverse set of constraint satisfaction problems and

34th Computational Complexity Conference (CCC 2019).
Editor: Amir Shpilka; Article No. 31; pp. 31:1–31:27

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

it is fundamental for database theory as *query evaluation* problem over relational structures. In this context, the relational structure is a relational database and the first-order definable property corresponds to queries to the database, see, e.g., [9] for an overview.

Given its expressiveness, it is no surprise that the complexity of this problem has been extensively studied under various angles: Among others, one distinguishes between the combined complexity (where both the first-order property and the relational structure are part of the input) and the data complexity (where the first-order property is fixed and the input only contains the relational structure) [54]. After classical works covered various aspects of these complexities (see, e.g., [31, 10, 30, 44]), later research turned towards parameterized analyses of such problems, see, e.g., [55, 37, 50] and the overview in [40, Section 4.3]. In this work, we pursue an even finer-grained complexity analysis of the data complexity of bounded-variable formulas, which capture a rich complexity landscape of low-degree polynomial-time problems [58, 41].

Consider the following examples for first-order properties, where the relational structure consists of the binary *edge relation* $E(x, x')$ over vertices in V, defining an (undirected) graph G:

- Triangle: $(\exists x_1 \in V)\,(\exists x_2 \in V)\,(\exists x_3 \in V)\,(E(x_1, x_2) \wedge E(x_2, x_3) \wedge E(x_3, x_1))$
  (determine whether G contains a triangle)
- TwinFree: $(\forall x_1 \in V)\,(\forall x_2 \in V)\,(\exists x_3 \in V)\,(\overline{E(x_1, x_2)} \vee (E(x_1, x_3) \not\leftrightarrow E(x_2, x_3)))$
  (determine whether G contains no adjacent vertices x_1, x_2 sharing the same neighborhood)

Such properties are called *graph properties*. Another example for a graph property (although we do not call the binary relation "edges" in this case) is a version of the Hitting Set problem, in which we are given an explicitly represented set family $\mathcal{S} = \{S_1, \ldots, S_n\}$ over some universe U. The explicit representation here is a list of tuples fulfilling the relation $s \in S_i, i \in \{1, \ldots, n\}$:

- HittingSet: $(\exists H \in \mathcal{S})\,(\forall S \in \mathcal{S})\,(\exists u \in U)\,(u \in H \wedge u \in S)$
  (determine whether there is some input set H that hits all (other) input sets)

A simple baseline algorithm solves the model-checking problem for any first-order formula in prenex normal form with $(k+1)$ quantifiers in time $O(m^k)$, where m denotes the size of the relational structure (i.e., the number of tuples satisfying the relations). This has recently been improved to $m^k/2^{\Omega(\sqrt{\log m})}$ [41]. However, we can surpass this bound significantly for specific formulas. In particular, TwinFree has a simple $O(m)$-time algorithm [42] and Triangle can be solved in time $O(m^{\frac{2\omega}{\omega+1}}) = O(m^{1.41})$ [12] (where $\omega \leq 2.373$ denotes the matrix multiplication exponent), while for HittingSet we do not know of any faster algorithm than the $m^2/2^{\Omega(\sqrt{\log m})}$-solution of Gao et al. [41] (in fact, this barrier has been used as a hardness assumption in its own right [6]). This raises the question: How can we determine the constant $c_\psi > 0$ in the optimal running time $m^{c_\psi \pm o(1)}$ for specific formulas ψ? In particular, when is a close-to-baseline time $m^{k \pm o(1)}$ the best possible?

1.1 Complete Problem: (Sparse) k-OV

Remarkable progress to this question has been made by Gao, Impagliazzo, Kolokolova, and Williams [41]. In the sense of admitting polynomial improvements over the $O(m^k)$-baseline, they identified the following problem as *complete for the class of model-checking problems for $(k+1)$-quantifier formulas*: (here, the input is a $(k+1)$-partite graph $G = (X_1 \uplus \cdots \uplus X_k \uplus Y, E)$)

- Sparse k-OV: $(\exists x_1 \in X_1) \ldots (\exists x_k \in X_k)\,(\forall y \in Y)\,(\overline{E(x_1, y)} \vee \ldots \vee \overline{E(x_k, y)})$

More precisely, Gao et al. show that Sparse k-OV has an $O(m^{k-\varepsilon})$-time algorithm for some constant $\varepsilon > 0$ if and only if for *all* $(k+1)$-quantifier formulas in prenex normal form, the model-checking problem can be solved in time $O(m^{k-\varepsilon'})$ for some constant $\varepsilon' > 0$. This identifies Sparse k-OV as one of the hardest problems in this class of problems.

Note that Sparse k-OV is a sparsely represented variant[1] of the following problem:

▶ **Problem 1** (k-OV)**.** Given k sets of 0-1 vectors $A_1, \ldots, A_k \subseteq \{0,1\}^d$, where $|A_i| = n$, determine whether there is a tuple $a_1 \in A_1, \ldots, a_k \in A_k$ such that $\prod_{i=1}^{k} a_i[\ell] = 0$ for all $\ell \in [d]$.

It is conjectured that for any constant k, k-OV cannot be solved in time $O(n^{k-\varepsilon}\operatorname{poly}(d))$ for any constant $\varepsilon > 0$, which is called the k-OV conjecture. Assuming this lower bound only for $k = 2$ is known simply as OV conjecture – this has been used as a hardness assumption for a number of conditional lower bounds in the quadratic-time regime [52, 18, 8, 13, 4, 20, 14, 6, 19, 21, 59, 2]. Likewise, the stronger k-OV conjecture has found uses to derive further polynomial-time lower bounds of different degree [51, 4, 1, 15, 48].

For almost a decade, the main support for the k-OV hypothesis was given by a reduction by Williams [57] using the so-called split-and-list technique to show that the Strong Exponential Time Hypothesis [45] implies the k-OV hypothesis. Only recently, Abboud et al. [5] show that the OV hypothesis is also implied by the weighted k-Clique hypothesis. Interestingly, the work by Gao et al. [41] gives additional evidence, as they show that existence of an $O(n^{k-\varepsilon}\operatorname{poly}(d))$-time algorithm for k-OV for constant $\varepsilon > 0$ is equivalent to the existence of an $O(m^{k-\varepsilon'})$-time model-checking algorithm for all $(k+1)$-quantifier formulas in prenex normal form for some constant $\varepsilon' > 0$. The study of first-order properties is thus tightly connected to the study of hardness and structure within P. In particular, our aim is to understand which properties make a first-order formula expressive enough to capture the hardest model-checking problems, i.e., Sparse k-OV, and which properties make a first-order formula easier to evaluate.

1.2 Classification à la Schaefer

Our questions ask for fine-grained analogues of classical results in computational complexity theory. Specifically, consider the case of Boolean constraint satisfaction problems (CSPs). As each Boolean CSP is in NP, Cook's theorem establishes that every CSP reduces to 3-SAT. Schaefer's Theorem [53] proves a dichotomy for reductions in the reverse direction: Assuming $\mathsf{P} \neq \mathsf{NP}$, every Boolean CSP either is polynomial-time solvable or requires superpolynomial time via a reduction from 3-SAT.

Translated to our setting, where first-order properties correspond to the class NP, Gao et al. give an analogue of Cook's Theorem: They give a fine-grained reduction from the model-checking problem of any $(k+1)$-quantifier first-order formula to Sparse k-OV. This raises the question:

Can we give a dichotomy analogous to Schaefer's classification, i.e., for each such formula either give an $O(m^{k-\varepsilon})$-time algorithm or show "Sparse k-OV-hardness"?

[1] To see the correspondence, let the vertex sets $X_i, i \in \{1, \ldots, k\}$ represent the vector sets A_i, let Y denote the dimensions $\{1, \ldots, d\}$ and let the binary relation $E(x_i, y)$ with $x_i \in X_i$ and $y \in Y$ hold if and only if $x_i[y] = 1$. We call this representation *sparse* as the relational structure only lists the 1-entries of the k-OV instance.

Our aim in this work is to initiate the investigation of such fine-grained classifications into *hardest* and *easier* first-order properties. Note that in the case of CSPs over finite domains, this line of research proved to be an effort requiring four decades of research in complexity, logic and algebra (see e.g., [34] for an early overview and the surveys [32, 25] for an introduction to the algebraic approach). In particular, after Schaefer's classification of the Boolean domain, Feder and Vardi raised the dichotomy conjecture that such a dichotomy also exists for CSPs over arbitrary finite domains [38, 39]. After a series of works developing an algebraic view on CSPs [47, 46, 26], and classifications of larger classes of CSPs (e.g., [43, 22, 23]), only very recently, Bulatov [24] and Zhuk [60] could finally resolve the conjecture.

Given the close connection to CSPs, we do not expect a full dichotomy for bounded-quantifier first-order properties to be within immediate reach of current techniques – thus, we focus on expressive fragments first. In particular, we focus on formulas with the quantifier structure $\exists^k\forall$, as it is the quantifier structure of the known complete problem Sparse k-OV (in fact, under a nondeterministic variant of SETH, $\exists^k\forall$ and the symmetric $\forall^k\exists$ are the only quantifier structure containing complete problems for first-order properties [28]). Note also that this quantifier structure is analogous to the quantifier structure for CSP solvability (existential quantifiers for variable assignments and a universal quantifier to check that all constraints are satisfied; this correspondence is best illustrated by Williams' split-and-list reduction from CNF satisfiability to OV [57]). We leave the remaining quantifier structures (analogous to the classification of quantified CSPs [53, 35, 34]) to be addressed in future work.

1.3 Further Related Work

Note that related work in database theory gives further flavors of fine-grained dichotomies for first-order properties: For the related setting of *query enumeration*, Bagan, Durand and Grandjean [16] classify each acyclic conjunctive query as either admitting constant-delay enumeration following linear-time precomputation or as hard under the assumption that Boolean matrix multiplication requires superquadratic time. This classification was recently extended to incorporate functional dependencies between attributes [27]. Further work gives fine-grained dichotomies under the OMv and OV hypotheses for dynamic databases [17].

1.4 Our Results

Our main result is a dichotomy for $\exists^k\forall$-quantified formulas over graphs under a plausible assumption about the complexity of MAX-3-SAT. Formally, $\exists^k\forall$-quantified first-order graph properties are formulas of the form

$$\psi = (\exists x_1 \in X_1) \ldots (\exists x_k \in X_k)\,(\forall y \in Y)\,\phi(x_1, \ldots, x_k, y), \tag{1}$$

where ϕ is an arbitrary Boolean formula defined over the atoms $E(v, v')$ with $v, v' \in \{x_1, \ldots, x_k, y\}$ and $v \neq v'$. Let $\mathrm{MC}(\psi)$ denote the corresponding model-checking problem: Check whether ψ holds for a given a $(k+1)$-partite[2] graph with vertex parts $X_1, \ldots, X_k, Y$.

▶ **Theorem 1** (Dichotomy). *For any $\exists^k\forall$-quantified graph property ψ, deciding* $\mathrm{MC}(\psi)$ *either requires time $m^{k-o(1)}$ under the assumption that* MAX-3-SAT *has no $O((2-\varepsilon)^n)$-time algorithm for any $\varepsilon > 0$, or we give an $O(m^{k-\varepsilon})$-time algorithm for some $\varepsilon > 0$.*

[2] A discussion on this assumption follows in Section 2.

In fact, we base our hardness results even on the weaker 3-uniform HyperClique assumption as introduced in [5, 49]. Formally, the h-uniform HyperClique hypothesis states for any parameter $h \geq 3$:

▶ **Hypothesis** (h-uniform HyperClique). *For no $\varepsilon > 0$ and $k \geq h+1$, there is an $O(n^{k-\varepsilon})$-time algorithm for detecting a k-clique in h-uniform hypergraphs.*

We defer a discussion of the plausibility of the MAX-3-SAT and h-uniform HyperClique hypotheses to Section 2.2 and the detailed treatment in [49, Section 7].

Beyond Theorem 1, we gain deeper insights into the complexity landscape of first-order graph properties. In particular, we expose a fine-grained hardness hierarchy purely depending on a hardness parameter $h = H(\psi)$ which we define below (illustrated in Figure 1): if $h = k$, then a lower bound of $m^{k-o(1)}$ can be derived from the k-OV conjecture, and thereby from SETH. On the other extreme, if $h \leq 2$, we give $O(m^{k-\varepsilon})$-time algorithms (for some $\varepsilon > 0$) – here, the difference between hardness 1 and 2 is precisely whether or not fast matrix multiplication techniques are likely to be necessary. For the remaining cases of $3 \leq h < k$, we can derive a lower bound of $m^{k-o(1)}$ under the 3-uniform HyperClique conjecture. In fact, we obtain increasing levels of hardness, as the lower bound for hardness-h formulas follows from the h-uniform HyperClique conjecture.

For the definition of our hardness parameter, it turns out that the decisive information is given by the atoms $E(x_i, y)$ for some existentially quantified variable x_i and the universally quantified y. Specifically, consider the formula ϕ_0 obtained by setting all atoms $E(x_i, x_j), 1 \leq i < j \leq k$ in ϕ to *false*. Observe that we can view ϕ_0 as a Boolean function $\{0,1\}^k \to \{0,1\}$ which maps the values $(E(x_1, y), \ldots, E(x_k, y))$ to a truth value. The hardness h of ψ is then given by the following hardness parameter $H(\phi_0)$. To state its definition, we need the following notation: For a propositional formula $f(z_1, \ldots, z_k)$ and an index set $I \subseteq [k]$, an *I-restriction of f* is a formula obtained from f after substituting all variables z_i, $i \in I$, by constant values from $\{0,1\}$.

▶ **Definition 2.** *We call a propositional formula $f(z_1, \ldots, z_k)$* h-hard, *$0 \leq h \leq k$, if, for any index set $I \in \binom{[k]}{k-h}$, there exists some I-restriction of f with exactly one falsifying assignment. Further define the* hardness $H(\psi)$ *as the maximum number h for which ψ is h-hard (for constant-valued f, we set $H(f) = 0$).*

Intuitively, $H(f)$ is the largest arity k such that whenever we fix an arbitrary subset of all but k variables, we can still obtain a "k-OV-like" function (a function with only a single falsifying assignment) as a restriction.

The following theorem is a fine-grained version of our dichotomy in Theorem 1.

▶ **Theorem 3** (Hardness levels). *For a first-order property ψ as in (1), let $\phi_0 : \{0,1\}^k \to \{0,1\}$ denote the formula obtained from ϕ by replacing all occurrences of $E(x_i, x_j)$ by false. We call $H(\psi) := H(\phi_0)$ the* hardness of ψ. *For $h = H(\psi)$, it holds that*

- *If $h \leq 1$, then $\mathrm{MC}(\psi)$ is decidable in time $O(m^{k-\varepsilon})$ for some $\varepsilon > 0$ combinatorially*[3].
- *If $h \leq 2 < k$, then $\mathrm{MC}(\psi)$ is decidable in time $O(m^{k-\varepsilon})$ for some $\varepsilon > 0$ using fast matrix multiplication. (Furthermore, $\mathrm{MC}(\psi)$ cannot be decided by a combinatorial $O(m^{k-\varepsilon})$-time algorithm unless the combinatorial k-Clique hypothesis*[4] *is false.)*

[3] Informally, by *combinatorial* algorithms we mean algorithms that do not rely on algebraic methods like fast matrix multiplication [3].

[4] The *combinatorial k-Clique hypothesis* as stated in, e.g., [3, 49] postulates that – even though an $O(n^{\frac{\omega}{3}k})$-time algorithm is known for k-Clique – no such polynomial improvement over the naïve $O(n^k)$ solution can be achieved by a *combinatorial* algorithm.

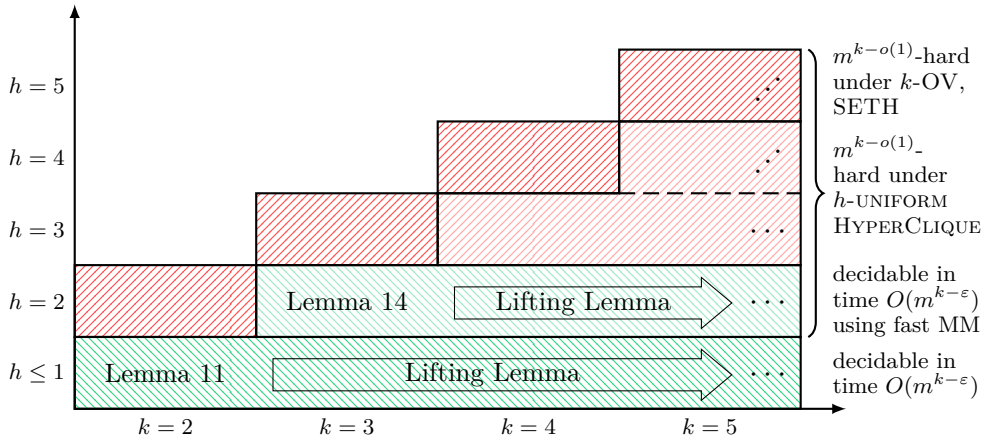

Figure 1 Visualizes the hardness of $\mathrm{MC}(\psi)$ for $\exists^k\forall$-quantified graph properties ψ of hardness $h = H(\psi)$. The green-hatched areas designate instances that allow polynomial improvements over the baseline, while the red regions turn out to be provably hard.

- *If $3 \leq h \leq k$, then $\mathrm{MC}(\psi)$ cannot be decided in time $O(m^{k-\varepsilon})$ for any $\varepsilon > 0$ unless the h-*UNIFORM HYPERCLIQUE *hypothesis fails.*
- *If $h = k$, then $\mathrm{MC}(\psi)$ cannot be decided in time $O(m^{k-\varepsilon})$ for any $\varepsilon > 0$ unless the k-*OV *hypothesis fails.*

Note that under the plausible assumption that h-UNIFORM HYPERCLIQUE gets strictly harder for increasing arity h, our classification exposes increasing levels of hardness within the first-order graph properties. This claim is substantiated by the following observation (whose proof is deferred to the full version of the paper).

▶ **Proposition 4.** *Let $h \geq 3$. There exist $k > h$, $\varepsilon > 0$, and an $\exists^k\forall$ graph property ψ of hardness h that can be decided in time $O(m^{k-\varepsilon})$ if and only if the h-*UNIFORM HYPERCLIQUE *hypothesis fails.*

To give a specific illustration of our results, consider the first-order property TWINFREE. By negating the formula, we obtain an equivalent graph property as in (1) where ϕ_0 is the constant false formula. As such, our classification yields that TWINFREE is decidable in time $O(m^{2-\varepsilon})$ (in fact, it is even decidable by an $O(m)$-time algorithm [42]).

Another interesting family of examples is the following k-Not-All-Equal Problem.

▶ **Example 5.** Let $\mathrm{NAE}(z_1, \ldots, z_k)$ be falsified only by the all-zero or all-one assignment. The k-Not-All-Equal (k-NAE) problem is given by the query

$$(\exists x_1 \in X_1) \ldots (\exists x_k \in X_k)\,(\forall y \in Y)\,\mathrm{NAE}(E(x_1, y), \ldots, E(x_k, y)).$$

It is easy to check that $H(k\text{-NAE}) = k - 1$. Thus, by Theorem 3,

- 2-NAE is decidable in time $O(m^{2-\varepsilon})$ for some $\varepsilon > 0$.
  (In fact, we give an $O(m)$-time algorithm.)
- 3-NAE is decidable in time $O(m^{3-\varepsilon})$ for some $\varepsilon > 0$ using fast matrix multiplication,
- k-NAE takes time $m^{k-o(1)}$ for any $k > 3$ unless the $(k-1)$-UNIFORM HYPERCLIQUE hypothesis fails.

Finally, we extend our results and discuss further directions in Section 7: In particular, we extend Theorem 3 to a counting dichotomy. Furthermore, we give tighter bounds on the running time exponent for properties that admit polynomial improvements over the baseline, using $k = 3$ as a case study.

1.5 Outline

After setting up notation and detailing the hardness assumptions used in this work in Section 2, we give a technical overview of our proof and introduce our main algorithmic tools in Section 3. Following the outline described in Section 3, we prove our main result in Sections 4, 5 and 6. Finally, we discuss our extensions and give an outlook for future work in Section 7. The proofs of our extensions are deferred to the full version of this paper.

2 Preliminaries

Let us clarify some notation first. For a non-negative integer k, let $[k] := \{1, \dots, k\}$. By $\uplus$ we denote the disjoint union of sets and for any set I, by $\binom{I}{k}$ we address the set of all k-element subsets of I. For a 0-1 vector x, we write $\bar{x}$ for the complement of x and $\|x\|$ to denote the Hamming weight (that is, the ℓ_1-norm) of x. Occasionally, we apply bit-wise binary operations to vectors understood as component-wise application. We further employ the Iverson bracket notation, that is, we write $[P]$ to denote the truth value of a proposition P. Let $f(z_1, \dots, z_k)$ be a propositional formula and let $I \subseteq [k]$. For any assignment $\alpha : I \to \{0, 1\}$, we write $f|_\alpha$ to denote the formula obtained from f after substituting z_i by α_i, for all $i \in I$. Finally, by $\omega \leq 2.373$ we denote the exponent of matrix multiplication.

2.1 Model-Checking

A *relational structure* consists of n objects and predicates of arbitrary arity relating these objects. These predicates are explicitly given as lists of records; let m denote the total number of such facts. Without loss of generality we assume $n \leq O(m)$ by ignoring objects not occurring in any relation. A *first-order property* is given by a quantified formula

$$(Q_1 x_1) \dots (Q_k x_k)\, \phi(x_1, \dots, x_k),$$

where each quantifier $Q_i \in \{\exists, \forall\}$ ranges over all objects of the relational structure. The proposition ϕ is allowed to contain Boolean connectives and its atoms are given by predicates relating the quantified objects. The problem $\mathrm{MC}(\psi)$ of checking whether a fixed first-order property ψ holds on a given sparse structure is called the *model-checking problem* (or *query evaluation problem*) for ψ.

In this paper, we consider a fragment that we call $\exists^k\forall$*-quantified graph properties*: Here the input is a $(k+1)$-partite graph $G = (X_1 \uplus \dots \uplus X_k \uplus Y, E)$ and the task is to model-check the fixed formula

$$\psi = (\exists x_1 \in X_1) \dots (\exists x_k \in X_k)\, (\forall y \in Y)\, \phi(x_1, \dots, x_k, y),$$

where ϕ is an arbitrary Boolean formula formed from a single edge predicate E of arity 2 (i.e., the atoms of ϕ are of the form $E(v, v')$ with $v, v' \in \{x_1, \dots, x_k, y\}$). We assume the edge predicate to be symmetric (i.e., G to be a symmetric graph). We adopt the convention that x_i (respectively y) ranges over X_i (respectively Y) and therefore omit to explicit state membership $x_i \in X_i$ when it is clear from the context. Borrowing the notation from the definition of Sparse OV, we let $d := |Y|$ denote the number of objects in the range of the universally quantified variable. Since $\mathrm{MC}(\psi)$ is solvable in time $O(m)$ for $k \leq 2$, we assume throughout the paper that $k \geq 2$.

Let us explicitly highlight a subtle point: An alternative natural formalization of graph properties would be to omit the assumption that the given graph is $(k+1)$-partite. This alters the flavor of model-checking problems slightly: While all of our upper bounds would also

hold for this alternative formulation (by replicating the vertex set), the assumption cannot be neglected for the lower bounds. In fact, there exist examples of first-order properties that we prove hard if each quantifier ranges over its own part, and which turn out as easy if instead each quantifier ranges over the whole vertex set.[5] We leave open an investigation of this alternative formulation for future work.

2.2 Hardness Assumptions

We briefly collect the hardness assumptions used in our classification result.

▶ **Hypothesis** (k-OV). *For no k and $\varepsilon > 0$, k-OV on n vectors with dimension d can be solved in time $O(n^{k-\varepsilon} \operatorname{poly} d)$.*

The fastest known algorithm solves k-OV in time $n^{2-\Omega(1/\log(d/\log n))}$ [7, 29]. On the hardness side, the k-OV hypothesis is implied by SETH [57], the weighted k-Clique hypothesis [5], and the hypothesis that not all $(k+1)$-quantifier first-order properties can be solved in time $O(m^{k-\varepsilon})$ [41]. We remark that we use the *moderate-dimensional* k-OV hypothesis here (in fact, SETH even implies a stronger version postulating hardness just above logarithmic dimension).

The most important hypothesis for this work concerns the HyperClique problem: Given an h-uniform hypergraph H, the h-uniform k-HyperClique problem asks to find vertices $v_1, \dots, v_k \in V(H)$ so that any size-h subset of $\{v_1, \dots, v_k\}$ is contained in $E(H)$. This gives rise to the following hypothesis for any $h \geq 3$.

▶ **Hypothesis** (h-uniform HyperClique). *For no $\varepsilon > 0$ and $k \geq h+1$, there is an $O(n^{k-\varepsilon})$-time algorithm for h-uniform k-HyperClique.*

The restriction $h \geq 3$ is indeed essential: The 2-uniform k-HyperClique problem – i.e., the k-Clique problem on ordinary graphs – is known to admit faster solutions. Lincoln et al. [49] provides a detailed analysis on why to believe that the $h \geq 3$ case should be significantly harder: As a main argument, any improvement over the $O(n^k)$-time k-Clique algorithm traces back to fast matrix multiplication, however, Strassen-like algebraic techniques can provably not be applied to the HyperClique setting [49]. Moreover, there is a reduction from MAX-3-SAT to h-uniform k-HyperClique ($h \geq 3$), showing that the h-uniform HyperClique conjecture is entailed by the following MAX-3-SAT hypothesis, which is the simplest justification for our hardness results.

Specifically, consider the MAX-3-SAT problem, which asks, given a 3-SAT instance, to find an assignment maximizing the number of satisfied clauses.

▶ **Hypothesis** (MAX-3-SAT). *For all $\varepsilon > 0$,* MAX-3-SAT *cannot be solved in time $O((2-\varepsilon)^n)$.*

The currently fastest known algorithm for MAX-3-SAT runs in time $2^{n-o(n)}$ [11]. This assumption implies both the 3-uniform k-HyperClique [49] and the OV [5] hypotheses, and thus provides the currently *easiest* barrier for algorithmic improvements upon formulas that we classify as hard.

---

[5] Consider the property $\psi = (\exists x_1 \in V) \dots (\exists x_k \in V)\,(\forall y \in V) \bigvee_{i=1}^{k} E(x_i, y)$, which is equivalent to the *k-Dominating Set* problem. It is easy to see that $\mathrm{MC}(\psi)$ can be decided in time $O(m^{k-1})$: For any solution $(x_1, \dots, x_k)$, there must exist one "heavy" vertex x_i dominating at least n/k vertices y. However, there can be at most $O(m/n)$ many such vertices x_i. It is feasible to explicitly enumerate all heavy vertices and solve the remaining k-quantifier problem in $O(n^{k-2}m)$ time using the baseline algorithm. The total running time is $O(m/n \cdot n^{k-2}m) = O(m^{k-1})$. However, if the quantifiers of ψ range over separate sets $X_1, \dots, X_k, Y$, then deciding $\mathrm{MC}(\psi)$ requires time $m^{k-o(1)}$ (Theorem 3).

3 Technical Overview

To prove our result, we introduce the following type of $\exists^k\forall$-quantified graph properties, in which we interpret the input graph $G = (V = (X_1 \uplus \cdots \uplus X_k \uplus Y), E)$ as sets of *vectors* $X_1, \ldots, X_k$, by setting the entry $x_i[y] \in \{0,1\}$ to be 1 if and only if the edge (x_i, y) is present in G, i.e, $x_i[y] = [E(x_i, y)]$ (for any $x_i \in X_i, i \in [k]$, and $y \in Y$). For any Boolean function $\phi : \{0,1\}^k \to \{0,1\}$, we define the corresponding *Vector Problem* $\mathrm{VP}(\phi)$

$$(\exists x_1 \in X_1) \ \ldots \ (\exists x_k \in X_k)\,(\forall y \in Y)\,\phi(x_1[y], \ldots, x_k[y]).$$

Intuitively, Vector Problems are a proper subclass of $\exists^k\forall$-quantified graph properties, since the latter additionally considers the edges $E(x_i, x_j)$. Note that Sparse k-OV coincides with $\mathrm{VP}(\phi)$ for $\phi(z_1, \ldots, z_k) = \bigvee_{i=1}^k \bar{z_i}$; in particular, this function only has a single falsifying assignment.

We prove our main dichotomy (Theorems 1 and 3) by first proving an analogous dichotomy for Vector Problems (see Theorem 6) and then showing an equivalence between Vector problems and general $\exists^k\forall$-quantified graph properties (see Theorem 7).

For the first step, we show that the complexity of a Vector Problem $\mathrm{VP}(\phi)$ is determined by the parameter $H(\phi)$ as defined in Definition 2:

▶ **Theorem 6.** *Let ϕ be a k-variable formula of hardness $h = H(\phi)$.*

- *If $h \leq 1$, then $\mathrm{VP}(\phi)$ is decidable in time $O(m^{k-1})$ combinatorially.*
- *If $h \leq 2 < k$, then $\mathrm{VP}(\phi)$ is decidable in time $O(m^{k-\varepsilon})$ for some $\varepsilon > 0$ using fast matrix multiplication. (Furthermore, $\mathrm{VP}(\phi)$ cannot be decided by a combinatorial $O(m^{k-\varepsilon})$-time algorithm unless the combinatorial k-clique hypothesis is false.)*
- *If $3 \leq h \leq k$, then $\mathrm{VP}(\phi)$ cannot be solved in time $O(m^{k-\varepsilon})$ for any $\varepsilon > 0$ unless the h-uniform HyperClique hypothesis fails.*
- *If $h = k$, then $\mathrm{VP}(\phi)$ cannot be solved in time $O(m^{k-\varepsilon})$ for any $\varepsilon > 0$ unless the k-OV hypothesis fails.*

This result is established as follows (see Figure 1 for an illustration). On the algorithmic side, we show that:

1. For any 2-variable formula ϕ of hardness 1, $\mathrm{VP}(\phi)$ can be solved in time $O(m)$ (Section 4.1).
2. For any 3-variable formula ϕ of hardness 2, $\mathrm{VP}(\phi)$ can be solved in time $O(m^{3-\varepsilon})$ for some $\varepsilon > 0$ (Section 4.2). This is our technically most demanding contribution, for which we introduce a framework of *Constrained Triangle* problems solvable in subcubic time (Section 3.1).
3. These algorithms can be lifted to higher number of variables k (Section 4.3). The idea is to brute force over all but 2 or 3 variables and to apply the algorithms above. However, these algorithms are not directly applicable, since the brute-forcing step does not reduce to solving a *single* $\mathrm{VP}(\phi)$ instance, but to solving a *mixture* of $\mathrm{VP}(\phi_i)$ instances for a constant number of formulas $\phi_1, \ldots, \phi_\ell$. To overcome this issue, we consider *Hybrid Vector Problems* $\mathrm{VP}(\Phi)$ for a set of formulas Φ and show that our algorithms from above apply whenever $\max\{H(\phi) \mid \phi \in \Phi\} \leq 1$ or 2, respectively. This extension to Hybrid Vector Problems is made possible by the generality of our Constrained Triangle framework that allows for a surprisingly simple combination of constraints for different formulas.

On the hardness side, for any k-variable hardness-h formula ϕ, we give a fine-grained reduction from finding a k'-clique in h-uniform hypergraphs to the model-checking problem for ϕ, where k' is a sufficiently large constant. Intuitively (and somewhat oversimplified), the variables $x_1, \ldots, x_k$ choose a k-clique in a k-partite hypergraph and the $(\forall y)\,\phi(x_1[y], \ldots, x_k[y])$-part verifies that $(x_1, \ldots, x_k)$ indeed forms a clique. To this end, y ranges over the non-edges

of the hypergraph and verifies that the vertices $v_1, \dots, v_h$ of such a non-edge are not included in $x_1, \dots, x_k$. Specifically, let J denote the parts containing $v_1, \dots, v_k$. Then by our hardness definition, we find a way to assign values to $x_i[y]$ for all $x_i \in X_i$ with $i \in [k] \setminus J$ such that we can exclude exactly the vertices $v_1, \dots, v_h$ by finding suitable values for $x_i[y]$ for all $i \in X_i$ with $i \in J$. The proof is given in Section 5.

In the second step, we extend our classification from Vector Problems to all $\exists^k\forall$ graph properties by the following equivalence.

▶ **Theorem 7.** *Let $\psi = (\exists x_1) \dots (\exists x_k)\,(\forall y)\,\phi(x_1, \dots, x_k, y)$ be an $\exists^k\forall$ graph property and let ϕ_0 denote the formula ϕ after substituting each predicate $E(x_i, x_j)$ by false.*

- *If $\mathrm{MC}(\psi)$ is decidable in time $T(m)$, then $\mathrm{VP}(\phi_0)$ is decidable in time $O(T(m))$.*
- *If $\mathrm{VP}(\phi_0)$ is decidable in time $O(m^{k-\varepsilon})$ for some $\varepsilon > 0$, then $\mathrm{MC}(\psi)$ is decidable in time $O(m^{k-\varepsilon'})$ for some $\varepsilon' > 0$.*

For an intuition for the proof of the non-trivial direction from general properties to Vector Problems, let us call $(x_1, \dots, x_k)$ a *solution* if $(\forall y \in Y)\,\phi(x_1, \dots, x_k, y)$ holds. We reduce the problem of detecting a solution $(x_1, \dots, x_k)$ such that for some $1 \leq i < j \leq k$, the edge $E(x_i, x_j)$ is present to the problem of counting triangles in a (sparse) graph. The remaining solutions *almost* correspond to solutions of the Vector Problem ϕ_0 – however, we need to additionally ensure that *no* edge $E(x_i, x_j)$ is present in such a solution. We overcome this technical issue using a counting argument that there are few solutions with at least one edge $E(x_i, x_j)$ present.

Theorems 1 and 3 then follow by combining Theorem 6 and 7.

3.1 Constrained Triangles Framework

Let us detail our main algorithmic tool in advance. We develop a convenient framework to detect triangles subject to an arbitrary combination of some well-behaved constraints. We achieve subcubic-time algorithms by employing fast matrix multiplication in combination with a careful constraint-specific analysis.

The problem of detecting three pairwise adjacent vertices in a tripartite graph $G = (V_1 \uplus V_2 \uplus V_3, E)$, is referred to as Triangle. It serves us as a combinatorial intermediate problem that immediately benefits from the significantly improved running time of fast matrix multiplication over the $O(n^3)$-time naïve approach. However, Triangle is of relatively little expressiveness as the only way to encode information into a Triangle instance is to customize the edge set E. We therefore strengthen Triangle by allowing certain *constraints* to further restrict the set of feasible solutions (v_1, v_2, v_3). Specifically, we make use of two types of constraints:

Sum: For edge weights $w : E \to \mathbb{Z}$, where $\sum_{e \in E} |w(e)| \leq O(n^2)$, and a target $t \in \mathbb{Z}$, we require that $w(v_1, v_2) + w(v_2, v_3) + w(v_3, v_1) = t$.

Equal: For edge weights $w : E \to \mathbb{Z}$, we require that $w(v_1, v_2) = w(v_1, v_3)$.

This prepares us to introduce *Constrained Triangle problems* in an inductive fashion: As the base case, Triangle is viewed as a Constrained Triangle problem. In addition, for any Constrained Triangle problem Δ and any constraint $C \in \{\text{Sum}, \text{Equal}\}$, by $\Delta[C]$ we understand the Constrained Triangle problem which is – on top of all constraints restricting Δ – constrained by C. We remark that each constraint features its own weight function (which is given as part of the input), so in particular an instance of the Constrained Triangle problem $\text{Triangle}[C_1] \cdots [C_r]$ is equipped with r weight functions corresponding to the respective Sum and Equal constraints C_i.

Arguably, Constrained Triangle problems seem to be a convenient "interface" to the algorithmic power of fast matrix multiplication, as we can specifically tailor constraints in the desired manner. Indeed, even subject to any constant number of Sum and Equal constraints, finding triangles remains subcubic.

▶ **Lemma 8.** *Let Δ be a Constrained Triangle problem. Then Δ can be decided in time $O(n^{3-\varepsilon})$ for some $\varepsilon > 0$.*

The proof of Lemma 8 is by induction on the structure of Δ; we consider the three possible cases below in Fact 1, Lemma 9 and Lemma 10. A crucial observation which we will exploit often, is that for any Constrained Triangle problem Δ, given (v_1, v_2, v_3), we can test in constant time whether (v_1, v_2, v_3) is a solution of Δ.

By applying fast (Boolean) matrix multiplication, it is well-known that Triangle is decidable in subcubic time:

▶ **Fact 1.** Triangle *is decidable in time $O(n^\omega)$.*

This settles the induction base. In the following two lemmas, we assume efficient algorithms for Δ and aim to find algorithms for Δ[Sum] and Δ[Equal], respectively.

We focus on Δ[Sum] first. Note that the restriction $\sum_{e\in E} |w(e)| \leq O(n^2)$ is indeed necessary: For unbounded weights, the problem of finding an exact-weight triangle is not known and in fact conjectured not to be decidable significantly faster than $O(n^3)$ [56]. Nevertheless, under this condition we achieve an efficient Δ[Sum] algorithm:

▶ **Lemma 9.** *If Δ is decidable in time $O(n^{3-\varepsilon})$ for some $\varepsilon > 0$, then Δ[Sum] is decidable in time $O(n^{3-\varepsilon'})$ for some $\varepsilon' > 0$.*

Proof. We call an edge *large-weight* if its weight exceeds n^δ in absolute value (where δ is yet to be fixed) and *small-weight* otherwise. Our first step is to eliminate all large-weight edges. By assumption, since $\sum_{e\in E} |w(e)| \leq O(n^2)$, there can be at most $O(n^2/n^\delta) = O(n^{2-\delta})$ many such edges. Thus, it is feasible to enumerate all large-weight edges (x_i, x_j) and all vertices x_ℓ in the remaining part X_ℓ, for all distinct i, j, ℓ. For each triple considered in that way, we explicitly check that

- (x_i, x_j, x_ℓ) forms a triangle, and
- (x_i, x_j, x_ℓ) satisfies all constraints of Δ, and
- $w(x_i, x_j) + w(x_j, x_\ell) + w(x_\ell, x_i) = t$.

Since all these tests run in constant time, this whole step takes time $O(n^{2-\delta} \cdot n) = O(n^{3-\delta})$. We accept if a solution was found, and otherwise continue by safely removing all large-weight edges from the graph.

So from now on, we can assume that all remaining edges are small-weight. For any combination of weights $w_{12}, w_{23}, w_{31} \in \{-n^\delta, \ldots, n^\delta\}$ summing exactly to t, we create a Δ instance that only includes edges (v_i, v_j) of the weight $w(v_i, v_j) = w_{ij}$. We proceed to solve all these instances and report if a solution was found. By construction, any solution to an instance created in that way satisfies all constraints of Δ and the additional Sum constraint.

Solving a single Δ subinstance takes time $O(n^{3-\varepsilon})$ by assumption. There are $O(n^{2\delta})$ many combinations of weights $w_{12}, w_{23}, w_{31} \in \{-n^\delta, \ldots, n^\delta\}$ with $w_{12} + w_{23} + w_{31} = t$, so we need time $O(n^{2\delta} \cdot n^{3-\varepsilon}) = O(n^{3+2\delta-\varepsilon})$ to solve all instances. By setting $\delta := \frac{\varepsilon}{3}$, the claim follows. ◀

▶ **Lemma 10.** *If Δ is decidable in time $O(n^{3-\varepsilon})$ for some $\varepsilon > 0$, then Δ[Equal] is decidable in time $O(n^{3-\varepsilon'})$ for some $\varepsilon' > 0$.*

Proof. For any vertex $v_1 \in V_1$, we define $\deg_i(v_1)$ as the number of edges of weight i incident to v_1. First, we enumerate all edges (v_1, v_3), and if $\deg_{w(v_1,v_3)}(v_1) \leq n^{1-\delta}$ (where δ is a parameter to be chosen later), then enumerate all edges (v_1, v_2) of the same weight $w(v_1, v_3)$. When finding a triple (v_1, v_2, v_3) forming a triangle and fulfilling all constrained imposed by Δ, we accept. Otherwise, we can safely remove all edges (v_1, v_3) where $\deg_{w(v_1,v_3)}(v_1) \leq n^{1-\delta}$. We can similarly remove all edges (v_1, v_2) with $\deg_{w(v_1,v_2)}(v_1) \leq n^{1-\delta}$. So we can assume that for all weights i, and all vertices v_1, $\deg_i(v_1)$ is either 0 or at least $n^{1-\delta}$. This step takes time $O(n^2 \cdot n^{1-\delta}) = O(n^{3-\delta})$.

Since each vertex v_1 is incident to at most n edges, each v_1 can be incident to edges of at most $n/n^{1-\delta} = n^\delta$ different weights; we denote these weights by $w_1^{v_1}, \ldots, w_{n^\delta}^{v_1}$ in an arbitrary order. Our strategy is as follows: We create n^δ many Δ instances, where the i-th instance contains, for each vertex v_1, only those edges incident to v_1 which are of weight $w_i^{v_1}$. Notice that in each instance, all edges incident to one fixed v_1 are of the same weight, even though edges incident to different v_1's are in general of different weights. In this way, we can now simultaneously search for all triangles (v_1, v_2, v_3) satisfying Δ's constraints and satisfying $w(v_1, v_2) = w(v_1, v_3) = w_i^{v_1}$. Solving all instances takes time $O(n^\delta \cdot n^{3-\varepsilon}) = O(n^{3+\delta-\varepsilon})$, under the assumption that Δ can be solved in time $O(n^{3-\varepsilon})$.

In total, the running time is bounded by $O(n^{3-\delta} + n^{3+\delta-\varepsilon})$, which is subcubic, namely $O(n^{3-\frac{\varepsilon}{2}})$, for $\delta := \frac{\varepsilon}{2}$. ◀

4 Algorithmic Results

In this section, we show the algorithmic part of Theorem 6. In particular, we show that for an k-variable hardness-h formula ϕ the Vector Problem $\mathrm{VP}(\phi)$ is easy if $k = 2$ and $h \leq 1$ (Section 4.1), or if $k = 3$ and $h \leq 2$ (Section 4.2). In fact, for both scenarios, we demonstrate how to solve the following more general version of Vector Problems that discriminates among dimensions in such a way that we can assert different formulas ϕ for different dimensions.

For a set of k-variable formulas Φ, and given a sparse structure over the vertex set $X_1 \uplus \ldots \uplus X_k \uplus Y$, where each dimension $y \in Y$ is associated to a formula $\phi_y \in \Phi$, the *Hybrid Vector Problem* $\mathrm{VP}(\Phi)$ is to check

$$(\exists x_1 \in X_1) \ldots (\exists x_k \in X_k)\, (\forall y \in Y)\, \phi_y(x_1[y], \ldots, x_k[y]).$$

Our hardness notion is generalized to sets of formulas Φ by $H(\Phi) := \max\{H(\phi) : \phi \in \Phi\}$. (Observe that we could equivalently define Hybrid Vector Problems as Vector Problems tolerating a multi-valued unary predicate on Y.)

4.1 Algorithms for $k = 2$

We first introduce a useful tool: partition refinement [42]. There exists a (simple) data structure which, once initialized with a universe U, explicitly maintains a partition $\biguplus_P P$ of U. It supports an operation $\textsc{Refine}(S)$ that splits each part P into $P \cap S$ and $P \setminus S$ and runs in time $O(|S|)$. In addition, we can always query, for each universe element $u \in U$, its current part P (with $u \in P$) and iterate over all elements in P in time $O(|P|)$.

▶ **Lemma 11.** *Let Φ be a set of 2-variable formulas of hardness $H(\Phi) \leq 1$. Then $\mathrm{VP}(\Phi)$ is decidable in time $O(m)$.*

Proof. Let $\phi \in \Phi$. We first prove that $\mathrm{VP}(\phi)$ is linear-time decidable, and later show the same for $\mathrm{VP}(\Phi)$. Since we consider $k = 2$ variables, we have a function $\phi\colon \{0,1\}^2 \to \{0,1\}$. We exhaustively discriminate all possible shapes of ϕ:

Case 0: ϕ has 0 or 4 satisfying assignments. Then $\mathrm{VP}(\phi)$ is constantly rejecting or accepting and trivially decidable.

Case 1: ϕ has 1 satisfying assignment. Then ϕ is of the shape $\phi(z_1, z_2) = \phi_1(z_1) \wedge \phi_2(z_2)$. Deciding $\mathrm{VP}(\phi)$ is equivalent to asserting the satisfiability of $(\exists x_1)\,(\forall y)\,\phi_1(x_1[y])$ and $(\exists x_2)\,(\forall y)\,\phi_2(x_2[y])$ separately, which we do by removing all vectors x_1 and x_2 violating $(\forall y)\,\phi_1(x_1[y])$ and $(\forall y)\,\phi_2(x_2[y])$, respectively. Focus on the former problem: We first precompute all values $\|x_1\|$ by iterating over the edge set once. Then we exclude all vectors x_1 with $\|x_1\| < d$ in case that $\phi_1(z_1) = z_1$ or $\|x_1\| > 0$ in case that $\phi_1(z_1) = \bar{z}_1$. In the remaining cases $\phi_1(z_1) = \mathit{true}$ and $\phi_1(z_1) = \mathit{false}$ we keep all or no vectors x_1, respectively.

Case 2: ϕ has 2 satisfying assignments.

Case 2a: $\phi(z_1, z_2) = (z_1 \leftrightarrow z_2)$. Our algorithm relies on the partition refinement technique: Initialize the partition refinement structure with $U = X_1 \uplus X_2$. Then, for each $y \in Y$, invoke $\textsc{Refine}(N(y))$, where $N(y) \subseteq X_1 \uplus X_2$ denotes the neighborhood of y. We claim two vectors x_1 and x_2 lie in the same partition if and only if $(\forall y)\, x_1[y] \leftrightarrow x_2[y]$. Indeed, assume that x_1 and x_2 are equal (as vectors). Then for any dimension $y \in Y$, we have $x_1 \in N(y)$ if and only if $x_2 \in N(y)$. Hence, there exists no entry $y \in Y$ separating x_1 from x_2. On the other hand, if $x_1[y] \leftrightarrow x_2[y]$ does not hold for some $y \in Y$, then $x_1 \in N(y)$ but $x_2 \notin N(y)$ (or vice versa), so $N(y)$ splits any part containing both x_1 and x_2. This approach takes time $\sum_{y \in Y} O(|N(y)|) = O(m)$ for the refinement steps and time $\sum_P O(|P|) \leq O(n)$ (where P ranges over all parts) to ultimately check whether some pair (x_1, x_2) was not separated.

Case 2b: $\phi(z_1, z_2) = z_1 \oplus z_2$. Our goal is to reduce to the Case 2a by transforming all vectors x_1, x_2 into x_1', x_2' in such a way that we have $x_1[y] \oplus x_2[y]$ if and only if $x_1'[y] \leftrightarrow x_2'[y]$, for all $y \in Y$. The naive way to achieve this is to negate all vectors $x_2 \in X_2$; however, we then potentially increase the total number of 1-entries inordinately, so that we do not obtain an $O(m)$-time algorithm. We circumvent this issue as follows. Let us call a vector x *heavy* if $\|x\| > d/2$, and *light* otherwise. Observe that in any solution (x_1, x_2), we must have that precisely one of x_1 and x_2 is heavy (assuming without loss of generality that d is odd). Now, assign $x_i' := x_i$ if x_i is light and $x_i' := \bar{x}_i$ otherwise. Then, for any solution (x_1, x_2), where, say, x_1 is heavy, $(x_1', x_2') = (\bar{x}_1, x_2)$ is a solution of $\mathrm{VP}(x_1' \leftrightarrow x_2')$. The other direction is not immediate as there could exist light vectors x_1, x_2 with $x_1' = x_1 = x_2 = x_2'$. To avoid these false positives, we introduce a fresh dimension y with $x_1'[y] = [x_1 \text{ is heavy}]$ and $x_2'[y] = [x_2 \text{ is light}]$, for all vectors x_1, x_2. In doing so, we achieve that for any solution (x_1', x_2') exactly one of the vectors x_1, x_2 is heavy.

It remains to bound the running time of the complementations. Since there exist at most $O(m/d)$ heavy vectors, and complementing a single vector takes time $O(d)$, this step takes time $O(m/d \cdot d) = O(m)$. Furthermore, notice that m cannot increase by replacing heavy vectors with light ones.

Case 3: ϕ has 3 satisfying assignments. Then ϕ would have exactly one falsifying assignment, so it would have hardness 2. Since ϕ has hardness at most 1 by assumption, this case cannot occur.

To arrive at an algorithm solving the hybrid problem, we take a close look at the preceding arguments: Either $\mathrm{VP}(\phi)$ reduces to an instance of $\mathrm{VP}(z_1 \leftrightarrow z_2)$ over the original vertex set (Case 2) or we remove certain vertices and accept if afterwards both parts X_1 and X_2 are still non-empty (Cases 0 and 1). In the same way, we can solve the Hybrid Vector Problem $\mathrm{VP}(\Phi)$: For all formulas $\phi \in \Phi$ falling into the latter category, we identify and remove all bad vertices x_1 and x_2. Then, for all remaining $\phi \in \Phi$, we invoke the reduction to $\mathrm{VP}(z_1 \leftrightarrow z_2)$ and concatenate all vectors corresponding to the same vertex x_i. Finally, it remains to solve the combined $\mathrm{VP}(z_1 \leftrightarrow z_2)$ instance as shown in Case 2a. ◀

4.2 Algorithms for $k = 3$

Next, we will show that, for $k = 3$, any Vector Problem of hardness at most 2 reduces to a Constrained Triangle problem Δ. The reduction is always of the following form: Given a VP(ϕ) instance G with vertices $V(G) = X_1 \uplus X_2 \uplus X_3 \uplus Y$, the corresponding Constrained Triangle instance is a graph G' over the vertex set $V(G') = X_1 \uplus X_2 \uplus X_3$. Moreover, we wish to satisfy that, for all (x_1, x_2, x_3):

$$(\forall y)\, \phi(x_1[y], x_2[y], x_3[y]) \iff (x_1, x_2, x_3) \text{ is a solution of } \Delta \text{ on } G'.$$

To this end, the following sections describe how to encode ϕ by Equal and Sum constraints step-by-step. More precisely, each constraint "covers" a pair of falsifying assignments of ϕ. Let $(\alpha^1, \beta^1), \ldots, (\alpha^r, \beta^r)$ be pairs of falsifying assignments of ϕ containing each falsifying assignment at least once. Then we reduce VP(ϕ) to a Constrained Triangle problem $\Delta = \textsc{Triangle}[C_1] \cdots [C_r]$ with $C_i \in \{\textsc{Equal}, \textsc{Sum}\}$ such that for all i and for all (x_1, x_2, x_3):

$$(\forall y)\, (x_1[y], x_2[y], x_3[y]) \notin \{\alpha^i, \beta^i\} \iff (x_1, x_2, x_3) \text{ satisfies the constraint } C_i.$$

Note that we indeed need the restriction $H(\phi) \leq 2$, since we can only cover the falsifying assignments of ϕ by pairs (α^i, β^i) if ϕ does not have exactly one falsifying assignment.

Recall that in Section 3.1, we measured the complexity of Constrained Triangles in terms of the number of vertices n. By viewing G' as a graph of m vertices (by potentially adding isolated nodes), we obtain algorithms whose running time solely depends on m. Furthermore, in the special case of Sum constraints, we thereby allow a total edge weight of up to $O(m^2)$ instead of $O(n^2)$.

4.2.1 Equal Constraints

We start by covering falsifying assignments α, β of Hamming distance 2 using Equal constraints.

▶ **Lemma 12.** *Let $\alpha, \beta \in \{0,1\}^3$ be of Hamming distance 2. In time $\tilde{O}(m^2)$, we can determine the edge weights of an* Equal *constraint C such that $(\forall y)\, (x_1[y], x_2[y], x_3[y]) \notin \{\alpha, \beta\}$ holds if and only if (x_1, x_2, x_3) satisfies C.*

Proof. Let $\alpha = (\alpha_1, \alpha_2, \alpha_3)$ and $\beta = (\beta_1, \beta_2, \beta_3)$. Without loss of generality, we assume that α equals β exactly in the first position. An Equal constraint is employed as follows by choosing each weight to be a dimension-d bit-vector (for the moment). Let $w(x_1, x_2)[y] := 1$ if and only if $(x_1[y], x_2[y]) = (\alpha_1, \alpha_2)$ and analogously let $w(x_1, x_3)[y] := 1$ if and only if $(x_1[y], x_3[y]) = (\beta_1, \beta_3)$.

Let y be arbitrary. It is easy to check that we have $w(x_1, x_2)[y] \neq w(x_1, x_3)[y]$ if and only if $(x_1[y], x_2[y], x_3[y]) \in \{\alpha, \beta\}$. Indeed, suppose that $1 = w(x_1, x_2)[y] \neq w(x_1, x_3)[y] = 0$. By our choice of $w(x_1, x_2)$, we must have $x_1[y] = \alpha_1$ and $x_2[y] = \alpha_2$. But $w(x_1, x_3)[y] = 0$, hence $x_3[y] = \bar{\beta}_3 = \alpha_3$. In summary: $(x_1[y], x_2[y], x_3[y]) = \alpha$. Symmetrically, $0 = w(x_1, x_2)[y] \neq w(x_1, x_3)[y] = 1$ entails $(x_1[y], x_2[y], x_3[y]) = \beta$. The converse direction is immediate.

However, computing the vectors $w(x_1, x_i)$ explicitly would take time $O(m^3)$, since there are $O(m^2)$ weights and each vector is of dimension $d = O(m)$. We therefore employ the following succinct computation and compression:

1. Compute all weights $w(x_1, x_i)$, $i = 2, 3$, stored in *run-length encoding*. That is, for each weight $w(x_1, x_i)$ we store a sequence of numbers $\mathrm{RLE}(w(x_1, x_i)) := \ell_1 \ell_2 \cdots \ell_r$ indicating the positions of 0-1 alternations in $w(x_1, x_i)$, i.e., $w(x_1, x_i) = 0^{\ell_1}\, 1^{\ell_2}\, \cdots\, 0^{\ell_{r-1}}\, 1^{\ell_r}$ written as a bit-string.

2. Interpret the run-length encoded weight vectors as strings, sort these strings to combine equal weight vectors, and replace every weight vector by its rank in the resulting sorted sequence. This replaces each weight vector by a number in $\{0, \dots, O(m^2)\}$ so that two edges share the same label if and only if their weights are equal.

To implement the first step, we identify Y with $[d]$ in an arbitrary order. Fix some $i = 2, 3$ and some edge (x_1, x_i) and let a 0-1 alternation occur at y, i.e. $w(x_1, x_i)[y] \neq w(x_1, x_i)[y+1]$. We observe that, for any choice of α, β, at least one of the entries $x_1[y]$, $x_i[y]$, $x_1[y+1]$ or $x_i[y+1]$ is 1. Thus, it is feasible to explicitly consider all "event points" y with $x_1[y] = 1$ or $x_i[y] = 1$ in increasing order. Whenever an alternation is detected at $w(x_1, x_i)[y-1]$ or $w(x_1, x_i)[y]$ we appropriately append the run-length encoding of $w(x_1, x_i)$. For a fixed pair (x_1, x_i), this approach takes time $O(\|x_1\| + \|x_i\|)$. In total, we obtain time $\sum_{x_1,x_i} O(\|x_1\| + \|x_i\|) = O(n^2+m^2) = O(m^2)$. By the same argument, it follows that the total length of all run-length encoding $\sum_{x_1,x_i} |\mathrm{RLE}(w(x_1, x_i))|$ is bounded by $O(m^2)$.

In spirit, the proof ends here. However, in requiring the edge labels of Equal constraints to be integers instead of arbitrary objects (here, vectors in run-length encoding), we did not have to worry about the bit-size of the edge weights in Section 3.1. So our second step is to associate all vectors $w(x_1, x_i)$ with integers in $\{0, \dots, O(m^2)\}$. We interpret each run-length encoded weight $\mathrm{RLE}(w(x_1, x_i)) := \ell_1\ell_2 \cdots \ell_r$ as a string of length r over alphabet $\Sigma = [d]$. We sort these strings, which leaves all equal weights as contiguous intervals in the sorted sequence. Finally, we replace each weight by its rank in this sorted sequence (i.e., by the number of distinct weights preceding it in the sorted sequence). It is well-known that M strings of total length N over an alphabet of size $\mathrm{poly}(N)$ can be sorted in time $\tilde{O}(N)$ using tries. This yields time $\tilde{O}(m^2)$ in our application. ◀

4.2.2 Sum Constraints

It remains to cover falsifying assignments α, β of odd Hamming distance, for which we will use Sum constraints. We start with some useful observations.

We aim to check whether some vectors $(x_1, \dots, x_k)$ satisfy $(\forall y)\, \phi(x_1[y], \dots, x_k[y])$. Say ϕ is falsified only by the all-ones input. Then, clearly, it is sufficient to check that there exists no vertex y connected to all vertices $x_1, \dots, x_k$ – we call such a configuration a $[k]$-*star*. More generally, by an I-*star*, $I \subseteq [k]$, we understand a subgraph centered at a vertex $y \in Y$ such that all edges (x_i, y), $i \in I$, are present. Notice that, for any vectors $(x_1, \dots, x_k)$, $\| \bigwedge_{i\in I} x_i\|$ exactly counts the number of I-stars. So, in the above example of ϕ, we can decide whether $(\forall y)\, (x_1[y], \dots, x_k[y])$ by checking whether the number of $[k]$-stars $\| \bigwedge_{i=1}^{k} x_i\|$ equals zero.

But what about other formulas ϕ? For each falsifying assignment $\alpha \in \{0, 1\}^k$, the obvious generalization is to require $\| \bigwedge_{i=1}^{k} x_i^{\alpha_i}\| = 0$; here, and for the remainder of this section, we write $x^1 := x$ and $x^0 := \bar{x}$. The following observations suggest how to transform an arbitrary expression of this form into a linear combination of terms $\| \bigwedge_{i\in I} x_i\|$ without complemented occurrences:

▶ **Observation 1.** *For all vectors x, x', $\|x \wedge \bar{x}'\| = \|x\| - \|x \wedge x'\|$.*

By induction, we obtain the following generalization:

▶ **Observation 2.** *For all vectors $x, x_1, \dots, x_k$, $\|x \wedge \bigwedge_{i=1}^{k} \bar{x}_i\| = \sum_{I\subseteq[k]} (-1)^{|I|} \|x \wedge \bigwedge_{i\in I} x_i\|$.*

Thus, if we could precompute the number of I-stars $\| \bigwedge_{i\in I} x_i\|$, then we could efficiently test whether $(\forall y)\, \phi(x_1[y], \dots, x_k[y])$ holds:

▶ **Observation 3.** *Given vertices $(x_1, \dots, x_k)$ and the number of I-stars $\| \bigwedge_{i \in I} x_i \|$ for all $I \subseteq [k]$, for any formula ϕ we can decide in constant time whether $(\forall y)\, \phi(x_1[y], \dots, x_k[y])$.*

As an example, consider the 3-Not-All-Equal problem VP(NAE), where $\mathrm{NAE}(x_1, x_2, x_3)$ has two falsifying assignments: 111 and 000. Equivalently, we could require that the triple (x_1, x_2, x_3) satisfies $\|x_1 \wedge x_2 \wedge x_3\| = 0$ and $\|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\| = 0$. By Observation 2, the second equation is rewritten in an inclusion-exclusion fashion as

$$d - \|x_1\| - \|x_2\| - \|x_3\| + \|x_1 \wedge x_2\| + \|x_1 \wedge x_3\| + \|x_2 \wedge x_3\| - \|x_1 \wedge x_2 \wedge x_3\| = 0$$

(here, $d = \| \bigwedge_{i \in \emptyset} x_i \|$).

However, even though we can efficiently determine all values $\| \bigwedge_{i \in I} x_i \|$ in time $O(m^{|I|})$, computing $\| \bigwedge_{i=1}^{k} x_i \|$ is infeasible if we want to beat the baseline algorithm. Therefore, our next algorithm makes use of another trick: When combining two equations – as in the NAE example – we can sometimes exploit cancellations to decide instances without actually computing $\| \bigwedge_{i=1}^{k} x_i \|$: Because the Hamming weight of all vectors is always non-negative, instead of testing $\|x_1 \wedge x_2 \wedge x_3\| = 0$ and $\|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\| = 0$, we can equivalently test whether $\|x_1 \wedge x_2 \wedge x_3\| + \|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\| = 0$. By expanding $\|\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3\|$ as above, the $\|x_1 \wedge x_2 \wedge x_3\|$ term cancels and it suffices to know the numbers of all I-stars, for $I \subsetneq [3]$.

▶ **Lemma 13.** *Let $\alpha, \beta \in \{0,1\}^3$ be of odd Hamming distance. In time $O(m^2)$, we can determine the edge weights of a* Sum *constraint C such that $(\forall y)\, (x_1[y], x_2[y], x_3[y]) \notin \{\alpha, \beta\}$ holds if and only if (x_1, x_2, x_3) satisfies C.*

Proof. Let $\alpha = (\alpha_1, \alpha_2, \alpha_3)$ and $\beta = (\beta_1, \beta_2, \beta_3)$. Without loss of generality, we may assume that, for some $i^* \in \{1, 3\}$, $\alpha_i \neq \beta_i$ for all $i \leq i^*$ and $\alpha_i = \beta_i$ for all $i > i^*$.

As argued before, any triple (x_1, x_2, x_3) satisfies $(\forall y)\, (x_1[y], x_2[y], x_3[y]) \notin \{\alpha, \beta\}$ if and only if (x_1, x_2, x_3) satisfies $\|x_1^{\alpha_1} \wedge x_2^{\alpha_2} \wedge x_3^{\alpha_3}\| = \|x_1^{\beta_1} \wedge x_2^{\beta_2} \wedge x_3^{\beta_3}\| = 0$. Since both sides of the left equation are always non-negative, we can equivalently demand that their sum be zero. This condition simplifies to:

$$\begin{aligned}
0 &= \|x_1^{\alpha_1} \wedge x_2^{\alpha_2} \wedge x_3^{\alpha_3}\| + \|x_1^{\beta_1} \wedge x_2^{\beta_2} \wedge x_3^{\beta_3}\| \\
&= \|(\bigwedge_{i>i^*} x_i^{\alpha_i}) \wedge (\bigwedge_{i \leq i^*} x_i^{\alpha_i})\| + \|(\bigwedge_{i>i^*} x_i^{\alpha_i}) \wedge (\bigwedge_{i \leq i^*} \bar{x}_i^{\alpha_i})\| \\
&= \|(\bigwedge_{i>i^*} x_i^{\alpha_i}) \wedge (\bigwedge_{i \leq i^*} x_i^{\alpha_i})\| + \sum_{I \subseteq [i^*]} (-1)^{|I|} \|(\bigwedge_{i>i^*} x_i^{\alpha_i}) \wedge (\bigwedge_{i \in I} x_i^{\alpha_i})\| && \text{Observation 2} \\
&= \sum_{I \subsetneq [i^*]} (-1)^{|I|} \|(\bigwedge_{i>i^*} x_i^{\alpha_i}) \wedge (\bigwedge_{i \in I} x_i^{\alpha_i})\|. && i^* \text{ is odd}
\end{aligned}$$

Notice that (possibly after applying Observation 2 again to get rid of complemented occurrences) this expression is a weighted sum of values d, $\|x_1\|$, $\|x_2\|$, $\|x_3\|$, $\|x_1 \wedge x_2\|$, $\|x_1 \wedge x_3\|$ and $\|x_2 \wedge x_3\|$. In particular, the problematic term $\|x_1 \wedge x_2 \wedge x_3\|$ canceled in the equation above due to i^* being an odd number. In summary, we can express the above constraint by $0 = \sum_{I \subsetneq [3]} \lambda_I \| \bigwedge_{i \in I} x_i \|$ for some constants λ_I.

It is easy to compute the values $\| \bigwedge_{i \in I} x_i \|$ for all $I \subsetneq [3]$: $\| \bigwedge_{i \in \emptyset} x_i \| = d$ is a constant and each number $\|x_i\|$ is obtained by once enumerating all edges between X_i and Y. Any value $\|x_i \wedge x_j\|$ is determined in time $O(m^2)$ by iterating over all pairs of edges between X_i and Y, and between X_j and Y, respectively. Finally, we annotate each edge of the Sum constraint accordingly. To this end, each edge (x_i, x_j) is labeled with the contribution of

$\|x_i \wedge x_j\|$ and additionally, we distribute the node-weights $\|x_1\|$, $\|x_2\|$ and $\|x_3\|$ to the edges:

$$w(x_1, x_2) := \lambda_{\{1,2\}}\|x_1 \wedge x_2\| + \lambda_{\{1\}}\|x_1\|,$$
$$w(x_2, x_3) := \lambda_{\{2,3\}}\|x_2 \wedge x_3\| + \lambda_{\{2\}}\|x_2\|,$$
$$w(x_3, x_1) := \lambda_{\{3,1\}}\|x_3 \wedge x_1\| + \lambda_{\{3\}}\|x_3\|.$$

The target t is set to $-\lambda_\emptyset d$.

It is left to show that $\sum_{e\in E} |w(e)| \leq O(m^2)$ as in the definition of Sum. It is clear that $\|x_1\|$, $\|x_2\|$ and $\|x_3\|$ only account for $O(m)$. Each edge (x_i, y) contributes to at most m values $\|x_i \wedge x_j\|$, thus $\sum_{x_i,x_j} \|x_i \wedge x_j\| \leq O(m^2)$. The values λ_I are fixed constants depending only on α, β. ◀

4.2.3 Combined Algorithm for $k = 3$

By combining the previous reductions from VP(ϕ) to Constrained Triangle problems, we find the desired algorithm for 3-variable Vector Problems of hardness at most 2:

▶ **Lemma 14.** *Let Φ be a set of 3-variable formulas of hardness $H(\Phi) \leq 2$. Then* VP(Φ) *is decidable in time $O(m^{3-\varepsilon})$ for some $\varepsilon > 0$ using fast matrix multiplication.*

Proof. Let $\phi \in \Phi$. Given an VP(ϕ) instance, we can obtain an equivalent Constrained Triangle formulation by covering each pair of falsifying assignments by an Equal or Sum constraint using Lemmas 12 and 13 (any pair of distinct 3-variable assignments is either of Hamming distance 2 or of odd Hamming distance).

The hybrid problem VP(Φ) is now solved by rephrasing each subproblem VP(ϕ), $\phi \in \Phi$, as a Constrained Triangle problem Triangle$[C_1^\phi] \cdots [C_{r_\phi}^\phi]$ with $C_i^\phi \in \{\text{Equal}, \text{Sum}\}$, and by "stacking" all constraints. Since $\Phi = \{\phi_1, \ldots, \phi_\ell\}$ is of constant size, we are left to solve an instance of Triangle$[C_1^{\phi_1}] \cdots [C_{r_{\phi_1}}^{\phi_1}] \cdots [C_1^{\phi_\ell}] \cdots [C_{r_{\phi_\ell}}^{\phi_\ell}]$. Lemma 8 yields an $O(m^{3-\varepsilon})$-time algorithm for some $\varepsilon > 0$ for any constant number of adjunct Equal and Sum constraints. ◀

4.3 Algorithms for Arbitrary k

For large k, we will tackle Vector Problems by first brute-forcing over a number of variables before solving the remaining $k' = 2$ or $k' = 3$ problem. To this end, we establish the following lemmas in order to lift fast algorithms from fewer quantifiers to more quantifiers:

▶ **Lemma 15.** *Let ϕ be a k-variable formula. Then, for any $k' \leq k$, there exists some $I' \in \binom{[k]}{k-k'}$ such that for any I'-restriction ϕ' of ϕ we have $H(\phi') \leq H(\phi)$.*

Proof. Let $h = H(\phi)$. The statement is clear for $k' \leq h$, so suppose $h < k' \leq k$. From $H(\phi) < h + 1$, it follows by definition that there exists some $I \in \binom{[k]}{k-h-1}$ such that any I-restriction ϕ' of ϕ has not exactly one falsifying assignment. Choose an arbitrary $I' \in \binom{I}{k-k'}$ and let ϕ' be any I'-restriction of ϕ. Then any $(I \setminus I')$-restriction of ϕ' has not exactly one falsifying assignment. Hence, $H(\phi') < k' - |I \setminus I'| = k' - (k - h - 1) + (k - k') = h + 1$ and thus $H(\phi') \leq h = H(\phi)$. ◀

▶ **Lemma 16** (Lifting). *Let $k' \leq k$, let ϕ be a k-variable formula and let Φ' contain all k'-variable formulas ϕ' of hardness $H(\phi') \leq H(\phi)$. If* VP(Φ') *is decidable in time $T(m)$, then* VP(ϕ) *is decidable in time $O(m^{k-k'}T(m))$.*

Proof. The idea is to appropriately brute-force over $k - k'$ variables $x_1, \ldots, x_{k-k'}$ and use the $\mathrm{VP}(\Phi')$ algorithm to decide the remaining instance. Here, it is crucial that we can indeed solve the *Hybrid* Vector Problem! This is because we have to specialize ϕ to the values of the brute-forced vectors $(x_1, \ldots, x_{k-k'})$, however, in general $(x_1[y], \ldots, x_{k-k'}[y])$ takes different values for different y's, so we have to instantiate ϕ "per dimension". This leads to queries allowing several formulas ϕ_y – exactly as permitted by Hybrid Vector Problems.

More formally, by Lemma 15, there exists some index set I' of size $k - k'$, such that any I'-restriction ϕ' of ϕ satisfies $H(\phi') \leq H(\phi)$. By interchanging the order of the existential variables, we can assume that $I' = \{1, \ldots, k - k'\}$. Our first step is to enumerate all $n^{k'-k}$ choices for the first $k - k'$ variables $(x_1, \ldots, x_{k-k'})$; fix such a tuple $(x_1, \ldots, x_{k-k'})$. It remains to solve the problem

$$(\exists x_{k-k'+1}) \ldots (\exists x_k)\, (\forall y)\, \phi_y(x_{k-k'+1}[y], \ldots, x_k[y]),$$

where ϕ_y denotes the I'-restriction of $\phi(x_1[y], \ldots, x_k[y])$ in which all occurrences of $x_1[y], \ldots, x_{k-k'}[y]$ are fixed as specified by the brute-forced vectors $x_1, \ldots, x_{k-k'}$. In other words, we are left to solve a Hybrid Vector Problem over the formula set $\bigcup_y \{\phi_y\}$. Recall that ϕ_y is an I'-restriction of ϕ, and therefore, as guaranteed by Lemma 15, $H(\phi_y) \leq H(\phi)$. Thus $\phi_y \in \Phi'$, by the definition of Φ'. By the assumption that we can solve $\mathrm{VP}(\Phi')$ in time $T(m)$, the total running time is $O(n^{k-k'} T(m))$, which is bounded by $O(m^{k-k'} T(m))$. ◀

By the Lifting Lemma, we conclude the following consequences of Lemma 11 and Lemma 14. This concludes the algorithmic part of Theorem 6.

▶ **Corollary 17.** *Let ϕ be a k-variable formula of hardness $H(\phi) \leq 1$. Then $\mathrm{VP}(\phi)$ is decidable in time $O(m^{k-1})$.*

▶ **Corollary 18.** *Let ϕ be a k-variable formula of hardness $H(\phi) \leq 2 < k$. Then $\mathrm{VP}(\phi)$ is decidable in time $O(m^{k-\varepsilon})$ for some $\varepsilon > 0$ using fast matrix multiplication.*

5 Hardness Results

In this section, we prove the hardness part of Theorem 6, see Lemmas 19 and 20, below.

Let ϕ be a k-variable formula with exactly one falsifying assignment, or equivalently, let ϕ be of hardness $H(\phi) = k$. The model-checking query $(\exists x_1) \ldots (\exists x_k)\, (\forall y)\, \phi(x_1[y], \ldots, x_k[y])$ – that is, $\mathrm{VP}(\phi)$ – is equivalent to the k-Orthogonal Vectors problem with respect to polynomial improvements over the $O(m^k)$-time baseline algorithm as shown by Gao et al. [41] (the authors of [41] refer to Vector Problems of hardness k as *Basic Problems*). This shows the k-OV hardness part of Theorem 6:

▶ **Lemma 19** ([41, Lemma 1.1, Lemma 5.1])**.** *Let $H(\phi) = k$. If $\mathrm{VP}(\phi)$ is decidable in time $O(m^{k-\varepsilon})$ for some $\varepsilon > 0$, then k-OV is decidable in time $O(n^{k-\varepsilon'} \operatorname{poly} d)$ for some $\varepsilon' > 0$.*

It remains to show the HyperClique hardness. Recall that given a k-partite h-uniform hypergraph $(V_1 \uplus \ldots \uplus V_k, E)$, the h-uniform k-HyperClique problem asks to find vertices $(v_1, \ldots, v_k) \in V_1 \times \ldots \times V_k$, so that any h vertices in $\{v_1, \ldots, v_k\}$ form a hyperedge in E.

▶ **Lemma 20.** *Let ϕ be a k-variable formula and let $2 \leq h \leq H(\phi)$. If $\mathrm{VP}(\phi)$ is decidable in time $O(m^{k-\varepsilon})$ for some $\varepsilon > 0$, then the h-uniform HyperClique hypothesis fails.*

For $h \geq 3$, Lemma 20 shows the hardness part of Theorem 6. Moreover, the special case $h = 2$ is also informative: The 2-uniform k-HyperClique problem is simply the k-Clique problems on ordinary graphs. It is well-known that k-Clique can be solved in time $O(n^{\frac{\omega k}{3}})$

using fast matrix multiplication. However, no combinatorial solution which is significantly faster than the $O(n^k)$-time exhaustive search approach is known. The combinatorial k-Clique hypothesis speculates that no polynomial improvement is possible without relying on algebraic methods [3], which in turn, confirms the need to fall back on fast matrix multiplication in Lemma 14.

Proof. We will fix $k' = k'(k,h,\varepsilon)$ later such that k divides k'. Given an h-uniform k'-HyperClique instance G', we can assume that G' is k'-partite by copying the vertex set k' times and by keeping only edges spanned between distinct copies. So let $V(G') = V'_{1,1} \uplus \ldots \uplus V'_{1,k'/k} \uplus \ldots \uplus V'_{k,1} \uplus \ldots \uplus V'_{k,k'/k}$. We proceed in a split-and-list fashion: First, we split $V(G')$ into the k parts $V'_i := V'_{i,1} \uplus \ldots \uplus V'_{i,k'/k}$. Then, for $i = 1, \ldots, k$, we let $X_i \subseteq V'_{i,1} \times \ldots \times V'_{i,k'/k}$ be all tuples of vertices that form an h-uniform clique in G'. We refer to the elements of X_i as *bundles*. Let $Y \subseteq \binom{V(G')}{h}$ contain all non-edges of G'. We say a vertex bundle $x_i \in X_i$ *avoids* a non-edge $y \in Y$ if not all vertices in $y \cap V'_i$ are contained in x_i.

Our next step is to assign the entries $x_i[y]$, for all $x_i \in X_i$ and $y \in Y$. Let $y = \{v_1, \ldots, v_h\} \in Y$ and collect all indices $J = \{j : v_i \in V'_j \text{ for some } i \in [h]\}$. Since ϕ is of hardness $H(\phi) \geq h$, it holds for all index sets I of size at least $k - h$ that there exists an I-restriction ϕ' of ϕ having exactly one falsifying assignment. Picking $I = [k] \setminus J$, there exists an I-restriction $\phi' = \phi|_\alpha$, for some $\alpha : I \to \{0,1\}$, such that ϕ' is falsified only by a single assignment $\beta : J \to \{0,1\}$. For all $i \in I$ and all vectors $x_i \in X_i$, we define $x_i[y] := \alpha_i$. For all $j \in J$ and all vectors $x_j \in X_j$, we define $x_j[y] := \bar{\beta}_j$ if x_j avoids y, and $x_j[y] := \beta_j$ otherwise.

We claim that there exists a tuple of vectors $(x_1, \ldots, x_k)$ with $(\forall y \in Y)\, \phi(x_1[y], \ldots, x_k[y])$ if and only if G' contains a k'-hyperclique. Suppose there exists some k'-hyperclique $(v_{1,1}, \ldots, v_{1,k'/k}, \ldots, v_{k,1}, \ldots, v_{k,k'/k})$ in G'. Since each tuple $(v_{i,1}, \ldots, v_{i,k'/k})$ by itself forms a hyperclique, we have that $x_i := (v_{i,1}, \ldots, v_{i,k'/k}) \in X_i$. We aim to prove that $(x_1, \ldots, x_k)$ satisfies $\phi(x_1[y], \ldots, x_k[y])$ for all $y \in Y$. Let $y \in Y$ be arbitrary, and let J, I, α and β as above. After plugging in all values $x_i[y] = \alpha_i$ for $i \in I$, there remains only one falsifying assignment of $\phi(x_1[y], \ldots, x_k[y])$, given by β. Since we started from a hyperclique, x_j must avoid y for some $j \in J$, and thus $x_j = \bar{\beta}_j$ for some $j \in J$. The vectors $(x_1, \ldots, x_k)$ thus satisfy $\phi(x_1[y], \ldots, x_k[y])$, for any y. The converse argument is essentially symmetric. This finishes the correctness proof.

Finally, we turn to the running time analysis. Observe that $|X_i| \leq O(n^{k'/k})$ for all i and $|Y| \leq O(n^h)$. Furthermore, constructing the X_i's and Y takes time $O(n^{k'/k})$. Assigning the entries $x_i[y]$ takes time $O(\sum_i |X_i| \cdot |Y|) = O(n^{k'/k+h})$. In particular, it follows that the number of edges is bounded by $m \leq O(n^{k'/k+h})$. By assumption, solving $\mathrm{VP}(\phi)$ is in time $O(m^{k-\varepsilon})$ for some $\varepsilon > 0$. Hence, in total we solve h-uniform k'-HyperClique in time $O(n^{(k'/k+h)(k-\varepsilon)})$. By setting $k' \geq hk^2/\varepsilon$ this is bounded by $O(n^{k'-h\varepsilon})$, contradicting the h-uniform HyperClique hypothesis. ◀

We remark that for $h = k$, Lemma 20 can also be derived in an alternative way: Abboud et al. [5] show that the k-OV hypothesis is implied by the h-uniform HyperClique hypothesis for all $h \geq 3$. So all we need to do is combine Lemma 19 with that reduction:

▶ **Lemma 21** ([5]). *If k-OV is decidable in time $O(n^{k-\varepsilon} \operatorname{poly} d)$ for some $\varepsilon > 0$, then, for any h, the h-uniform HyperClique hypothesis fails.*

This finishes the proof of Theorem 6.

6 Equivalence of Vector Problems and $\exists^k\forall$ Graph Properties

This section is devoted to proving that Vector Problems $\mathrm{VP}(\phi)$ capture the core difficulty of model-checking $\exists^k\forall$ graph properties. Specifically, we will prove Theorem 7, which together with Theorem 6 proves Theorems 1 and 3.

▶ **Theorem 7** (Restated). *Let $\psi = (\exists x_1)\dots(\exists x_k)\,(\forall y)\,\phi(x_1,\dots,x_k,y)$ be an $\exists^k\forall$ graph property and let ϕ_0 denote the formula ϕ after substituting each predicate $E(x_i,x_j)$ by false.*

- *If $\mathrm{MC}(\psi)$ is decidable in time $T(m)$, then $\mathrm{VP}(\phi_0)$ is decidable in time $O(T(m))$.*
- *If $\mathrm{VP}(\phi_0)$ is decidable in time $O(m^{k-\varepsilon})$ for some $\varepsilon > 0$, then $\mathrm{MC}(\psi)$ is decidable in time $O(m^{k-\varepsilon'})$ for some $\varepsilon' > 0$.*

Recall that $\exists^k\forall$ graph properties are strictly more general than Vector Problems in the sense that graph properties are sensitive to edges $E(x_i,x_j)$. We will continue to interchangeably interpret the x_i's as vertices and vectors depending on the context. For the remainder of this section, let ψ, ϕ and ϕ_0 be as in Theorem 7.

We start by demonstrating the following preliminary lemma:

▶ **Lemma 22.** *Given a set of two-element subsets $\emptyset \neq I \subseteq \binom{[k]}{2}$, we say that vertices $(x_1,\dots,x_k)$* respect *I if for any $i<j$ we have $E(x_i,x_j)$ if and only if $\{i,j\} \in I$. We can compute the values $\|\bigwedge_{i=1}^{k} x_i\|$ for all vectors $(x_1,\dots,x_k)$ respecting I in time $O(m^{k-\frac{1}{2}})$.*

Proof. Since I is non-empty, there exists at least some element $\{i,j\} \in I$. We start by brute-forcing over all $k-2$ vectors x_ℓ, $\ell \notin \{i,j\}$. As in the statement, we only care about the combinations that respect I. In the same manner, we only keep the vertices x_i, x_j consistent with I. In particular, for all x_i, x_j overcoming this step, (x_i,x_j) indeed forms an edge. It is enough to compute $\|x_i \wedge x_j\|$ for these (x_i,x_j) in time $O(m^{\frac{3}{2}})$, because thereby we achieve a total running time of $O(n^{k-2}m^{\frac{3}{2}}) = O(m^{k-\frac{1}{2}})$. In other words, we are left to count, for each pair (x_i,x_j), the number of triangles containing (x_i,x_j) in the remaining graph with partitions X_i, X_j and Y. For the sake of completeness, we proceed to sketch the well-known procedure of counting triangles in sparse tripartite graphs [12].

We call a vertex *heavy* if it is of degree at least m^δ (where δ is yet to be fixed), and *light* otherwise. All light vertices can be eliminated as follows: Enumerate all edges $\{u,v\}$ and if v is light, then further iterate over all edges $\{v,w\}$ such that u, v and w stem from different partitions. Remember each triangle found in that manner and remove all light vertices afterwards. This step accounts for $O(m \cdot m^\delta) = O(m^{1+\delta})$ time.

The remaining graph consists only of heavy vertices, and, since there are only m edges, at most $O(m/m^\delta) = O(m^{1-\delta})$ such. We may apply the naïve algorithm by explicitly considering each triple of vertices. This step takes time $O(m^{3(1-\delta)})$, so in total the algorithm runs in time $O(m^{1+\delta} + m^{3(1-\delta)})$. The claim follows by setting $\delta := \frac{1}{2}$.

We remark that using fast matrix multiplication to solve the instance including only heavy vertices, we can achieve a slightly faster algorithm running in total time $O(m^{k-2+\frac{2\omega}{\omega+1}})$. ◀

Proof of Theorem 7. The first part is easy to see: The Vector Problem $\mathrm{VP}(\phi_0)$ constitutes a special case of model-checking ψ.

So let us focus on the second part. For a set $I \subseteq \binom{[k]}{2}$, let ϕ_I denote ϕ after substituting each predicate $E(x_i,x_j)$ by *true* if $\{i,j\} \in I$ and by *false* otherwise. In particular, $\phi_\emptyset = \phi_0$. Furthermore, we define

$$\psi_I := (\exists x_1)\dots(\exists x_k)\left(\left(\bigwedge_{\{i,j\}\in\binom{[k]}{2}} E(x_i,x_j) \leftrightarrow \big[\{i,j\} \in I\big]\right) \wedge (\forall y)\,\phi_I(x_1[y],\dots,x_k[y])\right).$$

Then clearly $\psi = \bigvee_{I \subseteq \binom{[k]}{2}} \psi_I$. So we can check the satisfiability of ψ by separately model-checking all properties ψ_I.

We claim that model-checking ψ_I for any $I \neq \emptyset$ is in time $O(m^{k-\frac{1}{2}})$: Observation 3 shows how to test, for a fixed tuple $(x_1, \dots, x_k)$ and given the values $\| \bigwedge_{j \in J} x_j \|$ for all $J \subseteq [k]$, whether it holds that $(\forall y)\, \phi_I(x_1[y], \dots, x_k[y])$. We thus start by precomputing all values $\| \bigwedge_{j \in J} x_j \|$ for all sets $J \subsetneq [k]$. To this end, enumerate all $|J|$-tuples of edges between X_j and Y, $j \in J$; this step takes time $O(m^{k-1})$. Using Lemma 22 we can further precompute $\| \bigwedge_{j \in [k]} x_j \|$ for all vertices $(x_1, \dots, x_k)$ respecting I in time $O(m^{k-\frac{1}{2}})$. Together, we now know $\| \bigwedge_{j \in J} x_j \|$ for all J and all vertices $(x_1, \dots, x_k)$ respecting I. Hence, it suffices to enumerate all vertices $(x_1, \dots, x_k)$ respecting I and to test as in Observation 3 whether $(\forall y)\, \phi_I(x_1[y], \dots, x_k[y])$ holds (this check takes constant time). Since there are at most $O(m^{k-1})$ many tuples $(x_1, \dots, x_k)$ respecting I, the running time is dominated by $O(m^{k-\frac{1}{2}})$.

It remains to find a model-checking procedure for $\psi_\emptyset$, i.e., for $I = \emptyset$. Note that $\mathrm{MC}(\psi_\emptyset)$ is not directly equivalent to $\mathrm{VP}(\phi_\emptyset)$, since an arbitrary solution $(x_1, \dots, x_k)$ of $\phi_\emptyset$ does not necessarily meet the condition that none of the edges (x_i, x_j) are present. The following reduction enforces this constraint.

Consider a given $\mathrm{MC}(\psi_\emptyset)$ instance G over the vertex partitions $X_1, \dots, X_k$ and Y and let $\delta > 0$ be a parameter to be fixed later. We call a vertex x_i *heavy* if it is of degree $\geq m^\delta$, and *light* otherwise. The first step is to eliminate all heavy vertices; there can exist at most $O(m/m^\delta) = O(m^{1-\delta})$ many such vertices. By interchanging the order of the existential quantifiers, we can always assume that x_1 is heavy and solve the remaining problem over $X_2, \dots, X_k$ in time $O(m^{k-1})$ using the model-checking baseline algorithm. If a solution $(x_1, \dots, x_k)$ is found in that manner, we accept. It thus takes time $O(m^{k-\delta})$ to safely remove all heavy vertices.

Next, partition each set X_i into several groups $X_{i,1}, \dots, X_{i,g}$ such that the total degree of all vectors is bounded by $m^\delta \leq \sum_{x_i \in X_{i,j}} \deg(x_i) \leq 2m^\delta$, for all groups $X_{i,j}$, except for possibly the last non-empty groups. This is implemented by greedily inserting vectors into $X_{i,j}$ until its total degree exceeds m^δ. As each vector inserted in that way is light, we can overshoot by at most m^δ. It follows that $g \leq O(m/m^\delta) = O(m^{1-\delta})$.

We assume that $\mathrm{VP}(\phi_0) = \mathrm{VP}(\phi_\emptyset)$ is decidable in time $O(m^{k-\varepsilon})$ for some $\varepsilon > 0$. Then we continue as follows:

1. For all combinations $(j_1, \dots, j_k) \in [g]^k$, solve the Vector Problem $\mathrm{VP}(\phi_\emptyset)$ with input $X_{1,j_1}, \dots, X_{k,j_k}$. If we find a solution, we call $(j_1, \dots, j_k)$ a *successful* combination.
2. If there are more than m^{k-1} successful combinations, we accept.
3. Otherwise, for any successful combination $(j_1, \dots, j_k)$, solve $\mathrm{MC}(\psi_\emptyset)$ using the baseline algorithm on $X_{1,j_1}, \dots, X_{k,j_k}$ and accept iff one of these invocations accepted.

We claim the above algorithm is correct. First of all, any solution $(x_1, \dots, x_k)$ of $\mathrm{MC}(\psi_\emptyset)$ is also a solution of $\mathrm{VP}(\phi_\emptyset)$. It is therefore safe to only consider those subinstances in step 3 for which we received a positive output in step 1. It remains to argue why step 2 is correct. How many tuples $(x_1, \dots, x_k)$ can be solutions of $\mathrm{VP}(\phi_\emptyset)$ but not of $\mathrm{MC}(\psi_\emptyset)$? At most $mn^{k-2} \leq m^{k-1}$, since at least one edge (x_i, x_j) must exist for any such tuple. Thus, if we witness $> m^{k-1}$ solutions of $\mathrm{VP}(\phi_\emptyset)$, among these there exists at least one solution of $\mathrm{MC}(\psi_\emptyset)$ by guarantee.

Finally, let us bound the running time of the above algorithm. Recall that removing heavy vertices accounts for $O(m^{k-\delta})$ time. In step 1, the $\mathrm{VP}(\phi_\emptyset)$ algorithm is applied $g^k = O(m^{k-\delta k})$ times on instances of size $O(m^\delta)$, which takes time $O(m^{k-\delta k+\delta(k-\varepsilon)}) = O(m^{k-\delta\varepsilon})$. Step 3 becomes relevant only if there are at most m^{k-1} successful combinations. For any

such combination, the model-checking baseline algorithm takes time $O((m^\delta)^k) = O(m^{\delta k})$. In total, our running time is $O(m^{k-\delta} + m^{k-\delta\varepsilon} + m^{k-1+\delta k})$. By picking δ so that $0 < \delta < \frac{1}{k}$, the claim follows.

This finishes the proof of Theorem 7, and thus completes the proof of our main result. ◀

7 Extensions and Outlook

Beyond our results of Theorems 1 and 3, we discuss several natural directions for extensions, present first results along these lines and give open problems for future work. In particular, we extend our results to a counting dichotomy and investigate the optimal exponent of low-complexity properties.

7.1 Determining the Optimal Exponent for Low-Complexity Properties

For the optimal exponent c_ψ for any $\exists^k\forall$-quantified graph property ψ, Theorems 1 and 3 establish either a (conditionally) tight value of $c_\psi = k$ or an upper bound of $c_\psi < k$. This begs the question: Can we obtain the (conditionally) exact value on c_ψ also in the latter case?

As an interesting exemplary case, we study Vector Problems $(\exists x_1 \in X_1)\,(\exists x_2 \in X_2)\,(\exists x_3 \in X_3)\,(\forall y \in Y)\,\phi(x_1[y], x_2[y], x_3[y])$ where ϕ is symmetric, i.e., $\phi(x_1, x_2, x_3)$ is invariant under interchanging the variables' order. Equivalently, ϕ is symmetric if its output depends only on the number of 1-inputs. We therefore identify a 3-variable symmetric formula with its *symmetric type*, a zero-based length-4 string $t \in \{\mathbf{0}, \mathbf{1}\}^4$ where $t_i = \mathbf{1}$ exactly if ϕ holds true on all inputs of i 1's and $(3 - i)$ 0's.

For symmetric formulas ϕ, we find a more immediate criterion to read off the hardness $H(\phi)$. Namely, $H(\phi)$ equals the maximum number h, such that $\mathbf{1}^h\mathbf{0}$ or $\mathbf{01}^h$ is a substring[6] of ϕ's symmetric type (and $H(\phi) = 0$ if neither constitutes a substring).

▶ **Theorem 7.** *Let $\phi(x_1, x_2, x_3)$ be symmetric. The complexity of* $\mathrm{VP}(\phi)$ *is as stated in Table 1.*

We prove Theorem 7 in the full version of this paper. As detailed there (and illustrated by Table 1), already in this exemplary case some interesting gaps remain and offer potential starting points for future work.

7.2 Counting Classification

For first-order graph properties with our quantifier structure $\exists^k\forall$, it is natural to ask if we can *count* the number of its witnesses. Specifically, for a given property $\psi = (\exists x_1 \in X_1)\ \ldots\ (\exists x_k \in X_k)\,(\forall y \in Y)\,\phi(x_1, \ldots, x_k, y)$, we might ask to output the number of tuples $(x_1, \ldots, x_k)$ such that $(\forall y)\,\phi(x_1, \ldots, x_k, y)$ holds (instead of merely detecting existence of at least one such tuple) – we call this problem the counting model-checking problem $\#\mathrm{MC}(\psi)$. Especially in the context of database queries, one often would like to report this number or even enumerate all such tuples. Indeed, related work [36] considers such questions for more general quantifier structures, but under other restrictions on the formulas and when the running time is measured in terms of the number of objects n rather than m.

Note that the analogous question for Boolean constraint satisfaction properties is resolved: For every Boolean CSP, we can either count the number of solutions in polynomial-time, or this task is #P-complete [33]. In our case, this dichotomy is a surprisingly simple consequence of our techniques. In particular, we achieve the same dichotomy as for the decision version.

[6] A contiguous sequence of characters within ϕ's symmetric type

Table 1 Lists all symmetric 3-variable formulas ϕ and the complexities of the respective Vector Problems VP(ϕ).

hardness $H(\phi)$	*symmetric type of* ϕ	*upper bound*	*lower bound*
0	**0000**	trivial	trivial
0	**1111**	trivial	trivial
1	**0001**, **1000**	$O(m)$	$\Omega(m)$
1	**0010**, **0100**	$O(m^2)$	$\Omega(m)$
1	**0101**, **1010**	$O(m^2)$	$m^{2-o(1)}$ under 3-XOR
1	**1001**	$O(m)$	$\Omega(m)$
2	**0011**, **1100**	$O(m^\omega)$	$m^{\omega-o(1)}$ under k-Clique
2	**0110**	$\tilde{O}(m^{\frac{3+\omega}{2}})$	$m^{\omega-o(1)}$ under k-Clique
2	**1011**, **1101**	$O(m^{\frac{9+\omega}{4}})$	$m^{\omega-o(1)}$ under k-Clique
3	**0111**, **1110**	$m^3/2^{\Omega(\sqrt{\log m})}$	$m^{3-o(1)}$ under 3-OV

▶ **Theorem 7.** *Let ψ be an $\exists^k\forall$ graph property of hardness $h = H(\psi)$.*

- *If $h \leq 1$, then $\#\mathrm{MC}(\psi)$ is solvable in time $O(m^{k-\varepsilon})$ for some $\varepsilon > 0$ combinatorially.*
- *If $h \leq 2 < k$, then $\#\mathrm{MC}(\psi)$ is solvable in time $O(m^{k-\varepsilon})$ for some $\varepsilon > 0$ using fast matrix multiplication. (Furthermore, $\#\mathrm{MC}(\psi)$ cannot be solved by a combinatorial $O(m^{k-\varepsilon})$-time algorithm unless the combinatorial k-Clique hypothesis is false.)*
- *If $3 \leq h \leq k$, then $\#\mathrm{MC}(\psi)$ cannot be solved in time $O(m^{k-\varepsilon})$ for any $\varepsilon > 0$ unless the h-uniform HyperClique hypothesis fails.*
- *If $h = k$, then $\#\mathrm{MC}(\psi)$ cannot be solved in time $O(m^{k-\varepsilon})$ for any $\varepsilon > 0$ unless the k-OV hypothesis fails.*

We defer the proof of this theorem to the full version of this paper.

7.3 Open Problems

Among the natural remaining challenges is, first and foremost, the following: Can we generalize or strengthen our dichotomy to larger fragments of first-order properties?

Specifically, one direction is to extend our results beyond graph properties: Allowing more than a single predicate or allowing a single predicate of higher arity already yields further very expressive classes. While our algorithmic techniques conveniently generalize, unfortunately, they do not yet seem to be sufficient to establish a complete dichotomy.

A second direction is to investigate other quantifier structures than $\exists^k\forall$ (and the equivalent $\forall^k\exists$). Note that establishing such a dichotomy might require different plausible hardness assumptions than the ones used in this work – in particular, it follows from the work of Carmosino et al. [28] that any such hardness assumption must have a lower nondeterministic *and* co-nondeterministic complexity than its deterministic complexity.

Finally, it is interesting to explore whether Proposition 4 can be strengthened: In particular, assume that for some hardness level $h \geq 3$, there is an algorithm that allows us to detect a k-clique in h-uniform hypergraphs in time $O(n^{k-\varepsilon})$ for all $k \geq h+1$. Can we then solve *all* hardness-h $\exists^k\forall$ graph properties in time $O(m^{k-\varepsilon'})$?

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 *Proceedings of the 2017 IEEE 58th Annual Symposium on Foundations of Computer Science*, FOCS '17, pages 192–203, 2017. doi:10.1109/FOCS.2017.26.
2. Amir Abboud, Arturs Backurs, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Or Zamir. Subtree Isomorphism Revisited. *ACM Trans. Algorithms*, 14(3):27:1–27:23, 2018. doi:10.1145/3093239.
3. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the Current Clique Algorithms Are Optimal, So is Valiant's Parser. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, FOCS '15, pages 98–117, 2015. doi:10.1109/FOCS.2015.16.
4. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight Hardness Results for LCS and Other Sequence Similarity Measures. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, FOCS '15, pages 59–78, 2015. doi:10.1109/FOCS.2015.14.
5. Amir Abboud, Karl Bringmann, Holger Dell, and Jesper Nederlof. More Consequences of Falsifying SETH and the Orthogonal Vectors Conjecture. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 253–266. ACM, 2018. doi:10.1145/3188745.3188938.
6. Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and Fixed Parameter Subquadratic Algorithms for Radius and Diameter in Sparse Graphs. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 377–391, 2016. doi:10.1137/1.9781611974331.ch28.
7. Amir Abboud, Ryan Williams, and Huacheng Yu. More Applications of the Polynomial Method to Algorithm Design. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 218–230, 2015. URL: http://dl.acm.org/citation.cfm?id=2722129.2722146.
8. Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of Faster Alignment of Sequences. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming*, pages 39–51. Springer Berlin Heidelberg, 2014.
9. Serge Abiteboul, Richard Hull, and Victor Vianu, editors. *Foundations of Databases: The Logical Level*. Addison-Wesley Longman Publishing Co., Inc., 1st edition, 1995.
10. Alfred V. Aho, Yehoshua Sagiv, and Jeffrey D. Ullman. Equivalences Among Relational Expressions. *SIAM J. Comput.*, 8(2):218–246, 1979. doi:10.1137/0208017.
11. Josh Alman, Timothy M. Chan, and R. Ryan Williams. Polynomial Representations of Threshold Functions and Algorithmic Applications. In *Proceedings of the 2016 IEEE 57th Annual Symposium on Foundations of Computer Science*, FOCS '16, pages 467–476, 2016. doi:10.1109/FOCS.2016.57.
12. Noga Alon, Raphael Yuster, and Uri Zwick. Finding and Counting Given Length Cycles. *Algorithmica*, 17(3):209–223, 1997. doi:10.1007/BF02523189.
13. Arturs Backurs and Piotr Indyk. Edit Distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the 47th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2015, pages 51–58. ACM, 2015. doi:10.1145/2746539.2746612.
14. Arturs Backurs and Piotr Indyk. Which Regular Expression Patterns Are Hard to Match? In *Proceedings of the 2016 IEEE 57th Annual Symposium on Foundations of Computer Science*, FOCS '16, pages 457–466, 2016. doi:10.1109/FOCS.2016.56.
15. Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Towards tight approximation bounds for graph diameter and eccentricities. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 267–280, 2018. doi:10.1145/3188745.3188950.

16 Guillaume Bagan, Arnaud Durand, and Etienne Grandjean. On Acyclic Conjunctive Queries and Constant Delay Enumeration. In Jacques Duparc and Thomas A. Henzinger, editors, *Computer Science Logic*, pages 208–222. Springer Berlin Heidelberg, 2007.

17 Christoph Berkholz, Jens Keppeler, and Nicole Schweikardt. Answering Conjunctive Queries Under Updates. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '17, pages 303–318. ACM, 2017. doi:10.1145/3034786.3034789.

18 Karl Bringmann. Why Walking the Dog Takes Time: Fréchet Distance Has No Strongly Subquadratic Algorithms Unless SETH Fails. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, FOCS '14, pages 661–670, 2014. doi:10.1109/FOCS.2014.76.

19 Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. A Dichotomy for Regular Expression Membership Testing. In *Proceedings of the 2017 IEEE 58th Annual Symposium on Foundations of Computer Science*, FOCS '17, pages 307–318, 2017. doi:10.1109/FOCS.2017.36.

20 Karl Bringmann and Marvin Künnemann. Quadratic Conditional Lower Bounds for String Problems and Dynamic Time Warping. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, FOCS '15, pages 79–97, 2015. doi:10.1109/FOCS.2015.15.

21 Karl Bringmann and Marvin Künnemann. Multivariate Fine-Grained Complexity of Longest Common Subsequence. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 1216–1235, 2018. doi:10.1137/1.9781611975031.79.

22 Andrei A. Bulatov. A Dichotomy Theorem for Constraints on a Three-Element Set. In *Proceedings of the 2002 IEEE 43rd Annual Symposium on Foundations of Computer Science*, FOCS '02, pages 649–658, 2002. doi:10.1109/SFCS.2002.1181990.

23 Andrei A. Bulatov. Tractable conservative Constraint Satisfaction Problems. In *18th IEEE Symposium on Logic in Computer Science*, page 321, 2003. doi:10.1109/LICS.2003.1210072.

24 Andrei A. Bulatov. A Dichotomy Theorem for Nonuniform CSPs. In *Proceedings of the 2017 IEEE 58th Annual Symposium on Foundations of Computer Science*, FOCS '17, pages 319–330, 2017. doi:10.1109/FOCS.2017.37.

25 Andrei A. Bulatov. Constraint satisfaction problems: complexity and algorithms. *SIGLOG News*, 5(4):4–24, 2018. doi:10.1145/3292048.3292050.

26 Andrei A. Bulatov, Peter Jeavons, and Andrei A. Krokhin. Classifying the Complexity of Constraints Using Finite Algebras. *SIAM J. Comput.*, 34(3):720–742, 2005. doi:10.1137/S0097539700376676.

27 Nofar Carmeli and Markus Kröll. Enumeration Complexity of Conjunctive Queries with Functional Dependencies. In *ICDT*, volume 98 of *LIPIcs*, pages 11:1–11:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

28 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic Extensions of the Strong Exponential Time Hypothesis and Consequences for Non-reducibility. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 261–270, 2016. doi:10.1145/2840728.2840746.

29 Timothy M. Chan and Ryan Williams. Deterministic APSP, Orthogonal Vectors, and More: Quickly Derandomizing Razborov-Smolensky. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 1246–1255, 2016. URL: http://dl.acm.org/citation.cfm?id=2884435.2884522.

30 Ashok K. Chandra and David Harel. Structure and Complexity of Relational Queries. *J. Comput. Syst. Sci.*, 25(1):99–128, 1982. doi:10.1016/0022-0000(82)90012-5.

31 Ashok K. Chandra and Philip M. Merlin. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In *Proceedings of the 9th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 1977, pages 77–90, 1977. doi:10.1145/800105.803397.

32 Hubie Chen. A rendezvous of logic, complexity, and algebra. *SIGACT News*, 37(4):85–114, 2006. doi:10.1145/1189056.1189076.

33 Nadia Creignou and Miki Hermann. Complexity of Generalized Satisfiability Counting Problems. *Inf. Comput.*, 125(1):1–12, 1996. doi:10.1006/inco.1996.0016.

34 Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of Boolean Constraint Satisfaction problems*. SIAM, 2001. doi:10.1137/1.9780898718546.

35 Víctor Dalmau. Some dichotomy theorems on constant free Boolean formulas. Technical Report LSI-97-43-R, Departament LSI, Universitat Politècnica de Catalunya, 1997.

36 Holger Dell, Marc Roth, and Philip Wellnitz. Counting Answers to Existential Questions. *arXiv e-prints*, February 2019. arXiv:1902.04960.

37 Rodney G. Downey, Michael R. Fellows, and Udayan Taylor. The Parameterized Complexity of Relational Database Queries and an Improved Characterization of W[1]. In *First Conference of the Centre for Discrete Mathematics and Theoretical Computer Science*, pages 194–213, 1996.

38 Tomás Feder and Moshe Y. Vardi. Monotone monadic SNP and constraint satisfaction. In *Proceedings of the 25th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 1993, pages 612–622, 1993. doi:10.1145/167088.167245.

39 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.

40 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. doi:10.1007/3-540-29953-X.

41 Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and Ryan Williams. Completeness for First-order Properties on Sparse Structures with Algorithmic Applications. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 2162–2181, 2017. URL: http://dl.acm.org/citation.cfm?id=3039686.3039827.

42 Michel Habib, Christophe Paul, and Laurent Viennot. A Synthesis on Partition Refinement: A Useful Routine for Strings, Graphs, Boolean Matrices and Automata. In *STACS 98, 15th Annual Symposium on Theoretical Aspects of Computer Science*, pages 25–38, 1998. doi:10.1007/BFb0028546.

43 Pavol Hell and Jaroslav Nesetril. On the complexity of H-coloring. *J. Comb. Theory, Ser. B*, 48(1):92–110, 1990. doi:10.1016/0095-8956(90)90132-J.

44 Neil Immerman. Upper and Lower Bounds for First Order Expressibility. *J. Comput. Syst. Sci.*, 25(1):76–98, 1982. doi:10.1016/0022-0000(82)90011-3.

45 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

46 Peter Jeavons. On the Algebraic Structure of Combinatorial Problems. *Theor. Comput. Sci.*, 200(1-2):185–204, 1998. doi:10.1016/S0304-3975(97)00230-2.

47 Peter Jeavons, David A. Cohen, and Marc Gyssens. Closure properties of constraints. *J. ACM*, 44(4):527–548, 1997. doi:10.1145/263867.263489.

48 Robert Krauthgamer and Ohad Trabelsi. Conditional Lower Bounds for All-Pairs Max-Flow. *ACM Trans. Algorithms*, 14(4):42:1–42:15, 2018. doi:10.1145/3212510.

49 Andrea Lincoln, Virginia Vassilevska Williams, and Ryan Williams. Tight Hardness for Shortest Cycles and Paths in Sparse Graphs. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 1236–1252, 2018. URL: http://dl.acm.org/citation.cfm?id=3174304.3175350.

50 Christos H. Papadimitriou and Mihalis Yannakakis. On the Complexity of Database Queries. *J. Comput. Syst. Sci.*, 58(3):407–427, 1999. doi:10.1006/jcss.1999.1626.

51 Mihai Patrascu and Ryan Williams. On the Possibility of Faster SAT Algorithms. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1065–1075, 2010. doi:10.1137/1.9781611973075.86.

52 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the 45th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2013, pages 515–524, 2013. doi:10.1145/2488608.2488673.

53 Thomas J. Schaefer. The Complexity of Satisfiability Problems. In *Proceedings of the 10th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 1978, pages 216–226, 1978. doi:10.1145/800133.804350.

54 Moshe Y. Vardi. The Complexity of Relational Query Languages (Extended Abstract). In *Proceedings of the 14th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 1982, pages 137–146, 1982. doi:10.1145/800070.802186.

55 Moshe Y. Vardi. On the Complexity of Bounded-Variable Queries. In *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 266–276, 1995. doi:10.1145/212433.212474.

56 Virginia Vassilevska and Ryan Williams. Finding, Minimizing, and Counting Weighted Subgraphs. In *Proceedings of the 41st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2009, pages 455–464. ACM, 2009. doi:10.1145/1536414.1536477.

57 Ryan Williams. A New Algorithm for Optimal 2-constraint Satisfaction and Its Implications. *Theor. Comput. Sci.*, 348(2):357–365, December 2005. doi:10.1016/j.tcs.2005.09.023.

58 Ryan Williams. Faster decision of first-order graph properties. In *Joint Meeting of the 23rd EACSL Annual Conference on Computer Science Logic (CSL) and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 80:1–80:6, 2014. doi:10.1145/2603088.2603121.

59 Ryan Williams. On the Difference Between Closest, Furthest, and Orthogonal Pairs: Nearly-Linear vs Barely-Subquadratic Complexity. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 1207–1215, 2018. doi:10.1137/1.9781611975031.78.

60 Dmitriy Zhuk. A Proof of CSP Dichotomy Conjecture. In *Proceedings of the 2017 IEEE 58th Annual Symposium on Foundations of Computer Science*, FOCS '17, pages 331–342, 2017. doi:10.1109/FOCS.2017.38.