Algorithmica (2022) 84:3156–3191
https://doi.org/10.1007/s00453-022-01014-x

The Fine-Grained Complexity of Multi-Dimensional Ordering Properties

Haozhe An[1] · Mohit Gurumukhani[2] · Russell Impagliazzo[3] · Michael Jaber[4] · Marvin Künnemann[5] · Maria Paula Parga Nina[3]

Received: 15 December 2021 / Accepted: 16 July 2022 / Published online: 22 August 2022

Abstract

We define a class of problems whose input is an n-sized set of d-dimensional vectors, and where the problem is first-order definable using comparisons between coordinates. This class captures a wide variety of tasks, such as complex types of orthogonal range search, model-checking first-order properties on geometric intersection graphs, and elementary questions on multidimensional data like verifying Pareto optimality of a choice of data points. Focusing on constant dimension d, we show that any such k-quantifier, d-dimensional problem is solvable in $O(n^{k-1}\log^{d-1} n)$ time. Furthermore, this algorithm is conditionally tight up to subpolynomial factors: we show that assuming the 3-uniform hyperclique hypothesis, there is a k-quantifier, $(3k-3)$-dimensional problem in this class that requires time $\Omega(n^{k-1-o(1)})$. Towards identifying a single representative problem for this class, we study the existence of complete problems for the 3-quantifier setting (since 2-quantifier problems can already be solved in near-linear time $O(n\log^{d-1} n)$, and k-quantifier problems with $k > 3$ reduce to the 3-quantifier case). We define a problem Vector Concatenated Non-Domination VCND_d (Given three sets of vectors X, Y and Z of dimension d, d and $2d$, respectively, is there an $x \in X$ and a $y \in Y$ so that their concatenation $x \circ y$ is not dominated by any $z \in Z$, where vector u is dominated by vector v if $u_i \leq v_i$ for each coordinate $1 \leq i \leq d$), and determine it as the "unique" candidate to be complete for this class (under fine-grained assumptions).

Keywords Fine-grained complexity · First-order logic · Orthogonal vectors

Haozhe An, Mohit Gurumukhani, Michael Jaber and Maria Paula Parga Nina: This research was conducted while the authors were undergraduates at UC San Diego.

✉ Mohit Gurumukhani
mgurumuk@cs.cornell.edu

Extended author information available on the last page of the article

2012 ACM Subject Classification Theory of computation: Complexity theory and logic · Theory of computation: Problems, reductions and completeness · Theory of computation: Verification by model checking

1 Introduction

Algorithmic problems based on comparing elements according to a total ordering relation are as fundamental as they are useful. Any introductory algorithms textbook starts with sorting and other comparison-based problems. For higher dimensional data, problems involving comparisons for multiple components, such as range queries, are equally fundamental in computational geometry. In databases, queries need to handle data with many fields that can be compared (beyond other relations on the data), such as listing all employees who are not managers of another employee, with seniority in one range and salary in another.

In this paper, we give a general, systematic study of the complexity of multi-dimensional comparison problems. We define complexity classes capturing the notion of "multi-dimensional comparison problems", as appropriate in geometry and in databases, with the classes PTO_d representing geometric problems in d dimensional data, and TO_d representing problems that combine ordering and other relations for such data, as would be found in databases. We then identify the maximum complexity of problems in these classes under standard assumptions in fine-grained complexity, and relate the classes to each other and other studied complexity classes. For many subclasses, we find natural complete or hard problems where progress on better algorithms for these problems would result in better algorithms for the entire subclass.

While our results are varied, with upper bounds, conditional lower bounds and completeness results, a consistent theme emerges. Our classes are intermediate between two previously studied classes of logically defined problems, first-order in the sparse representation (e.g., graph problems in adjacency list format) and first-order in the dense representation (e.g, graph problems in adjacency matrix format). While orderings are dense relations, with quadratically many pairs for which they hold, they are a special case that can be represented succinctly, by giving an array of ranks for each element. What emerges in our results is that multi-dimensional ordering problems are very tightly connected to first-order in the sparse representation, and not directly connected to the dense representation. Thus, while they give substantially different settings, we give many senses in which sparse relations can be coded in terms of orders, and where orderings can be reduced to sparse relations.

1.1 A Class of Geometric Ordering Problems: $PTO_{k,d}$

As an example for multi-dimensional comparison problems, consider 2D orthogonal range searching: given a set of 2-dimensional data points D, answer Boolean queries of the form

$$\exists x \in D : x \in [\ell_1, u_1] \times [\ell_2, u_2],$$

where $[\ell_1, u_1] \times [\ell_2, u_2]$ is a given *orthogonal range*. Note that here, we may without loss of generality replace each point's coordinate in dimension d by its *rank* among the coordinates in dimension d of all points in D. Typical variants include to report, count or optimize over all elements in the query range. A long line of research starting in the 70s, including [11, 19, 23, 41, 44, 46], gives fast algorithms for such tasks, e.g., an algorithm to preprocess D such as to answer queries in time $O(\log \log n)$ using space $O(n \log \log n)$, see [19]. Many more complex algorithmic tasks can be solved using orthogonal range techniques, see [8, 26] for an overview.

Also more complex tasks than mere orthogonal range searching arise naturally: In a set of d-dimensional data points D, consider a *feature* (or property) F of the data points that can be described as being contained in an orthogonal range $[\ell_1, u_1] \times \cdots \times [\ell_d, u_d]$. Given a family $\mathcal{F}$ of such features, there are several natural questions to ask:

- decide if all features are present in the dataset:
  $\forall F = [\ell_1, u_1] \times \cdots \times [\ell_d, u_d] \in \mathcal{F} \;\; \exists x \in D : \;\; x \in F$
- decide if some data point displays all features:
  $\exists x \in D \;\; \forall F = [\ell_1, u_1] \times \cdots \times [\ell_d, u_d] \in \mathcal{F} : \;\; x \in F$
- decide if two different features are equivalent on D:
  $\exists F_1 \in \mathcal{F} \;\; \exists F_2 \in \mathcal{F} \;\; \forall x \in D : \;\; F_1 \neq F_2 \wedge (x \in F_1 \leftrightarrow x \in F_2)$.

Some of these questions can be quickly answered using orthogonal range reporting queries, for others it seems that already the output size of single such query might pose a possibly unnecessary bottleneck. Furthermore, some features might be comparison-based, but more complex than a simple orthogonal range, e.g.,[1]

$$\begin{aligned} & x \in F(\ell_1, u_1, \ldots, \ell_d, u_d) \\ & \quad \Longleftrightarrow \; (x_1 \in [\ell_1, u_1] \rightarrow (x_2 \in [\ell_2, u_2])) \wedge (x_1 \notin [\ell_1, u_1] \\ & \quad \rightarrow (x_3, \ldots, x_d) \in [\ell_3, u_3] \times \cdots \times [\ell_d, u_d])). \end{aligned}$$

In such cases, it would not be immediate whether orthogonal range search techniques can be used at all.

We formalize a notion of "multi-dimensional comparison problems" by introducing a class of problems $PTO_{k,d}$ (for "purely total ordering property") of model-checking a k-quantifier first-order property on a relational structure with d total ordering relations (each succinctly represented as a sorted list of objects) as well as unary relations (to enable comparison of coordinates with constants). In particular, this class contains any property ψ of the form

$$\psi = Q_1 x^{(1)} Q_2 x^{(2)} \ldots Q_k x^{(k)} : \phi(x^{(1)}, \ldots, x^{(k)}),$$

where $Q_i \in \{\exists, \forall\}$, $x^{(i)}$ ranges over d-dimensional vectors (which we also call *objects*), and ϕ is an arbitrary Boolean formula involving only comparisons of the form $x_i^{(a)} \leq x_i^{(b)}$ with $1 \leq a, b \leq k$ (here, $x_i^{(a)}, x_i^{(b)}$ denotes the i-th dimension of $x^{(a)}, x^{(b)}$, respectively), as well as comparisons with constants. We will refer to d as the *dimension*

[1] The given expression could model the following feature: if a person is of working age ($x_1 \in [\ell_1, u_1]$), use criterion $x_2 \in [\ell_2, u_2]$, otherwise use $(x_3, \ldots, x_d) \in [\ell_3, u_3] \times \cdots \times [\ell_d, u_d]$.

of a formula $\psi \in PTO_{k,d}$. For this paper throughout, we think of ϕ as fixed formula, and thus k, d are constants. See Sect. 2.1 for further details.

The class $PTO_{k,d}$ includes all problems as mentioned above, but also tasks such as verifying Pareto optimality of a given set of d-dimensional data points[2], or given a set of d-dimensional geometric objects, determine whether there are k distinct such objects whose bounding boxes intersect.

We furthermore extend this class to $TO_{k,d}$, where we allow, beyond d total ordering relations, also arbitrary additional relations (represented explicitly). These two classes encompass in particular the following types of problems:

- Model-checking first-order properties of **geometric intersection graphs**: Presence of an edge in an intersection graph of axis-parallel boxes can be decided using comparisons of coordinates. Thus, any k-quantifier first-order property on such geometric intersection graphs in $\mathbb{R}^d$ can be formulated as a problem in $PTO_{k,d}$, such as finding k pair-wise non-intersecting d-dimensional axis-parallel unit-cubes [42].[3]
- **temporal logic**: using a single total ordering relation, we may represent *precedence* in a time domain. Thus, we may express temporal logical statements involving expressions over future or past events in $TO_{k,1}$.
- relational databases with **ordered types**: in relational databases, we may use totally ordered data types (salaries of employees, time events, rank in a sorted list, etc.) as succinct representation to enable comparisons. In this context, studying the complexity of a problem in $TO_{k,d}$ corresponds to studying the *data complexity* of a fixed query.

1.2 Our Results

Let $k \geq 2$. We show that any problem in $PTO_{k,d}$ involving n objects can be solved in time $O(n^{k-1}\log^{d-1} n)$ which is $\tilde{O}(n^{k-1})$ for any constant dimension d. We extend this algorithm to run in time $O(m^{k-1}\log^{d-1} m)$ for sentences in $TO_{k,d}$, where m denotes the sum of the number of objects and the size of the additional relations, i.e., the number of tuples contained in the relation. We show the matching conditional lower bound that there is some sentence in $PTO_{k,3k-3}$ that requires time $\Omega(n^{k-1-o(1)})$ under the 3-uniform hyperclique hypothesis [2, 15, 37, 40] – this hypothesis postulates that $n^{k\pm o(1)}$ running time is essentially best possible for finding cliques in hypergraphs. (See Sect. 2.2 for further details.)

Beyond these general upper and lower bounds, we also seek to identify *hard* or even *complete* problems for this class. Such problems capture the full generality of these classes, in the sense that finding a significantly improved algorithm for this problem would give an improved algorithm for all problems in the class. We use the following fine-grained notion of hardness/completeness: Formally, let P be a problem whose best known algorithm runs in time $T_P(n)$ and let C be a class of problems whose best

[2] Recall that a set X is Pareto optimal if there are no distinct points $x, x' \in X$ such that x is coordinate-wise at least as large as x'.

[3] For even more involved types of algorithmic tasks beyond k-quantifier first-order properties, see, e.g., [20] (All-Pairs Shortest Paths) or [25] (NP-hard problems).

known algorithms runs in time $T_C(n)$. We say P is *hard* for a class of problems C, if any $T_P(n)^{1-\epsilon}$-time algorithm for P with $\epsilon > 0$ gives a $T_C(n)^{1-\epsilon'}$-time algorithm for all problems in C for some $\epsilon' > 0$. We say that P is *complete* for C, if it is hard for C and contained in C. In particular, if P is complete for C, then P admits substantial improvements over time $T_P(n)$ if and only if *all* problems in C admit substantial improvements over $T_C(n)$. We use fine grained reductions to show such results. Refer to Sect. 2 for its formal definition.

We identify such problems for specific quantifier structures. In particular, we focus on the 3-quantifier case, since all 2-quantifier $O(1)$-dimensional total order properties can be solved in near-linear time $\tilde{O}(n)$ (Theorem 3), and all k-quantifier properties with $k > 3$ can be reduced to the 3-quantifier case via brute forcing (Corollary 5). Focusing on $PTO_{k,d}$, we obtain the following results (see Table 1):

1. For existentially quantified pure total ordering properties (denoted by $PTO_{\exists\exists\exists,d}$), we give an $\tilde{O}(n^{2\omega/(\omega+1)}) = \tilde{O}(n^{1.407})$ time algorithm and identify the well-studied triangle detection in sparse graphs as a complete problem[4].
2. For the quantifier structure $\forall\exists\exists$, we also give an $\tilde{O}(n^{2\omega/(\omega+1)}) = \tilde{O}(n^{1.407})$ time algorithm by showing that the problem of *counting*, for each edge in a sparse graph, the number of triangles containing this edge is *hard* for the class $PTO_{\forall\exists\exists,d}$. Since we reduce to a counting problem rather than a member of this class, we do not obtain a completeness result, however.
3. For the quantifier structure $\exists\forall\exists$, we were unable to find a complete or hard problem. Nevertheless, we give evidence that this quantifier structure does not contain a complete problem for $PTO_{k,d}$ by showing that all $PTO_{\exists\forall\exists,d}$ problems have a $\tilde{O}(n)$-time nondeterministic and co-nondeterministic algorithm. Since we also show a $n^{2-o(1)}$ SETH[5]-based lower bound for $PTO_{3,d}$ when $d \to \infty$, this rules out existence of such a complete problem using deterministic reductions under NSETH, a nondeterministic variant of SETH [17]. We also give a conditional lower bound of $n^{2-o(1)}$ under the Hitting Set conjecture.
4. Finally, for the seemingly most difficult quantifier structure of $\exists\exists\forall$, we show $n^{2-o(1)}$-time conditional lower bounds under SETH and the 3-uniform hyperclique hypothesis, and identify the following complete problem for $PTO_{\exists\exists\forall,d}$, which we call Vector Concatenated Non-Domination VCND_d: Given three sets of vectors X, Y and Z of dimension d, d and $2d$, respectively, is there an $x \in X$ and a $y \in Y$ so that their concatenation $x \circ y$ is not dominated by any $z \in Z$, where vector u is dominated by vector v if $u_i \leq v_i$ for each coordinate $1 \leq i \leq d$.

Note that this covers all quantifier structures for $k = 3$, as deciding $Q_1 Q_2 Q_3 \phi$ with $Q_i \in \{\exists, \forall\}$ is equivalent to deciding $\overline{Q_1}\,\overline{Q_2}\,\overline{Q_3}\,\overline{\phi}$ where $\overline{\forall} = \exists$, $\overline{\exists} = \forall$ and $\overline{\phi}$ is the negation of ϕ.

These results identify the VCND_d problem as the essentially *only* candidate (up to fine-grained equivalence) to be complete for $PTO_{3,d}$ under NSETH: It is complete for $\exists\exists\forall$, and all problems with a different 3-quantifier structure have either

[4] Strictly speaking, we identify the following 3-dimensional problem (which is linear-time equivalent to triangle detection in sparse graphs) as complete for $PTO_{\exists\exists\exists,d}$: $\exists x, y, z : x_1 = z_1 \wedge x_2 = y_2 \wedge y_3 = z_3$.

[5] Strong Exponential Time Hypothesis (SETH) for CNF-SAT: For all $\epsilon > 0$, there exists a k so that k-CNF-SAT cannot be solved in time $O(2^{n(1-\epsilon)})$ [34].

improved deterministic or (co-)nondeterministic algorithms, and thus cannot be complete without major consequences in fine-grained complexity. It remains a challenge to prove or disprove completeness of VCND_d for $PTO_{3,d}$ (beyond its completeness for $PTO_{\exists\exists\forall,d}$).

Since the above results motivate VCND_d as a central problem for $PTO_{k,d}$, we work towards algorithmic improvements for this problem. In particular, we obtain an $\tilde{O}(n^{2-\frac{1}{2^d}})$-time algorithm for VCND whenever vectors in X have dimension 2 and vectors in Y have dimension d (or vice versa). Note that obtaining such an $O(n^{2-\epsilon(d)})$ time algorithm with $\epsilon(d) > 0$ for general VCND_d would refute the 3-uniform hyperclique hypothesis by our conditional lower bound and completeness result.

Finally, we show that our algorithmic results extend to the class $TO_{k,d}$ (see Sect. 3 for details), while all hardness results trivially apply, since they are already proven for the subclass $PTO_{k,d}$. Generally speaking, this shows that the database setting (with additional sparse relations) does not increase the fine-grained complexity compared to the geometric setting of purely total ordering properties.

1.3 Previous Work

This work continues a relatively new direction, fine-grained complexity of complexity classes. *Fine-grained complexity* aims to not only qualitatively classify problems as "easy" or "hard", but (to the extent possible) pin-point their exact complexities. We now have a wide collection of standard algorithmic problems where any significant improvements in algorithmic running time would refute one or more conjectures about well-studied problems, such as the k-SUM problem [27], All Pairs Shortest Paths [3, 40, 49], SAT [34, 45], or Orthogonal Vectors [1, 2, 6, 9, 12–14, 16, 38, 43]. Recent work in fine-grained complexity has gone from considering problems one at a time to following traditional complexity in considering *classes* of problems. Fine-grained reductions often cut across the usual complexity classes (with reductions from NP-complete problems to first-order properties, for example), but on the other hand, fine-grained complexity distinguishes between problems with the same traditional complexities (e.g., two different NP-complete problems might have very different properties in fine-grained complexity). Nevertheless, there are now a number of classes of problems, grouped by logical structure or common format, whose fine-grained complexity is at least partially understood: dense first-order properties [48]; sparse first-order properties [15, 17, 30]; several extensions of first-order [29]; and certain formats of dynamic programming problems [28, 38].

The most closely related previous work to our results are [30, 48]. Both of these papers consider the class of *first-order definable properties*, the first for the dense case (where each relation is given as a matrix, aka adjacency matrix format), and the second for the sparse case (where the input is given as a list of tuples in the relations, e.g., for graphs, adjacency list format). This class is natural both in terms of computational complexity, where it is the uniform version of AC_0 ([31]), and in database theory, because these are the queries expressible in basic SQL [7]. First-order logic can also express many polynomial time computable problems: Orthogonal Vectors, k-Orthogonal Vectors, k-Clique, k-Independent Set, k-Dominating Set, etc. Not only

were the likely complexities of the hardest problems (as a function of number of quantifiers) given, but in the second paper, a natural complete problem was identified, the Orthogonal Vectors problem (OV). The conclusion was that there were substantial improvements possible in the worst-case complexity of model checking for first-order properties if and only if the known algorithms for Orthogonal Vectors can be substantially improved. Using a recent sub-polynomial improvement in OV algorithms by [4, 21], they obtained a similar improvement in model checking for every first-order property. [29] extends this work to related logics such as transitive closure logics, first-order logic on totally ordered sets, and first-order logic with function symbols. They show that model checking for first-order logic with a single total ordering is actually equivalent to that for unordered structures under fine-grained reductions. In contrast, we show that for even two orderings, the model checking problem becomes substantially harder, meaning we require new techniques to characterize the complexity of problems on multi-dimensional data.

There is also work on classes of problems that are related in spirit, but do not form a well-studied complexity class. V.-Williams and Williams [49] study problems related to shortest paths in graphs, and shows that many are subcubic-time equivalent. Künnemann et al. [38] study dynamic programming problems with a similar structure and give a unified treatment of their fine-grained complexities. Gao [28] extends this class of dynamic programming problems from lines to tree-like structures such as bounded treewidth graphs.

2 Preliminaries

The following notion of *fine-grained reductions* was introduced in [49].

Definition 1 *(Fine-grained reduction)* Let $(\Pi_1, T_1(m)) \leq_{\text{FGR}} (\Pi_2, T_2(m))$ denote that for every $\epsilon > 0$ there is a $\delta > 0$ and a Turing reduction from Π_1 to Π_2 so that the time for the reduction (not counting oracle calls) is $O(T_1(m)^{1-\delta})$ and $\sum_q (T_2(|q|))^{1-\epsilon} = O(T_1(m)^{1-\delta})$, where the sum is over all oracle calls q made by the reduction on an instance of size m.

In other words, if there is some $\epsilon > 0$ such that problem Π_2 is in $\mathsf{TIME}((T_2(m))^{1-\epsilon})$, then problem Π_1 is in $\mathsf{TIME}((T_1(m))^{1-\delta})$ for some $\delta > 0$, i.e., if Π_2 can be solved substantially faster than T_2 then Π_1 can be solved substantially faster than T_1. If both T_1 and T_2 are $\Theta(m^2)$, the reduction is called a subquadratic reduction. We say that Π_1 and Π_2 are *fine-grained equivalent* if there is a fine-grained reduction from Π_1 to Π_2 and vice versa.

We use this notation not only on single problems but also on classes of problems. Let C_1 and C_2 be classes of problems. $(C_1, T_1(m)) \leq_{\text{FGR}} (C_2, T_2(m))$ if for all problems $\Pi_1 \in C_1$ there is a $\Pi_2 \in C_2$ so that $(\Pi_1, T_1(m))$ fine-grained reduces to $(\Pi_2, T_2(m))$.

2.1 Details on $PTO_{k,d}$ and $TO_{k,d}$

In this paper, we consider the fine-grained complexity of model checking problems definable in first-order logic on structures with d binary relations $x \leq_i y$, $1 \leq i \leq d$, where each binary relation is a total pre-order of the universe (i.e., transitive, reflexive, total, but not necessarily anti-symmetric.)

Total orders. We use $x \leq_i y$ to represent the i'th relation in our family holding between x and y. Such a relation is dense, holding for $\Theta(n^2)$ pairs of elements. However, we can represent such a representation succinctly, by giving an array which for each element specifies its rank in a list sorted by the ordering relation (with some elements having the same rank, if inequality holds in both directions). It is in this succinct format that ordering relations are described for our problems.

Equivalently, we may represent all ordering relations by representing each object x as a d-dimensional vector $(x_1, \ldots, x_d)$, where x_i denotes the rank of x in the i'th ordering relation. Thus, it is equivalent to write $x \leq_i y$ or $x_i \leq y_i$, and we will switch between these two based on which seems clearer for the given circumstance.

The vectors we get in this way are very special, in that the coordinates are always positive integers from 1 to n. However, also problems defined about d dimensional vectors over any totally ordered domain (such as $\mathbb{R}$) fall into our setting. This is because we will still have only n vectors in our setting from that domain and in $O(n \log n)$ time we can replace each x_i with its rank in the set of i'th coordinates of vectors.

Unary relations. We also allow unary relations, or, equivalently, comparisons to constants. More precisely, any unary relation U is represented as a list of objects for which U holds. Apart from allowing us to put objects into categories (sometimes called *colored* properties), this enables us to express comparisons of coordinates with constants: To express whether $x \leq_i \gamma$ for some constant γ, we introduce a unary relation symbol $U_i^{\leq \gamma}$ that holds for all x with $x_i \leq \gamma$. Thus from now on, it suffices to declare constants γ explicitly, and afterwards we may express arbitrary comparisons like $x_i \neq \gamma$ or $x_i > \gamma$. Note that since we always consider fixed formulas ψ, each considered property will use $O(1)$ constants for comparisons.

Definition of $PTO_{k,d}$. We denote the class of purely total ordering model-checking problems for first-order formulas in pre-orderings and unary relations specified as above where the formula has d distinct ordering relations and k total occurrences of quantifiers by $PTO_{k,d}$. PTO_k is the union of $PTO_{k,d}$ over all constants d. We can further divide PTO_k into 2^k sub-classes based on the quantifier structure, so for example $PTO_{\exists\exists\exists}$ is the sub-class of PTO_3 where the model-checking problems are for formulas of the form $\exists x \exists y \exists z \Phi(x, y, z)$ where Φ is quantifier-free. We let n be the size of the universe of the structure, which is also, up to constant factors, the size in terms of $O(\log n)$-bit words required to specify all total pre-orderings and unary relations. Algorithm time for problems in PTO is thus measured in terms of n. In this

format, it is a constant time operation to evaluate whether any relation is true or false for specified elements.

Definition of $TO_{k,d}$. We generalize $PTO_{k,d}$ to the class $TO_{k,d}$ by also allowing the formula and models to have any constant number of sparse relations of any constant arity. These are specified as lists of tuples where the relation holds. Let the problem size be denoted by m, which is equal to the sum of the number of elements n and the number of tuples.

We assume all algorithms start with quasi-linear time preprocessing steps to create data structures such as hash tables or binary search trees that allow fast determination (constant time or logarithmic time) of whether a relation holds for given elements, and allows one to list the tuples in a relation that contain a given element in at most poly-log time + poly-log time times the number of such tuples.

On the difference. $PTO_{k,d}$ is a more "geometric" class of problems, and so it is interesting when we can reduce combinatorial problems to this class. Therefore, we will focus on these classes when giving conditional hardness results. $TO_{k,d}$ is closer to the type of problems that might arise in applications such as database queries. Therefore, we will focus on $TO_{k,d}$ when giving algorithms or other upper bounds on complexity. Since $PTO_{k,d} \subseteq TO_{k,d}$, lower bounds for $PTO_{k,d}$ are stronger results, and upper bounds for $TO_{k,d}$ are stronger results.

Further examples of problems in $PTO_{k,d}$ To define further well-studied problems in $PTO_{k,d}$, we say that a vector u *dominates* a vector v if $u_i \geq v_i$ for all $1 \leq i \leq d$, and denote this by $u \geq_{dom} v$. Furthermore, given a set of d-dimensional real vectors A, we say that A is *Pareto optimal* if there are no distinct $a, a' \in A$ so that a is coordinate-wise at least as large as a'.

- Vector Domination Problem (see, e.g. [18, 32]): Given two sets of d-dimensional real vectors A and B, are there two vectors $u \in A$ and $v \in B$ such that $u \geq_{dom} v$? In small dimensions, this problems turns out to be equivalent to the low-dimensional Orthogonal Vectors problem by a recent result of Chan [18].
- Pareto Optimality Verification (see, e.g. [33]): Given a set A of vectors, determine if A is Pareto optimal.

From the definition, both problems are in $PTO_{2,d}$. As we will see, they can be solved in time $O(n \log^{d-1} n)$. For superconstant dimension d, [18, 32] give further improvements.

2.2 Conjectures From Fine-Grained Complexity

We list the fine-grained hardness assumptions used in this paper. While some of these assumptions imply others (or are implied by them), they might turn out to be very different: It is conceivable, e.g., that SETH turns out to be false, while the Orthogonal Vectors Conjecture might indeed hold. Hence, we aim to classify our conditional hardness results by the weakest hypothesis that suffices.

Specifically, we use the following hypotheses:

SAT hypotheses

- Strong Exponential Time Hypothesis (SETH) [34]: For all $\epsilon > 0$, there exists a k such that k-CNF-SAT cannot be solved in time $O(2^{n(1-\epsilon)})$.
- Nondeterministic Strong Exponential Time Hypothesis (NSETH) [17]: For every $\varepsilon > 0$, there exists a k so that k-TAUT is not in $\mathsf{NTIME}(2^{(1-\epsilon)n})$, where k-TAUT is the language of all k-DNF which are tautologies, i.e., always true.

OV hypotheses

The Orthogonal Vectors problem (OV) is defined as follows: Given sets A, B of vectors in $\{0,1\}^d$, the task is to determine whether there exists an *orthogonal pair* $a \in A, b \in B$, i.e., the inner product $\sum_{k=1}^{d} a[k] \cdot b[k]$ is equal to 0. While this problem is clearly solvable in time $O(n^2\mathsf{poly}(d))$, it conjectured that we cannot achieve strongly quadratic running time:

- Low-dimension OV conjecture (LDOVC), or Strong OV conjecture: For all $\epsilon > 0$, there is a C so that there is no $O(n^{2-\epsilon})$ time algorithm for OV with dimension $d = C \log n$. This is implied by SETH by [34] and [47].
- Moderate-dimension OV conjecture (MDOVC): For all $\epsilon > 0$, there is no $O(n^{2-\epsilon}\mathsf{poly}(d))$ time algorithm that solves OV with dimension d. This is trivially implied by LDOVC.
- Sparse OV conjecture (SOVC): For all $\epsilon > 0$, there is no $O(m^{2-\epsilon})$ time algorithm for OV where m is the total Hamming weight of all the input vectors [30]. This is equivalent to MDOVC [30].

Relevant for our work is also the correspondence of OV to model-checking first-order properties:

- First-order property conjecture (FOPC) [30]: There exists an integer $k \geq 2$ so that there is a $(k+1)$-quantifier first-order property that cannot be decided in time $O(m^{k-\epsilon})$ for any $\epsilon > 0$ (here structures are over universe size n and a list of constant arity relations over these structures is given. The total number of relations given is m). This is equivalent to MDOVC [30].

Finally, we also use the following generalization of OV to a problem with conjectured complexity $n^{k\pm o(1)}$: The k-Orthogonal Vectors (k-OV) problem asks to determine, given a set A of vectors in $\{0,1\}^d$ whether there are $a_1, \ldots, a_k \in A$ such that $\sum_{j=1}^{d} a_1[j] \cdots a_k[j] = 0$.

- Moderate-dimension k-OV conjecture: For all $\epsilon > 0$, there is no $O(n^{k-\epsilon}\mathsf{poly}(d))$ time algorithm that solves k-OV with dimension d. This is implied by SETH [47].

Hitting Set hypothesis

The Hitting Set problem is defined as follows: Given two families of subsets over the same universe U, is there a set in the first family that has non-empty intersection with each set in the second family? Equivalently, given two sets A, B of vectors in $\{0,1\}^d$, determine whether there is an element $a \in A$ such that for all $b \in B$ there is some $k \in [d]$ such that $a[k] = b[k] = 1$.

- Hitting set conjecture: For all $\epsilon > 0$, there is a C so that there is no $O(n^{2-\epsilon})$ time algorithm for Hitting Set with dimension $d = C \log n$. The Hitting Set conjecture

implies the Low-Dimension OV conjecture [5], but there are reasons to believe it is not implied by SETH ([17]).

Hyperclique hypothesis

- h-uniform k-HyperClique Hypothesis: Let $k > h > 2$ be integers. The h-uniform k-HyperClique Hypothesis states that for no $\epsilon > 0$, we can detect a k-clique in a h-uniform hypergraph on n nodes in time $O(n^{k-\epsilon})$, see [40] for a detailed discussion of its plausibility and [2, 15, 37, 40] for recent applications.

For all of these conjectures, complexity is measured in the word RAM model with $O(\log n)$ bit words.

2.3 The VCND Problem

We formally define the perhaps most important problem for $PTO_{k,d}$.

Definition 2 *(Vector Concatenated Non-Domination)* Given a set X of d_1-dimensional vectors, a set Y of d_2-dimensional vectors, and a set Z of (d_1+d_2)-dimensional vectors, all with entries in $\mathbb{Z}$, we define the language (d_1, d_2)-*Vector Concatenated Non-Domination* to be the decision problem asking if

$$\exists x \in X \exists y \in Y \forall z \in Z (x \circ y \not\leq_{dom} z),$$

where $x \circ y \in \mathbb{Z}^{d_1+d_2}$ denotes the concatenation of x and y. We denote by VCND_d the special case of $d_1 = d_2 = d$.

We can view VCND as first constructing the set $X \circ Y = \{x \circ y \mid x \in X, y \in Y\}$ and then asking whether there is some $z \in Z$ that dominates some element of $X \circ Y$. There are other operations which could replace concatenation, such as the coordinate-wise max operation $\mathrm{Max}(X, Y)$.

2.4 Relationships to Other Classes

For fine-grained complexity, the representation of the input is significant. In considering the complexity of dense first-order properties, we view the input as a matrix or tensor representing each relation; for binary relations, this is the familiar adjacency matrix representation for graphs. While the relations are not necessarily dense, the algorithms cannot assume or utilize sparsity. The "sparse" version of the same properties represents the input relations as lists of tuples where the relation holds, generalizing the adjacency list representation for graphs. The input is not necessarily sparse, but the algorithm is allowed more time for denser instances, so sparse instances are the most difficult ones. Total order relations are intermediate between "dense" and "sparse" relations, because while they are actually dense, containing a quadratic number of pairs, they can be succinctly represented by the sorted list. In particular, total orders can be obtained as the transitive closure operation performed on the sparse "successor" relation. So our hardness results also imply hardness for sparse first-order augmented by transitive closures, a class considered in [29].

3 Technical Overview

In this section, we give the main ideas for all of our results, see Table 1 for an overview. One of our main results is an upper bound on model-checking sentences in $PTO_{k,d}$ and $TO_{k,d}$.

Theorem 3 *There is an algorithm running in time $O(n \log^{d-1}(n))$ for model-checking a two-quantifier formula $Q_1 x Q_2 y \varphi(x, y)$ with d ordering relations and unary predicates.*

Specifically, we obtain this result using the following lemma, which we obtain by a reduction to orthogonal range counting.

Lemma 4 *Given a formula $\varphi(x, y)$ with d ordering relations and unary predicates and two sets X, Y of vectors in $\mathbb{R}^d$, there is an $O(n \log^{d-1}(n))$ time algorithm that returns an array A indexed by each $x \in X$ so that $A[x]$ is the number of $y \in Y$ such that $\varphi(x, y)$ is true.*

Combining the above theorem with exhaustive search over the first $k-2$ quantifiers yields

Corollary 5 *Model-checking formulas in $PTO_{k,d}$ is in TIME$(n^{k-1} \log^{d-1}(n))$.*

If we have additional explicitly represented relations, more work is required. For such cases, throughout the paper, we will always assume that these relations are *sparse*, i.e., the total input size is $m = O(n)$. In this case, we obtain the same asymptotic running time.

Theorem 6 *Model-checking formulas in $TO_{k,d}$ is in TIME$(m^{k-1} \log^{d-1}(m))$.*

The idea is to reduce the problem to the purely totally ordered case by assuming that all sparse relations are empty; using Lemma 4 for the 2-quantifier case, we can obtain for each x the number of y satisfying the condition. We then repair these counts to the true values by iterating over the additional sparse relations, similar to the baseline algorithm in [30].

We prove our baseline algorithms in Sect. 4. Note that in Sect. 3.4, we discuss a lower bound proving these baseline algorithms to be conditionally optimal under fine-grained hardness assumptions.

In the remainder of the section, we distinguish our results based on the quantifier structure. Since any k-quantifier formula with $k > 3$ reduces to the 3-quantifier setting via brute force over the first $k-3$ quantifiers, we only regard 3-quantifier structures.

3.1 Quantifier Structures Ending in ∃∃∃

Recall that informally, we call a problem *complete* for a class if it is contained in the class and model-checking any sentence in the class reduces to our problem. For sentences in $PTO_{k,d}$ ending in ∃∃∃, we show that detecting triangles in a sparse graph is complete for this class. By current running time bounds for the problem [10], we obtain a running time of $\tilde{O}(n^{2\omega/(\omega+1)}) = \tilde{O}(n^{1.407\ldots})$.

Table 1 Our results for $PTO_{k,d}$, where we assume that d is an arbitrarily large constant

Quantifier structure	3 quantifiers		k quantifiers, $k > 3$
$\ldots\exists\exists\forall$ (sym.: $\ldots\forall\forall\exists$) complete: VCND_d (Thm. 14)	$\tilde{O}(n^2)$	$n^{2-o(1)}$ for $d = 6$ (3-unif. $3k-3$-HC, Thm. 15) $n^{2-o(1)}$ for $d \to \infty$ (SETH, Thm. 16)	$\tilde{O}(n^{k-1})$ $n^{k-1-o(1)}$ for $d = 3k-3$ (3-unif. HC, Thm. 15)
$\ldots\exists\forall\exists$ (sym.: $\ldots\forall\exists\forall$) complete: open	$\tilde{O}(n^2)$	$n^{2-o(1)}$ for $d \to \infty$ (Hitting Set, Thm. 12)	$\tilde{O}(n^{k-1})$
	$\tilde{O}(n)$ (co-)nondet.		$\tilde{O}(n^{k-2})$ (co-)nondet. $n^{k-2-o(1)}$ for $d = 2$ (SETH, Thm. 13)
$\ldots\forall\exists\exists$ (sym.: $\ldots\exists\forall\forall$) complete: open hard: ETC (Thm. 9)	$\tilde{O}(n^{\frac{2\omega}{\omega+1}}) = O(n^{1.41})$		$\tilde{O}(n^{k-\frac{\omega+3}{\omega+1}}) = O(n^{k-1.59})$
$\ldots\exists\exists\exists$ (sym.: $\ldots\forall\forall\forall$) complete: triangle det. (Thm. 7)	$\tilde{O}(n^{\frac{2\omega}{\omega+1}}) = O(n^{1.41})$		$\tilde{O}(n^{k-\frac{\omega+3}{\omega+1}}) = O(n^{k-1.59})$

Theorem 7 *The triangle detection problem in sparse graphs is fine-grained equivalent to a problem that is complete for model-checking ∃∃∃ formulas with only ordering relations and unary relations.*

More precisely, the following ordering property is shown to be complete: $\exists x \exists y \exists z : x_1 = z_1 \wedge x_2 = y_2 \wedge y_3 = z_3$ which is easy to be seen equivalent to triangle detection in sparse graphs.

Intuitively, we reduce to this problem as follows: Given a formula $\exists x \exists y \exists z \phi(x, y, z)$, we can determine whether $\phi(x, y, z)$ holds once we know all comparisons between x, y, z in each dimension i. A challenge here is to reduce comparisons like $x_i < y_i$ to an equality check: Similar to a trick used in [50], we do this by guessing the highest-order bit of *divergence* between x_i and y_i to obtain a "proof" only involving equalities; since we may assume that $1 \leq x_i, y_i \leq n$ (by working in *rank* space), there are only $O(\log n)$ choices for a single comparison. The key observation is that the quantifier structure is sufficiently well behaved to make this reduction work: we only need to guess these bits of divergence for $O(d)$ many comparisons and can express correctness of all proofs for comparisons between x and z using equality on the first dimension, between x and y using the second dimension, and between y and z using the third dimension. In total, this results in an admissible blow-up of $\log^{O(d)} n$. We prove the result in Sect. 5.1.1.

We turn to the setting with additional sparse relations, i.e., formulas in $TO_{\exists\exists\exists,d}$. Here we establish the triangle *counting* problem in sparse graphs as hard for the class. Since the approach of [10] also gives a counting algorithm in the same running time as detection, we establish the same algorithmic upper bound.

Theorem 8 *Every problem in $TO_{\exists\exists\exists,d}$ reduces to the problem of counting the number of triangles in a sparse graph via reductions that preserve time up to polylog factors.*

Handling the additional sparse relations is highly non-trivial. In particular, to obtain our result, we first show that the triangle counting problem is hard for model-counting ∃∃∃ formulas in the sparse setting of [30], which is interesting in its own right. For the proof, we refer to Sect. 5.1.2.

Since triangle detection is a classical problem, improving the bound of $O(n^{1.407})$ for ∃∃∃ structures already in the purely total ordering case would be a major algorithmic result.

3.2 Quantifier Structures Ending in ∀∃∃

For quantifier structures ending in ∀∃∃, we obtain a hard problem: We show that every problem in $TO_{\forall\exists\exists,d}$ (and thus also $PTO_{\forall\exists\exists,d}$) reduces to that of determining, for each edge in a sparse graph, how many triangles contain this edge; we call this problem *Edgewise Triangle Counting (ETC)*. Again, currently the best algorithm for this problem is essentially the same as that for triangle detection and counting [10].

Theorem 9 *Edgewise Triangle Counting is hard for model-checking $TO_{\forall\exists\exists,d}$ formulas.*

Since the high-level arguments for this result substantially build on the completeness result for $TO_{\exists\exists\exists,d}$ given in the previous section, we defer a discussion of the techniques to Sect. 5.1.3, where we give the proof.

3.3 Quantifier Structures Ending in ∃∀∃

For the quantifier structure of ∃∀∃, we are unable to establish a complete problem. However, this quantifier structure admits (co-)nondeterministic algorithms that are faster than the baseline algorithm.

Theorem 10 *Model-checking formulas in $PTO_{k,d}$ ending in ∃∀∃ can be done in nondeterministic and co-nondeterministic time $O(n^{k-2}\log^{d-1}(n))$.*

The main idea is as follows: Consider any $\exists x\forall y Qz\phi(x, y, z)$ property. For the nondeterministic algorithm, we simply (nondeterministically) guess x and solve the remaining 2-quantifier problem $\forall y Qz\phi(x, y, z)$ in time $O(n\log^{d-1} n)$ using the baseline algorithm. For the co-nondeterministic algorithm, we need to verify that $\forall x\exists y\overline{Q}z\overline{\phi}(x, y, z)$. Here, for every x, we (nondeterministically) guess a witness y_x and solve the remaining $\overline{Q}z\overline{\phi}(x, y_x, z)$ formula using the approach of Theorem 3.

For the case of total ordering properties with additional sparse relations, this approach is not directly applicable: If, e.g., all guessed witnesses y_x happen to participate in many tuples of the sparse relations, we have to repeatedly solve problems with a large input size. We remedy this problem by taking care of such large degree witness y_x explicitly; while this incurs a certain slow-down, we can limit it to a factor of $O(\sqrt{n})$.

Theorem 11 *Model-checking formulas in $TO_{k,d}$ ending in ∃∀∃ can be done in nondeterministic and co-nondeterministic time $O(m^{k-3/2}\log^{d-1}(m))$.*

We prove the above (co-)nondeterministic algorithms in Sect. 5.2.

As a consequence of the above nondeterministic algorithms, assuming NSETH [17], we cannot establish hardness beyond $n^{k-2-o(1)}$ for $PTO_{\exists\forall\exists,d}$ using deterministic SETH-based reductions. However, by reducing from a problem with low (co-)nondeterministic complexity, specifically, the Hitting Set conjecture [5], we can give a conditional lower bound already for $PTO_{\exists\forall\exists,d}$ (as $d \to \infty$) that matches our baseline algorithm.

Theorem 12 *Assuming the Hitting Set conjecture, for all $\epsilon > 0$, there exists some d such that model checking formulas in $PTO_{\exists\forall\exists,d}$ requires time $\Omega(n^{2-\epsilon})$.*

The proof of this result is reminiscent to some reductions in [24] and is given in Sect. 6. We reduce from Hitting Set (given sets of vectors $A, B \subseteq \{0, 1\}^{c\log n}$ for arbitrary c, determine whether some $a \in A$ is non-orthogonal to all $b \in B$) to a formula $\exists x\forall y\exists z\psi(x, y, z)$ as follows: We think of x ranging over vectors $a \in A$, y ranging over $b \in B$, and think of z as a "proof" of the fact that a, b are non-orthogonal, given by a prover Merlin. There is a trade-off between size of the proofs and the required dimension to represent the vectors, which we set in a way that bounds the number of

possible proofs to $O(n)$, resulting in a dimension d growing only with c (independently of n).

We also give a conditional lower bound from SETH for $k > 3$ that matches the NSETH barrier following from the (co-)nondeterministic algorithms. Notably, this lower bound already applies to dimension $d = 2$.

Theorem 13 *Assuming SETH, model checking formulas in $PTO_{k,2}$ ending in $\exists\forall\exists$ requires time $\Omega(n^{k-2-\epsilon})$ for any $\epsilon > 0$.*

We reduce the k-Orthogonal Vectors problem into an $\exists^k\forall\exists$-quantified 2-dimensional formula. Intuitively, the first k existential quantifiers choose k vectors, the $\forall$-quantifier ranges over all vector-dimensions to test, and crucially, the final $\exists$-quantifier enables to guess which of the k vectors has a 0-coordinate in this vector-dimension. Here, the final $\exists$-quantifier is instrumental in making the formula's dimension independent of the vector dimensions. We give the full proof in Sect. 6.

3.4 Quantifier Structures Ending in ∃∃∀

For sentences in $PTO_{k,d}$ ending in $\exists\exists\forall$, we obtain the complete problem VCND_d: Given three sets of vectors X, Y and Z of dimension d, d and $2d$, respectively, determine if there an $x \in X$ and a $y \in Y$ so that their concatenation $x \circ y$ is not dominated by any $z \in Z$.

Theorem 14 *For all d, there exists a d' such that $\mathsf{VCND}_{d'}$ is complete for model-checking $\exists\exists\forall$ formulas in $PTO_{k,d}$.*

This is one of our most interesting results, proven in Sect. 5.3. We reduce a formula $\exists x \in X \exists y \in Y \forall z \in Z : \psi(x, y, z)$ to VCND_d as follows: We carefully divide all pairs in $X \times Y$ into instances $(X_1, Y_1), \ldots, (X_L, Y_L)$ such that for each instance (X_ℓ, Y_ℓ), all comparisons $x_i < y_i, x_i = y_i, x_i > y_i$ for all dimensions i have the same outcome among pairs $x \in X_\ell, y \in Y_\ell$. Thus, for each ℓ, we may simplify ψ to a formula ψ_ℓ not involving comparisons between x and y. In particular, we may express ψ_ℓ in CNF, where each clause is a disjunction of $\{<, \leq, \geq, >\}$-comparisons between x_i and z_i or between y_i and z_i (in some dimension i). Since all such clauses need to be fulfilled simultaneously, for each $z \in Z$ and clause C, we introduce some z_C chosen such that the clause C is falsified if and only if $x \circ y$ are dominated by z_C.

We show a matching conditional lower bound of $n^{k-o(1)}$ for $PTO_{\exists^k\forall,d}$ under the 3-uniform hyperclique hypothesis.

Theorem 15 *For $k \geq 2$ and $h \geq 3$, under the h-uniform hk-HyperClique hypothesis, model checking formulas in $PTO_{k+1,hk}$ ending in $\exists\exists\forall$ requires time $\Omega(n^{k-o(1)})$.*

We use the first k quantifiers to represent a choice of clique nodes, each represented in its own dimension, and use the $\forall$ quantifier to check that no forbidden configuration is used (a non-edge in the given hypergraph). Naively, this would create $\Theta(n^h)$ rather than $O(n)$ objects, which we remedy by reducing from finding hypercliques of size hk (rather than k). The proof is given in Sect. 6.

We also establish a SETH-based lower bound directly for VCND_d. The reduction (given in Sect. 6) is very similar to our Hitting-Set-based lower bound for $\exists\forall\exists$-structures.

Theorem 16 *Assuming SETH, for every $\epsilon > 0$, there is a d such that VCND_d requires time $\Omega(n^{2-\epsilon})$.*

Specialized algorithm for VCND_d Since our completeness results establish VCND_d as a central problem for the study of $PTO_{k,d}$, we consider special cases of the problem in Sect. 7. In particular, if X contains vectors of dimension 2 and Y contains vectors of dimension d, we show the following algorithm, which uses the Erdös-Szekeres Theorem as main ingredient. We use this to extract lists of vectors so that when we restrict to any dimension, the vectors appear in monotonic increasing or decreasing order. This way, the vectors that dominate some fixed vector x form an interval, which allows us to take advantage of fast segment trees that solve an interval covering problem.

Theorem 17 *There is a $\widetilde{O}(n^{2-\frac{1}{2^d}})$ time algorithm for VCND when one set of vectors is of dimension 2 and the other is of dimension d.*

Note that such an improvement to $\tilde{O}(n^{2-\epsilon(d)})$ with $\epsilon(d) > 0$ for the general VCND_d problem would refute the 3-uniform hyperclique hypothesis by Theorem 15. Furthermore, we also give an algorithm for VCND_d when $d = O(n)$.

4 Baseline Algorithms

In this section, we give our baseline algorithms via a reduction to orthogonal range counting. We note that we do not aim to optimize logarithmic factors.

Lemma 4 *Given a formula $\varphi(x, y)$ with d ordering relations and unary predicates and two sets X, Y of vectors in $\mathbb{R}^d$, there is an $O(n \log^{d-1}(n))$ time algorithm that returns an array A indexed by each $x \in X$ so that $A[x]$ is the number of $y \in Y$ such that $\varphi(x, y)$ is true.*

Proof Consider a fixed x in the domain. The task is to count the number of y such that $\varphi(x, y)$ is satisfied. Assume the unary relations in the vocabulary are $R_1, \ldots, R_k$. The truth value of $\varphi(x, y)$ will depend on two factors: the order between x and y in each of the d dimensions, and the unary relations $R_1, \ldots, R_k$ satisfied by y. We will denote the first by a vector α and the second by a vector β. Since k and d are constant, there are finitely many possibilities for α and β, so we may consider them all. So, when we find an α and β that makes $\varphi(x, y)$ true, we can orthogonal range search for vectors x, y that have order α and vectors y that satisfy the unary relations given by β.

Formally, consider the truth value of $\varphi(x, y)$ with respect to some $\alpha \in \{0, 1, -1\}^d$ and $\beta \in \{0, 1\}^k$, where the comparison of x and y under the i^{th} ordering is given the truth value according to $\alpha[i]$ and every unary relation on y is given the truth value according to β. Here, $\alpha[i] = 1$ denotes $x >_i y$, $\alpha[i] = 0$ denotes $x =_i y$, and

$\alpha[i] = -1$ denotes $x <_i y$. If $\beta[i] = 1$ then we set $R_i(y)$ to true and otherwise set $R_i(y)$ to false. To compute $|\{y \mid \varphi(x, y) \text{ is satisfied}\}|$, for each (α, β) that satisfies $\varphi(x, y)$, we will count the number of $y \in Y$ such that their order with x is given by α and the unary relations they satisfy is given by β. Then, we will sum these values over every (α, β) that satisfy $\varphi(x, y)$.

Specifically, observe that in linear time, we can compute, for each $\beta \in \{0, 1\}^k$, the set Y_β of vectors with unary relations given by β. To count, given $\alpha \in \{0, 1, -1\}^d$, the number of $y \in Y_\beta$ so that their order with x is α, we apply a standard orthogonal range counting [22, 35]. If $\alpha_i = 1$, the range in the i^{th} dimension will be $[0, x_i)$. If $\alpha_i = 0$, then the range in the i^{th} dimension will be $\{x_i\}$, and if $\alpha_i = -1$, then the range in the i^{th} dimension will be $(x_i, n]$. For example, let $x = (3, 4, 5)$, and let $\alpha = (1, 0, -1)$. Then, we query the number of $y \in Y_\beta$ that lie in the range $[0, 3) \times \{4\} \times (5, n]$. Such a query can be done in time $O(\log^{d-1} n)$, see [22, 35]. Overall, we can compute the total number of y satisfying $\varphi(x, y)$ in time $O(\log^{d-1} n)$.

Performing this for each $x \in X$ takes total time $O(n \log^{d-1} n)$. □

We obtain our baseline algorithm for purely total ordering relations by observing that the above information is sufficient to decide any $PTO_{2,d}$ formula.

Theorem 3 *There is an algorithm running in time $O(n \log^{d-1}(n))$ for model-checking a two-quantifier formula $Q_1 x Q_2 y \varphi(x, y)$ with d ordering relations and unary predicates.*

Proof From Lemma 4, we can compute an array A indexed by vectors $x \in X$ so that $A[x] = \#\varphi(x, \cdot)$ in time $O(n \log^{d-1} n)$. If $Q_1 Q_2$ is $\exists\exists$, it is enough to check that $\#\varphi(x, \cdot) > 0$ for some $x \in X$. Similarly, if $Q_1 Q_2$ is $\exists\forall$, it is enough to check that $\#\varphi(x, \cdot) = |Y|$ for some $x \in X$. Both can be done by simply scanning the array. All other formulas are equivalent to one of these cases by negation. □

Corollary 5 *Model-checking formulas in $PTO_{k,d}$ is in TIME$(n^{k-1} \log^{d-1}(n))$.*

Proof Simply brute force search over the first $k-2$ quantifiers, then use the 2-quantifier algorithm that runs in time $O(n \log^{d-1}(n))$. This takes time $O(n^{k-1} \log^{d-1}(n))$. □

In fact, we can extend these ideas to give a baseline algorithm for the class $TO_{k,d}$.

Theorem 6 *Model-checking formulas in $TO_{k,d}$ is in TIME$(m^{k-1} \log^{d-1}(m))$.*

Proof We will exhaustively search over the first $k - 2$ quantifiers. Then, our plan will be to try to count $|\{y \mid \varphi(x, y) \text{ is satisfied}\}|$ by separating into two cases: these are when objects x and y appear (or do not appear) together in a sparse relation. We will create an auxiliary formula $\varphi^*(x, y)$ where every relation $R(x, y)$ that appears in $\varphi(x, y)$ is set to *false*. Then, we use the algorithm from Lemma 4 to compute an array A^* with $A^*[x] = |\{y \mid \varphi^*(x, y) \text{ is satisfied}\}|$. If y shares no relations with x, then $\varphi(x, y)$ is true if and only if $\varphi^*(x, y)$ is true. However, if x and y appear together in some relation, then it is possible that $\varphi^*(x, y)$ and $\varphi(x, y)$ have different truth values. Since our relation is sparse, we can correct this in $O(m)$ time by exhaustively searching over the vectors y that appear in some relation with x. If $\varphi(x, y)$ is *true* and

$\varphi^*(x, y)$ is *true*, then we do not alter our current count. If $\varphi(x, y)$ is *true* but $\varphi^*(x, y)$ is *false*, then we increment our count by 1. Similarly, if $\varphi(x, y)$ is *false* and $\varphi^*(x, y)$ is *true*, we decrement our count by 1. Lastly, if both $\varphi(x, y)$ and $\varphi^*(x, y)$ are *false*, we do not alter the count. With this, we can compute an array A indexed by x in time $O(m \log^{d-1} m)$ where $A[x] = |\{y \mid \varphi(x, y) \text{ is satisfied}\}|$.

Therefore, given a sentence in $TO_{k,d}$, we exhaustively search over the first $k-2$ quantifiers then compute this array A. If the last two quantifiers are $\exists\exists$, it is enough to check that some entry in the array is greater then 1, and if the last two quantifiers are $\exists\forall$, it is enough to check that some entry in the array is the size of the domain for the last quantifier. Overall this takes time $O(m^{k-1} \log^{d-1} m)$. □

5 Completeness for Quantifier Structures

In this section, we give our completeness results for each quantifier structure, including evidence why we do not expect any quantifier structure other than $\exists\exists\forall$ to contain a complete problem for the full classes $PTO_{k,d}, TO_{k,d}$.

5.1 Quantifier Structures Reducing to Triangle Problems

In this section, we characterize the complexity of problems for the class of $\exists\exists\exists$ and $\forall\exists\exists$ formulas with ordering relations. We show that for $PTO_{\exists\exists\exists}$, the complexity of the hardest problem in the class is equivalent to that of triangle detection in sparse graphs. In other words, triangle detection is (equivalent to) a complete problem for this class. We show that every problem in $TO_{\exists\exists\exists}$ (i.e., when we also allow sparse relations in addition to orderings) reduces to the problem of counting triangles in sparse graphs. (Thus, the complexity of this class is somewhere between deciding whether a triangle exists and counting the number of such triangles. Currently, the best algorithms for these problems are identical ([10]), but there is no known proof of equivalence.) We show that every problem in $TO_{\forall\exists\exists}$ reduces to that of determining, for each edge in a sparse graph, how many triangles contain this edge. Again, currently the best algorithm for this problem is essentially the same as that for triangle detection and counting ([10]).

In particular, these results show that these classes are all decidable in time $O(m^{1.41})$. Hence, these quantifier structures are significantly easier to check than the others we consider (assuming SETH and Low-dimension hitting set conjectures).

5.1.1 $PTO_{\exists\exists\exists}$

Theorem 7 *The triangle detection problem in sparse graphs is fine-grained equivalent to a problem that is complete for model-checking* $\exists\exists\exists$ *formulas with only ordering relations and unary relations.*

Proof The triangle detection problem in sparse graphs asks if for a graph $G = (V, E)$ given in adjacency list format, there exist $x, y, z \in G$ such that $(x, y), (y, z), (x, z) \in E$. We first show that this problem is equivalent under exact-time preserving reductions

to an ordering problem with the same logical structure and dimension 3. The problem is: Given three sets of vectors A, B, C of dimension 3, are there $a \in A, b \in B$, and $c \in C$ with $a_1 = c_1, a_2 = b_2$ and $b_3 = c_3$? To reduce triangle detection to this problem, assign the vertices names that are positive integers, i.e., we identify V with $\{1, \ldots, n\}$. For each edge $(x, y) \in E$, we create vectors $(x, y, 0) \in A$, $(0, x, y) \in B$ and $(y, 0, x) \in C$. Thus, the number of vectors is linear in the number of edges in our graph, and the sets of vectors can be created in linear time. If there is a triangle, x, y, z in the graph, $(x, y, 0) \in A$, $(0, y, z) \in B$, and $(x, 0, z) \in C$ are three vectors satisfying the constraints. Contrapositively, any three vectors satisfying the constraints must be of the above form, so must correspond to a triangle in the original graph.

In the reverse direction, for each vector $(a_1, a_2, a_3) \in A$, create vertices $(1, a_1)$ and $(2, a_2)$ if not already present and add an edge from vertex $(1, a_1)$ to vertex $(2, a_2)$. Similarly, we have edges from $(2, b_2)$ to $(3, b_3)$ for all $(b_1, b_2, b_3) \in B$, and from $(3, c_3)$ to $(1, c_1)$ for $(c_1, c_2, c_3) \in C$. The graph created has a linear number of edges and triangles correspond exactly to solutions to our problem. So this problem is equivalent to triangle detection.

Thus, the maximum complexity of predicates in $PTO_{\exists\exists\exists}$ is at least that of triangle detection, since it is equivalent to a member of this class.

We next show how to reduce any $\exists\exists\exists$ pure ordering problem to triangle detection. The first step is to reduce such a problem to one with only equality checks.

Consider a formula of the form $\exists x \exists y \exists z \varphi(x, y, z)$, where there are d ordering relations. Say that $a < b$ for positive integers a and b, where $a = a_k \ldots a_0$ in binary, and $b = b_k \ldots b_0$ in binary. Call the *position of divergence* the first j (starting at the high order bits) so that that $b_j > a_j$. Then $a_k \ldots a_{j+1} = b_k \ldots b_{j+1}, a_j = 0 < b_j = 1$. If $a = b$, we call -1 the point of divergence. We break up the possible triples x, y, z into cases based on their ordering in the d different orders, and the point of divergence between their ranks for every pair in every order. Since the ranks are integers from 1 to n, there are $1 + \log n$ possible points of divergence. Furthermore, since there are only two possible orderings for each pair per order, there are at most $O(\log^{3d} n)$ cases in total. We further break up into sub-cases based on which subsets of unary relations are true for x, y, z, which is at most constantly many sub-cases per case.

We determine for each case, whether there is an x, y, z with those comparisons and points of divergence and unary relations for which $\varphi(x, y, z)$ holds. However, since each case specifies all comparisons and unary relations, $\varphi(x, y, z)$ is either constantly true or constantly false for these cases. So this simplifies to determining, for each sub-case where $\varphi(x, y, z)$ is true, whether there is a triple (x, y, z) consistent with that sub-case. For this, we use the characterization above. First, for each vector, we discard it if it does not match the unary relations for this sub-case. Secondly, if in the case $x_i < y_i$, and the point of divergence for the comparison is j, we discard x as a possibility if $x_{i,j} \neq 0$ and y as a possibility if $y_{i,j} \neq 1$, and the reverse if $x_i > y_i$.

For each non-discarded vector x, we create a new vector X of dimension 3, where in the second coordinate we concatenate in order of i all the strings $x_{i,k}, ..x_{i,j_i+1}$ where j_i is the point of divergence for x_i and y_i, and do likewise for the points of divergence for x_i and z_i in the first coordinate. The third coordinate has a default value like -1. For y, we do likewise, putting the parts related to the points of divergence with x in

the second coordinate, and with z in the third to create Y, and for z we put the parts related to x in the first coordinate, and the parts related to y in the third.

Then as above, we ask: is there a triple X, Y, Z so that $X_1 = Z_1$, $X_2 = Y_2$ and $Y_3 = Z_3$? If so, since all the strings concatenated have a fixed length, each concatenated string must be identical, so the corresponding x, y, z do have the orders and the points of divergence for the sub-case we are considering. Conversely, if x, y and z have those points of divergence, each string concatenated will be identical, so the equations will hold. As noted above, this problem is equivalent to triangle detection.

Thus, triangle detection is hard for $PTO_{\exists\exists\exists}$, and the equivalent problem is complete for this class. □

Corollary 18 *Model checking for every problem in $PTO_{\exists\exists\exists}$ can be solved in $\tilde{O}(n^{1.407\cdots})$ time.*

Proof Combine the above reduction with the algorithm from [10]. □

5.1.2 $TO_{\exists\exists\exists}$

A more general statement of the construction above is:

Lemma 19 *Let X, Y, Z be sets of vectors of constant dimension d. In time $O(n \log^{O(1)} n)$, we can construct a family of $O(\log^{O(1)}(n))$ tripartite multi-graphs so that:*

1. *For the tripartition of vertices into A, B, C of each graph G_i, each element $x \in X$ corresponds to at most a single edge e_x between A and B, each element $y \in Y$ corresponds to at most a single edge e_y between B and C, and each element $z \in Z$ corresponds to at most a single edge e_z between A and C, and there are no other edges. Given x, i, one can compute e_x in constant time, or say it doesn't exist.*
2. *Given i, we can compute in constant time a complete set of values for all unary relations on x, y, and z, and values for order relations between x_j, y_j and z_j for $1 \leq j \leq d$, so that for every triangle e_x, e_y, e_z in G_i, x, y, z satisfy these relations.*
3. *For every triple x, y, z, e_x, e_y, e_z form a triangle in exactly one G_i.*

We will use this lemma to show:

Theorem 8 *Every problem in $TO_{\exists\exists\exists,d}$ reduces to the problem of counting the number of triangles in a sparse graph via reductions that preserve time up to polylog factors.*

Proof We will first show that the triangle counting problem is complete for the class $\#FO_3$: given a quantifier-free formula $\Phi(x, y, z)$ with only sparse relations, count the number of solutions x, y, z.

Lemma 20 *We can reduce a problem Φ in $\#FO_3$ to the case where we have to count the number of solutions for constantly many formulas where each is a conjunction of positive relations.*

Proof We first reduce from the general case to the case when Φ is a conjunction of relations and negated relations. We branch on all possible settings of the relations that could hold among x, y, z, and we count the number of triples satisfying each conjunction that satisfies Φ (ie., we write Φ as a DNF of mutually exclusive terms, and count each term). Then we add up the results. Secondly, we can reduce to the case when all relations appear positively. If $\neg R(x, y)$ (or any other subset of variables) appears in the conjunction, we can write $\Phi = \neg R(x, y) \wedge \Psi(x, y, z)$, where Ψ has strictly fewer negated relations, as does $R(x, y) \wedge \Psi(x, y, z)$. If we count both the number of triples that satisfy Ψ and $R(x, y) \wedge \Psi$, their difference is the number that satisfy Φ. Thus, we can reduce any such counting problem with $i \geq 1$ negated relations to two such problems with $i - 1$ negated relations. Applying this repeatedly, we reduce to the case with no negations. □

Lemma 21 *Sparse triangle counting is complete for* $\#FO_3$.

Proof We apply Lemma 20 and reduce to a set of counting problems which are conjunctions of positive relations. If the set of binary or greater relations is empty, we can individually count in linear time the number of elements x that satisfy all unary relations $R(x)$ and similarly for y and z, and return their product. If there is a relation that involves all three variables, we enumerate the the triples satisfying that relation and compute the number of those that satisfy Φ. If there are no 3-ary relations, and the binary relations are only between two specific variables (e.g., x and y), we can enumerate all such pairs x, y, then separately count the z's that satisfy unary relations and multiply these two counts. If there are specified relations on one of the variables, say x, and relations between x and y as well as x and z, but no relations between the y and z, we can compute, for each x both the number of consistent y's and the number of consistent z's in time equal to the number of tuples containing x. Then, we multiply these counts and sum up the results. This takes $O(m)$ time total. Finally, if there are relations specified between every pair of variables, we can use these to specify the edges of a tripartite graph where the vertices are the elements that satisfy the unary relations and edges are pairs that satisfy all binary relations. This graph has at most $O(m)$ edges, so it is an instance of sparse triangle counting of the same size as our original problem. □

Now we return to the theorem. Given a formula $\Phi(x, y, z)$ with both ordering and sparse relations as well as input relations, we use the ordering and unary relations to construct the family of multi-graphs G_i as in the lemma. Because each triple x, y, z appears as a triangle in exactly one graph, it suffices to decide whether there is an i and a triple x, y, z so that e_x, e_y, e_z form a triangle in G_i and $\Phi(x, y, z)$ holds. Because for each i, all unary and ordering relations are fixed for triangles, we can compute a restricted formula $\Phi_i(x, y, z)$ with only sparse binary or greater arity relations in x, y, z so that $\Phi_i(x, y, z)$ is equivalent to $\Phi(x, y, z)$ for triangles in G_i. Thus, it is equivalent to decide whether there is an i and a triple x, y, z so that e_x, e_y, e_z form a triangle in G_i and $\Phi_i(x, y, z)$.

G_i is a multigraph, because different elements x might map to edges e_x with the same endpoints (but the elements themselves might have different binary relations and therefore be distinguishable). Let H_i be the graph corresponding to G_i when we

combine parallel edges. For each tuple in a relation and each pair of elements x, y in the tuple, if e_x and e_y do not share an endpoint, x and y cannot be part of a triangle, and if they do, the endpoints form a triple of vertices in H_i. We can enumerate all such triples in $O(m)$ time. Any triangle in H_i that is not one of these triples corresponds to three elements that have no true relations. We can tell if there is such a triangle by counting the total number of triangles in H_i, and subtracting the number of triangles among the $O(m)$ special triples. If Φ_i is true when all relations are true, and there is such a triangle, we can return true. If not, any x, y, z that form a triangle and satisfy Φ_i must form a triangle on our list.

We can, as a linear-time pre-processing step, for each edge (a, b), compute the set of elements x so that e_x goes from a to b, and for each triple in our collection store the set of relations among x, y, z. For each triple $T_j = (a, b, c)$ in our collection, we have sets X_j mapping to (a, b), Y_j mapping to (b, c), and Z_j mapping to (c, a). Let m_j be the number of relations among the elements of these sets. Since any two elements in X_j, Y_j, or Z_j with no relations are indistinguishable, we can remove all but one such element from each set so that there are at most $O(m_j)$ elements total. Since each relation determines at most a single triangle, we have $\sum_j m_j \leq m$. If we can count triangles in time $O(m^{1+\alpha})$ for sparse graphs, as we saw earlier, we can compute the number of triples among X_i, Y_j, Z_j that satisfy Φ_i in $O(m_j^{1+\alpha})$ time. If any count is positive, we return true. If all counts are 0, we return false. The total time is $\sum_j O(m_j^{1+\alpha}) \leq \sum_j O(m^\alpha m_j) \leq O(m^{1+\alpha})$. Thus, the exponent for the general class is the same as for counting triangles in sparse graphs. □

5.1.3 $TO_{\forall\exists\exists}$

We can get a similar but more complex hard problem for $TO_{\forall\exists\exists}$. Consider the problem of given a tripartite graph in adjacency list format, creating an array indexed by edges and giving the number of triangles containing that edge. The method of [10] can be used to give an algorithm with the same complexity for this problem as for counting triangles or deciding whether a triangle exists. Call this problem Edgewise Triangle Counting (ETC).

We will show that the complexity of this problem is an upper bound for the complexity of any problem in $TO_{\forall\exists\exists}$.

Consider the class of problems: For a formula $\Phi(x, y, z)$ and a model given as lists of tuples for each relation, create an array indexed by x, giving the number of y, z so that $\Phi(x, y, z)$ holds. We use similar argument as for the $TO_{\exists\exists\exists}$ case to show that this class of problems reduces to ETC: We apply Lemma 20 and reduce to counting for constantly many formulas which are conjuction of positive relations. We observe that each such formula is essentially either counting triples overall, counting triples containing a single edge, a path of length 2, a triangle, or a hyperedge. All but the triangle can be solved in linear time, even in the array version. In the triangle case, we are counting for each x, the number of triangles involving a single vertex x. However, we can compute the number for a vertex by summing up all the numbers for adjacent edges to some y, since every triangle in the tripartite graph contains exactly one such edge.

Next, to decide a $\forall x \exists y \exists z \Phi(x, y, z)$, we create the graphs G_i again just as before, and define $\Phi_i(x, y, z)$ containing only sparse bipartite or 3-ary relations as before. For each graph, we will find the set of x so that there is a triple x, y, z so that $\Phi_i(x, y, z)$ and e_x, e_y, e_z form a triangle in G_i. The union of these sets is thus the set of x so that there are y, z with $\Phi(x, y, z)$, and we check to see if that is all x. As before, we find the set of triangles in H_i determined by tuples in a relation in $O(m)$ time. For every edge in H_i, we count the number of triangles in H_i containing this edge, and subtract the number of such triangles on our list. For each edge in H_i where this is positive, if Φ_i is satisfied by all false relations, we add the corresponding elements to our set. We then need to also include those x so that there is some triple x, y, z in our set with $\Phi_i(x, y, z)$. As before, if we can solve ETC in time $O(m^{1+\alpha})$, we can solve the array counting problem for the elements corresponding to each triple in $O(m_j^{1+\alpha})$ time, and then mark those elements with a positive array position. All elements with no relations should be marked or unmarked identically, so we only need to include one such element in our sub-routine, but mark all such elements.

Corollary 22 *Model-checking for sentences in $TO_{\forall\exists\exists}$ can be done in time $\tilde{O}(m^{1.41})$.*

5.2 Nondeterministic Complexity of $TO_{\exists\forall\exists}$

Here, we show why problems in $TO_{\exists\forall\exists}$ are unlikely to be SETH-hard. In particular, we will show that every problem in $PTO_{\exists\forall\exists}$ is in $\mathsf{NTIME}(n \log^{O(1)} n) \cap$ co-$\mathsf{NTIME}(n \log^{O(1)} n)$ and every problem in $TO_{\exists\forall\exists}$ is in $\mathsf{NTIME}(m^{3/2} \log^{O(1)} m) \cap$ co-$\mathsf{NTIME}(m^{3/2} \log^{O(1)} m)$. From [17], it then follows that if the Nondeterministic Strong Exponential Hypothesis is true, then no reduction can show these problems are SETH-hard with exponent greater than 1.5. Via direct reduction, for PTO_k, no SETH-hardness can be shown for exponent greater than $k - 2$ for any quantifier sequence ending with $\exists\forall\exists$. Later, we will show that SETH hardness is possible up to that same exponent (Theorem 13).

Lemma 23 *Model-Checking sentences in $PTO_{\exists\forall\exists}$ can be done in nondeterministic and co-nondeterministic time $O(n \log^{d-1} n)$. Similarly, model-checking in $TO_{\exists\forall\exists}$ can be done in nondeterministic and co-nondeterministic time $O(m^{3/2} \log^{d-1} m)$*

Proof Let $\exists x \forall y \exists z \Phi(x, y, z)$ be a problem in $PTO_{\exists\forall\exists}$. To solve it using a nondeterministic algorithm, we guess element x^* nondeterministically and verify $\forall y \exists z \Phi(x^*, y, z)$. This latter is a two quantifier statement and so can be solved in quasi-linear time using the base-line algorithm, once we add unary relations $U(y) \equiv (x^* \leq_i y)$ for each comparison relation $\leq_i$.

The complementary problem is $\forall x \exists y \forall z \neg \Phi(x, y, z)$. To solve it nondeterministically, for each x we guess a y_x. Then we create a new comparison relations $(x \leq_i' z) \equiv (y_x \leq_i z)$ for every comparison relation $\leq_i$, and new unary relations $U(x) \equiv (x \leq y_x)$ for each comparison relation $\leq_i$ and $U'(x) = U(y_x)$ for each unary relation U. This can be done in linear time. Then we can rewrite $\neg \Phi(x, y_x, z) = \Psi(x, z)$ by replacing relations involving y_x with these new relations. Then we verify that $\forall x \forall z \Psi(x, z)$ using the baseline algorithm in quasi-linear time.

If $\Phi(x, y, z)$ also has sparse relations, we can use the same method. However, for the co-nondeterministic algorithm, we need to create relations $R'(x, z) \equiv R(y_x, z)$ for sparse relations R. If many y_x are in many tuples for R, this can blow up the number of tuples in the relation. We use the low-degree/high-degree method to get around this. Without loss of generality, we can assume that the variables x, y, z come from disjoint sub-sets of elements, possibly by duplicating elements. For each y^* that appears in $\geq m^{1/2}$ tuples, we use the baseline algorithm to compute $\{x | \forall z \Phi(x, y^*, z)\}$. We then delete this set of elements x as candidates for the first quantifier since we have shown that the statement $\exists y \forall z \Phi(x, y, z)$ is true for these x, and delete y^* since it cannot be used for any further x's. There are at most $O(\sqrt{m})$ such y, and each use of the baseline algorithm takes quasi-linear time. At the end, all y's appear in at most $\sqrt{m}$ tuples, and we can nondeterministically guess y_x for each remaining x, and create the relations previously described, as well as the relations described above for each sparse relation. Since each x will appear in at most $O(\sqrt{m})$ new tuples, the total number of tuples in the new model is $O(m^{3/2})$. Using the baseline algorithm on the resulting two quantifier model-checking problem thus takes time $O(m^{3/2} \log^{d-1} n)$. □

Exhaustively searching over the first $k - 3$ quantifiers gives us the following.

Theorem 10 *Model-checking formulas in $PTO_{k,d}$ ending in $\exists\forall\exists$ can be done in nondeterministic and co-nondeterministic time $O(n^{k-2} \log^{d-1}(n))$.*

Theorem 11 *Model-checking formulas in $TO_{k,d}$ ending in $\exists\forall\exists$ can be done in nondeterministic and co-nondeterministic time $O(m^{k-3/2} \log^{d-1}(m))$.*

5.3 Quantifier Structure ∃∃∀

Lastly, we will show that VCND is a complete problem for model-checking ∃∃∀ formulas with ordering relations.

Theorem 14 *For all d, there exists a d' such that $\mathsf{VCND}_{d'}$ is complete for model-checking $\exists\exists\forall$ formulas in $PTO_{k,d}$.*

Proof We start with a first-order formula $\exists x \exists y \forall z \varphi(x, y, z)$ containing ordering relations between x, y, and z. We want to reduce to VCND_d: Given sets of d-dimensional vectors X, Y and $2d$-dimensional vectors Z, is there a pair $x \in X$ and $y \in Y$ such that $x \circ y \not\leq_{dom} z$ for all $z \in Z$. We will use a similar technique as in the proof of Theorem 7.

Lemma 24 *We can write*

$$\varphi(x, y, z) \equiv \bigvee_{\alpha \in \{0,1,-1\}^d} \psi_\alpha(x, y) \wedge \varphi_\alpha(x, y, z),$$

where for each α, $\varphi_\alpha(x, y, z)$ does not contain any comparisons between x and y.

Proof For vectors $x \in X$ and $y \in Y$, define $v_{x,y} \in \{0, 1, -1\}^d$ where $v_{x,y}[i] = -1$ if $x <_i y$, $v_{x,y}[i] = 0$ if $x =_i y$, and $v_{x,y}[i] = 1$ if $x >_i y$. The vector $v_{x,y}$ captures the

relationship between x and y with respect to the total orderings $\leq_i$. Thus, we consider the formula

$$\varphi(x, y, z) \equiv \bigvee_{\alpha \in \{0,1,-1\}^d} \psi_\alpha(x, y) \wedge \varphi_\alpha(x, y, z),$$

where $\psi_\alpha(x, y)$ is *true* if and only if $v_{x,y} = \alpha$ and $\varphi_\alpha(x, y, z)$ is obtained by replacing any predicates comparing x and y under the i^{th} ordering relation with the truth value given by α_i. □

Lemma 25 *For each α, we can efficiently construct a set I_α and for each $\ell \in I_\alpha$, construct sets X_ℓ, Y_ℓ with the following properties:*

- *For every pair $x \in X$ and $y \in Y$ with $v_{x,y} = \alpha$, there exists exactly one $\ell \in I_\alpha$ such that $x \in X_\ell$, $y \in Y_\ell$.*
- *For every $\ell \in I_\alpha$, $x \in X_\ell$, $y \in Y_\ell$ it holds that $v_{xy} = \alpha$.*

Proof As before, say that $a < b$ for positive integers a and b, where $a = a_k \dots a_0$ in binary, and $b = b_k \dots b_0$ in binary. The *position of divergence* is the first j starting at the high order bits so that $b_j \neq a_j$. Then, $a_k \dots a_{j+1} = b_k \dots b_{j+1}, a_j = 0 < b_j = 1$. If $a = b$, we call -1 the point of divergence. Recall that we are working with d-dimensional vectors in X and Y with integer entries from 1 to n. Consider the set $S = \{(i_1, \dots, i_d) \mid -1 \leq i_j \leq \log n\}$. We will use elements of S to "guess" the points of divergence between two vectors. Consider arbitrary $w \in S$.

We will alter the vectors in X and Y according to w. Say that $w_i = j$. If $x_i = a_k \dots a_1$ and $y_i = b_k \dots b_1$ in binary, then we replace the i^{th} coordinate in x with two coordinates, these being $a_k \dots a_{j+1}$ and a_j. Similarly, we replace the i^{th} coordinate in y with two coordinates $b_k \dots b_{j+1}$ and b_j. If $j = -1$ we can simply put a special symbol for the second coordinate. We perform this operation for each coordinate i and each $x \in X$ and $y \in Y$.

Then, we sort these vectors by the first dimension. We will group together the vectors that have the same value in the first coordinate. If $\alpha_1 = 1$ (i.e. its required that the first entry of x has value larger than y), then we will discard all the vectors in the group that belong to the set X and have 0 in the second coordinate (since at point of divergence y will have larger value). Similarly, we will discard all the vectors in the group that belong to the set Y and have 1 in the second coordinate. Analogously, if $\alpha_1 = -1$, we discard vectors from X which have 1 in the second coordinate and vectors from Y when they have 0 in the second coordinate. If $\alpha_1 = 0$, we discard the all the vectors unless $w_1 = -1$. Notice now that for every pair of vectors $x \in X$ and $y \in Y$ that belong to this group, the relationship of x and y under $\leq_1$ agrees with α_1. We recurse on each dimension and perform this for each group. The vectors that came from X then form the set X_ℓ and the vectors from Y form Y_ℓ. The set I indexes each of the possible groups that were formed. □

Now, for each pair of sets X_ℓ, Y_ℓ we will create a VCND instance such that it is a yes instance if and only if there exist $x \in X_\ell$, $y \in Y_\ell$ such that $\varphi_\alpha(x, y, z)$ is true for every $z \in Z$.

We will assume that the vectors in X_ℓ and Y_ℓ appear in their original form, rather than how they were altered in the previous step. Additionally, to make the reduction work, each vector $x = (x_1, \ldots, x_d)$ will be altered to be $(x_1, -x_1, \ldots, x_d, -x_d)$. We perform the same operation to vectors in Y. We can assume that $\varphi_\alpha(x, y, z)$ is written in conjunctive normal form. For each clause C in $\varphi_\alpha(x, y, z)$ and each $z \in Z$, we create a new vector z_C in $4d$ dimensions. Let $z = (z_1, \ldots, z_{2d})$. If the comparison $x >_i z$ appears in the clause C, then we set the $(2i-1)^{th}$ coordinate to z_i and the $2i^{th}$ coordinate to ∞. If the comparison $x <_i z$ appears in the clause C, then we set the $(2i-1)^{th}$ coordinate to ∞ and the $2i^{th}$ coordinate to $-z_i$. We perform the same operation for comparisons between y and z, this time making the changes in the corresponding dimensions in the last $2d$ dimensions of z_C. If $x \leq_i z$ appears, then as our vectors have integer entries, we can treat this as $x - 1 <_i z$ (the same trick works for $x \geq_i z$). We can assume $x =_i y$ does not appear in any clause since $((x =_i y) \vee C)$ where C is some clause is equivalent to $((x \geq_i y) \vee C) \wedge ((x \leq_i y) \vee C)$. To give an example, when $d = 2$, and we have the clause $(x >_1 z) \vee (y <_2 z)$, we create the vector $(z_1, \infty, \infty, \infty, \infty, \infty, \infty, -z_2)$. Thus, if $x \circ y \not\leq_{\text{dom}} z$, then x, y, z satisfy this clause. We create this vector z_C for each clause C in $\varphi_\alpha(x, y, z)$ and each $z \in Z$. At least one of these VCND instances is a yes-instance if and only if there is some $x \in X$ and $y \in Y$ so that x, y, z satisfy each clause of $\varphi(x, y, z)$ for all $z \in Z$.

The last point to make is that this reduction is fine-grained. We have many VCND instances of the form X_ℓ, Y_ℓ, Z where $\ell \in I_\alpha$. If $|X_\ell|$ or $|Y_\ell|$ is of size less than $n^{1-\epsilon/3}$, then we use the data structure from [18] to decide this instance in time $O(|X_\ell||Y_\ell| \log^{O(d)})$. Doing this for each instance where either $|X_\ell|$ or $|Y_\ell|$ is less than $n^{1-\epsilon/3}$ can take time at most $n^{2-\epsilon/3}$. Otherwise, $|X_\ell||Y_\ell| \geq n^{2-2\epsilon/3}$. Since there are at most $|X||Y| = O(n^2)$ many pairs that can arise, we are in this case at most $n^{2\epsilon/3}$ times. If we use the improved $O(n^{2-\epsilon})$ time algorithm on these instances, we will use time at most $O(n^{2-\epsilon/3})$. Combining this with the previous step gives an $O(n^{2-\epsilon/3})$ algorithm for model-checking the sentence $\exists\exists\forall\varphi(x, y, z)$. □

6 Hardness Results

In this section, we will present the proofs of Theorem 15, 12 and 13. These results will establish hardness for model-checking sentences ending in $\exists\exists\forall$ or $\exists\forall\exists$.

Theorem 15 *For $k \geq 2$ and $h \geq 3$, under the h-uniform hk-HyperClique hypothesis, model checking formulas in $PTO_{k+1,hk}$ ending in $\exists\exists\forall$ requires time $\Omega(n^{k-o(1)})$.*

Proof For simplicity, we will state the proof for $h = 3$; the adaptation to $h > 3$ is straightforward. We will reduce determining if a 3-uniform hypergraph contains a $3k$-HyperClique to deciding a $k+1$ quantifier sentence in $PTO_{k+1,3k}$. As a warmup, we will reduce 3-uniform k-HyperClique to a sentence in $PTO_{k+1,k}$ with $O(n^3)$ objects, which gives an $\Omega(n^{k/3-o(1)})$ lower bound. Then, we will describe how to alter the sentence by reducing from 3-uniform $3k$-HyperClique to give the desired lower bound (for $h > 3$, this will correspond to reduction from h-uniform hk-HyperClique).

Without loss of generality, we may assume that we are given a k-partite 3-uniform hypergraph $G = (V_1 \cup \cdots \cup V_k, E)$ using standard color-coding arguments. We view each V_i as a disjoint copy of $\{1, \ldots, n\}$.

The symbols $\square$ and $*$ are special constants which we use to differentiate between different *vertex* and *non-edge* objects, which will introduce now. For $1 \leq i \leq k$ and each vertex $v \in V_i$, we introduce a *vertex object* of dimension k where the i-th entry is set to v, and remaining $k-1$ entries are set to $\square$. This allows us to represent a choice for including some vertex v for part V_i into our clique

For each non-edge $\{v_a, v_b, v_c\} \notin E$ with $v_a \in V_a, v_b \in V_b, v_c \in V_c$, we create a *non-edge object*: We set the a-th, b-th and c-th dimension to v_a, v_b and v_c, respectively, and set all other dimensions to the special constant $*$. Intuitively, the *non-edge objects* represents all forbidden configuration of our clique.

The claim is that deciding the following formula decides the existence of a k-HyperClique:

$$\exists x_1 \exists x_2 \ldots \exists x_k \forall y T(x_1, \ldots, x_k) \wedge \left(E(y) \rightarrow \left(\bigwedge_{1 \leq i_1 < i_2 < i_3 \leq k} C(x_{i_1}, x_{i_2}, x_{i_3}, y) \right) \right),$$

where

- $T(x_1, \ldots, x_k)$ checks if each x_i is a *vertex* object for the part V_i. Doing so will simply involve checking that for all i, all but the i-th coordinate of x_i is the $\square$ constant.
- $E(y)$ checks if y is a *non-edge* object. This can be done by checking whether some coordinate is the $*$ constant.
- $C(x_{i_1}, x_{i_2}, x_{i_3}, y)$ checks if the "forbidden" edge represented by y is different from the edge given by the vertices that $x_{i_1}, x_{i_2}, x_{i_3}$ represent. This can be done by checking that y is different from at least one of x_{i_1}, x_{i_2}, or x_{i_3} in the i_1-th, i_2-th, and i_3-th coordinate respectively.

However, there are $O(n^3)$ many vectors in the domain. We will now remedy this by reducing from $3k$-HyperClique. To this end, let G be a $3k$-partite 3-uniform hypergraph with vertex parts $V_1, \ldots, V_{3k}$. This time, each *vertex* object will represent a choice of 3 vertices: we group the $3k$ vertex parts into k groups $V'_1, \ldots, V'_k$ of three vertex parts each. For each V'_i (representing the three vertex parts $V_{3i+1}, V_{3i+2}, V_{3i+3}$), and every triplet of vertices $v \in V_{3i+1}, v' \in V_{3i+2}, v'' \in V_{3i+3}$, we create a vertex object of dimension $3k$, where we set the coordinates $3i+1, 3i+2$ and $3i+3$ to v, v', and v'', respectively, and all other to the special constant $\square$.

The edge objects will be constructed as before. The formula will change very slightly to implement the same idea as before: The formula $T(x_1, \ldots, x_k)$ will again check that the $x_1, \ldots, x_k$ are *vertex* objects and $E(y)$ will check that y is a *non-edge* object. For any non-edge object y, we need to ensure that the non-$*$ dimensions a, b, c are not all equal to the dimensions a, b, c in the corresponding vertex objects $x_{a'}, x_{b'}, x_{c'}$, where a', b', c' denote the groups $V_{a'}, V'_{b'}, V'_{c'}$ containing V_a, V_b, V_c, respectively. Again, we have $O(n^3)$ objects, but this time we reduced from $3k$-HyperClique: An $O(n^{k-\epsilon})$-

time algorithm for model-checking the above sentence (in $PTO_{k+1,3k}$) would give an $O(n^{3k-3\epsilon})$ algorithm for 3-uniform $(3k)$-HyperClique. □

The following result establishes hardness of ∃∀∃ with no additional quantifiers.

Theorem 12 *Assuming the Hitting Set conjecture, for all $\epsilon > 0$, there exists some d such that model checking formulas in $PTO_{\exists\forall\exists,d}$ requires time $\Omega(n^{2-\epsilon})$.*

Proof Consider the Hitting Set problem: we are given sets U, V of n vectors in $\{0,1\}^d$ with $d = c \log n$, and the task is to determine whether there is some $u \in U$ such that for all $v \in V$ there is some $k \in [d]$ with $u[k] = v[k] = 1$. Recall that the Hitting Set conjecture is that for all $\varepsilon > 0$ there is some c such that Hitting Set with $d = c \log n$ cannot be solved in time $O(n^{2-\varepsilon})$.

The idea is to block the d vector-dimensions into $b = \lceil d/s \rceil$ blocks of size $s = \log(n)/2$, and define a $2b$-dimensional order property: For each vector $u \in U$, we define an object whose first b dimensions represent u. Here, each dimension i encodes the i-th block of s bits of u, i.e., each dimension i uses an (arbitrary) total order on the block configurations $\{0,1\}^s$. Likewise, for each vector $v \in V$, we define an object whose last b dimensions represent the bits of v.

The formula will be:

$$\exists x \in X \forall y \in Y \exists z \in Z : \bigwedge_{i=1}^{b} (z_i = z_{b+i} = \infty \text{ or } (x_i = z_i \wedge y_{i+b} = z_{i+b})) .$$

Here, for any x, y, an appropriately chosen $z \in Z$ is supposed to serve as a witness that there is some $k \in [d]$ with $x[k] = y[k] = d$. To do this, for any block i, we consider pairs of admissible configurations of the i-th blocks of x and y, namely: for any $\alpha, \beta \in \{0,1\}^s$ such that there is some $k \in [s]$ with $\alpha[k] = \beta[k] = 1$, we define the object $z_{i,\alpha,\beta}$ such that its i-th dimension in the first half is α, its i-th dimension in the second half is β, and all other dimensions are ∞.

By this construction, the formula is satisfied if and only if for there is some $u \in U$ such that for all $v \in V$ we can find a block i and a corresponding bit k in block i in which both x and y have a 1, i.e., u and v are non-orthogonal. Since $|Z| \leq 2^{2s} = n$, we obtain our lower bound, assuming the Hitting Set conjecture: Let $\varepsilon > 0$ and take a c such that Hitting Set on dimension $d = c \log n$ has no $O(n^{2-\varepsilon})$ time algorithm. Then, a $O(n^{2-\varepsilon})$ time algorithm for $PTO_{\exists\forall\exists,b}$ with $b \leq \lceil 2c \rceil$ would give a Hitting Set algorithm on dimension $d = c \log n$ in time $O(n^{2-\varepsilon})$, contradicting the assumption. This concludes the claim. □

Finally, for k-quantifier sentences ending in ∃∀∃, we have the following result.

Theorem 13 *Assuming SETH, model checking formulas in $PTO_{k,2}$ ending in ∃∀∃ requires time $\Omega(n^{k-2-\epsilon})$ for any $\epsilon > 0$.*

Proof We will reduce k-Orthogonal Vectors to deciding a first-order sentence with $k+2$ quantifiers ending in ∃∀∃ with 2 ordering relations. We will associate the elements of the domain with 2-dimensional vectors. Let $A = \{a_1, \ldots, a_n\}$ be our k-Orthogonal

vectors instance. We will assume that for every coordinate j, there is some vector $a \in A$ with $a[j] = 0$. Otherwise, this is trivially a no-instance. For each vector a_i and coordinate j where $a_i[j] = 0$, we introduce a vector (i, j) into our domain. Thus, we have $O(nd)$ many vectors in our domain. The claim is that deciding the following sentence on this new domain correctly decides if there are k-Orthogonal vectors in A:

$$\exists x_1 \exists x_2 \ldots \exists x_k \forall y \exists u \bigvee_{i=1}^{k} (u_1 = x_{i,1} \wedge u_2 = y_2) .$$

Say the sentence is satisfied by our domain. Let the first coordinate of x_i be o_i. Then, we claim that $a_{o_1}, \ldots, a_{o_k}$ are a k-orthogonal set of vectors. By our assumption, for every $1 \leq j \leq d$ there is some vector $a_v \in A$ with $a_v[j] = 0$. The universal quantifier ensures that for the correspdonding vector $y = (v, j)$, our sentence is satisfied and so, there is some object x_i of the form (o_i, j). Therefore, there is a 0 in the j^{th} coordinate of a_{o_i}. As this is true for all $1 \leq j \leq d$, we infer that this choice of vectors is indeed k-orthogonal.

Conversely, if there is a k-orthogonal tuple $a_{o_1}, \ldots, a_{o_k} \in A$, then choose $x_1, \ldots, x_k$ such that $x_i = (o_i, j_i)$ for each $1 \leq i \leq k$ (and an arbitrary 0-coordinate j_i). Observe that for any choice of $y = (i', j')$ there is some a_i with $a_i[j'] = 0$, and thus $u = (i, j')$ satisfies the condition.

Consequently any $O(n^{k-\epsilon})$-time algorithm for this $(k + 2)$-quantifier sentence would give a $O(n^{k-\epsilon}\mathsf{poly}(d))$ algorithm for k-OV, contradicting the Moderate-dimension k-OV conjecture and thus SETH. □

We prove our $n^{2-o(1)}$ lower bound for VCND_d under SETH via reduction from the low-dimensional Orthogonal Vectors Hypothesis which is well-known to be implied by SETH (see, e.g. [30]). This reduction is very similar to our Hitting Set reduction for $\exists\forall\exists$.

Theorem 16 *Assuming SETH, for every $\epsilon > 0$, there is a d such that VCND_d requires time $\Omega(n^{2-\epsilon})$.*

The result follows from the following lemma, since the low-dimension Orthogonal Vectors Hypothesis states that for every ϵ, there exists some c such that OV with dimension $d = c \log n$ cannot be solved in time $O(n^{2-\epsilon})$.

Lemma 26 *For constant $c > 0$ and $T(n) = \omega(n)$, if VCND_{4c} is solvable in time $O(T(n))$, then $\mathsf{OV}_{c\log(n)}$ is solvable in time $\widetilde{O}(T(n))$, or*

$$(\mathsf{OV}_{c\log n}, T(n)) \leq_{FGR} (\mathsf{VCND}_{4c}, T(n)) .$$

Proof Consider an Orthogonal Vectors instance on dimension $c \log n$ vectors. Let $s = \frac{\log n}{2}$. First, for each vector $x = (x_1, x_2, \ldots, x_{c\log n})$, we will create a new vector $x' = (x'_1, x'_2, \ldots, x'_b)$ where $b = \frac{c\log n}{s} = 2c$ and x'_i is the integer given by the bits $x_{s(i-1)+1} x_{s(i-1)+2} \ldots x_{si}$. In other words, we are grouping s bits of x at a time and converting them to integer values. Lastly, we will convert each vector $x' =$

$(x'_1, x'_2, \ldots, x'_b)$ to $(x'_1, -x'_1, x'_2, -x'_2, \ldots, x'_b, -x'_b)$. The set of x' vectors become X', and Y' is obtained by performing the same operation to the vectors in Y. Lastly, we want to create a set of vectors Z that will "encode" witnesses to non-orthogonality. Consider a vector $\alpha \in \{0,1\}^s$. Let $\alpha^\perp$ denote the set of vectors orthogonal to α. For each $i \in [b]$, $\alpha \in \{0,1\}^s$, and $\beta \in \{0,1\}^s \setminus \alpha^\perp$ we create vectors of the form

$$(\infty, \ldots, \underset{2i-1}{\alpha}, \underset{2i}{-\alpha}, \ldots, \infty) \circ (\infty, \ldots, \underset{2i-1}{\beta}, \underset{2i}{-\beta}, \ldots, \infty)\,.$$

Here, in the vector notation, we are implicitly viewing α and β as integers specified by the binary strings α, β. This vector has the property that it dominates some other vector if and only if the two agree on the indices that are not ∞ in our gadget. These vectors then form our Z set. Now, assume the Orthogonal Vectors instance indeed contained an orthogonal pair of vectors x and y. Consider the vectors $x' \in X'$ and $y' \in Y'$. Assume for the sake of contradiction that some vector $z \in Z$ dominates $x' \circ y'$. Then, $x' \circ y'$ must have agreed on the non-∞ entries of z. But these are exactly the entries given by some α and β, where α and β have binary representations that are not orthogonal, which contradicts the assumption that x is orthogonal to y. Similarly, if all of the pairs of vectors x, y were not orthogonal, then they must have agreed with some vector in z on the non-∞ entries, so they could not form a yes-instance of VCND.

Creating the set of X' and Y' vectors takes time linear in the number of vectors in the OV instance. Creating the set of Z vectors takes time $O(2^{2s} \cdot b) = O(n)$ with our choice of s, b. □

7 Specialized Algorithms for VCND$_d$

Since VCND$_d$ turned out to be the only candidate for completeness of $PTO_{k,d}$, we study this problem in more detail in this section.

First, we will present an improved algorithm for VCND$_d$ when the dimension of one of the sets is very small. Then, when $d = O(n)$, we give an improved algorithm for VCND$_d$ using fast matrix multiplication.

Theorem 17 *There is a $\widetilde{O}(n^{2-\frac{1}{2^d}})$ time algorithm for VCND when one set of vectors is of dimension 2 and the other is of dimension d.*

Proof We will assume the set X contains vectors of dimension d and Y contains vectors of dimension 2. We utilize the well-known Erdös-Szekeres Theorem to preprocess the vectors in X. There are many equivalent formualations of this theorem, but the version we will use is as follows: In a list of n integers, there is a monotonic subsequence of size at least $\lceil\sqrt{n}\rceil$. Consider the vectors in X restricted to their first coordinate. This is indeed a list of length n. We compute the longest increasing subsequence on the list in order and in reverse, which is guaranteed to return a monotonic sequence of length at least $\lceil\sqrt{n}\rceil$. We then recurse on the list of vectors whose first coordinate is part of the monotonic subsequence, this time considering the next dimension of these vectors. We repeat this process for the remaining set of at most $n - \lceil\sqrt{n}\rceil$ vectors. The result of this preprocessing is $O(n^{1-\frac{1}{2^d}})$ lists each of size $O(n^{\frac{1}{2^d}})$ where each of the

lists are monotonic in each dimension. This preprocessing takes time $O(n^{3/2} \log n)$ since we can compute longest increasing subsequences in time $O(n \log n)$. Note that each of these lists will have the property that when the vectors are viewed restricted to some dimension, the vectors appear in either increasing or decreasing order.

For each of the $O(n^{1-\frac{1}{2^d}})$ lists, we begin with the first vector x in the list. We can keep multiple copies of the Z set where in each copy the list is sorted with respect to a certain dimension. Then, we use binary search to compute the set of $z \in Z$ that dominate the given x. We can keep pointers in place at each iteration so that updating this set is fast for the next x' in the list. Let L_x denote the list of $z \in Z$ that dominate x. The last observation is that since the set of Y vectors is of dimension 2, we can remove vectors to make the set Pareto optimal. Then, we can assume that the set of Y have the property that they are in increasing order in the first dimension, and in decreasing order in the second dimension. Therefore, for a fixed z, the set of $y \in Y$ that are dominated by z form a contiguous interval. We can compute this interval with two binary searches. Thus, for each $z \in L_x$, we add the interval of y vectors that are dominated by z to a segment tree [19], adding 1 to each entry occupied by an interval. We then query the min-element in the segment tree. If the min-element is 0, then there was some $y \in Y$ that was not dominated by any $z \in L_x$, in which case we accept. Otherwise, we continue to compute $L_{x'}$, where x' is the next vector in the list. Since we preprocessed the list, we can update L_x using the saved pointers to compute $L_{x'}$. We will perform at most n updates to L_x to compute $L_{x'}$ for any x' in the list.

The running time of this algorithm is $\widetilde{O}(n^{2-\frac{1}{2^d}})$ since in each of the $O(n^{1-\frac{1}{2^d}})$ lists, we perform $O(n)$ updates to the tree at each step. Each of the queries to the segment tree are logarithmic in n. □

One might hope to extend this algorithm to VCND on two sets of d-dimension vectors using a generalization of the segment tree seen here. Indeed, a multidimensional segment tree supporting addition and min-queries in time poly-logarithmic in n would provide a truly subquadratic algorithm for VCND_d. However, this may be too much to hope for since a multidimensional segment tree supporting these operations would violate SETH [36, 39].

Lastly, when the dimension $d = O(n)$, we can use a similar idea from Williams [48] to obtain speedups using fast matrix multiplication.

Theorem 27 *When $d = O(n)$, there is an algorithm running in time $O(n^\omega + n^2 d)$ for VCND_d where ω is the matrix multiplication constant.*

Proof For each $\mathbf{x} \in X$, we create a bit vector $\mathbf{v_x}$ where $\mathbf{v_x}[i] = 1$ if and only if $\mathbf{x}$ is dominated by the i^{th} vector in Z. We define $\mathbf{v_y}$ similarly. Computing the $\mathbf{v_x}$ and $\mathbf{v_y}$ takes time $O(n^2 d)$. Then, create matrices $\mathbf{A}$ and $\mathbf{B}$ where the columns of $\mathbf{A}$ and $\mathbf{B}$ are the $\mathbf{v_x}$ and $\mathbf{v_y}$, respectively. Compute $\mathbf{A}^\top \mathbf{B}$ using fast matrix multiplication, and check if any of the entries in the resulting matrix are 0. In the resulting matrix, a 0 appears in the location (i, j) iff the vectors at indices i and j are a witness to VCND_d □

8 Conclusion and Open Problems

We have introduced general classes $TO_{k,d}$, $PTO_{k,d}$ of multidimensional ordering problems as model-checking problems for k-quantifier first-order formulas over d succinctly represented ordering relations (with or without additional explicitly represented relations). We gave a conditionally tight algorithm running in time $O(m^{k-1}\log^d m)$ for all these problems. For $PTO_{k,d}$, we gave complete or hard problems for most quantifier structures, and identified a problem VCND_d as the essentially only candidate to be complete for $PTO_{k,d}$.

The main open problem is to prove or disprove that VCND_d is complete for $PTO_{k,d}$. The major challenge here is to reduce $\exists\forall\exists$-quantified ordering problems to the $\exists\exists\forall$-quantified VCND_d. Such a reduction is possible in the unordered setting [30], but its unclear how to make this approach work in our setting. Likewise, can we prove that a hybrid version of VCND_d and the orthogonal vectors problem (which is complete for the sparse-relational setting [30]) is complete for $TO_{k,d}$? An intermediate step could be to find a complete problem for $\exists\forall\exists$-quantified ordering problems.

A further general algorithmic question is to study existence of improved algorithms for very small constant dimensions d, such as $d = 1$ and $d = 2$, in particular the existence of $O(n^{2-\epsilon(d)})$ time algorithms with $\epsilon(d) > 0$, for 3-quantifier problems. In this direction, we have given an $O(n^{2-\frac{1}{2^d}})$-time algorithm for the central VCND problem where one set of vectors has dimension 2 and the other has dimension d. Note that by our results, such an algorithm for the general VCND_d problem would refute the 3-uniform HyperClique conjecture. Can we classify which problems admit such improved algorithms for small dimensions?

Acknowledgements We would like to thank Rex Lei, Jiawei Gao, and Victor Vianu for helpful comments and discussion.

Funding *Russell Impagliazzo:* Work supported by the Simons Foundation and NSF grant CCF-1909634. *Marvin Künnemann:* Research supported by Dr. Max Rössler, by the Walter Haefner Foundation, and by the ETH Zürich Foundation. Part of this research was performed while the author was employed at MPI Informatics.

Declarations

Conflict of interest The authors are not aware of any conflict of interests.

References

1. Abboud, A., Backurs, A., Williams, V.V.: Tight hardness results for LCS and other sequence similarity measures. In: Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on, pp. 59–78. IEEE (2015)
2. Abboud, A., Bringmann, K., Dell, H., Nederlof, J.: More consequences of falsifying SETH and the orthogonal vectors conjecture. In: Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, pp. 253–266. ACM (2018)
3. Abboud, A., Grandoni, F., Williams, V.V.: Subcubic equivalences between graph centrality problems, apsp and diameter. In: Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms, pp. 1681–1697. SIAM (2014)

4. Abboud, A., Williams, R., Yu, H.: More applications of the polynomial method to algorithm design. In: Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 218–230. SIAM (2015)
5. Abboud, A., Williams, V.V., Wang, J.R.: Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In: Krauthgamer, R. (ed.) Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, January 10-12, 2016, pp. 377–391. SIAM, Arlington, VA, USA (2016)
6. Abboud, A., Williams, V.V., Weimann, O.: Consequences of faster alignment of sequences. In: International Colloquium on Automata, Languages, and Programming, pp. 39–51. Springer (2014)
7. Abiteboul, S., Hull, R., Vianu, V.: Foundations of databases: The logical level. Addison-Wesley Longman Publishing Co., Inc., Boston, MA (1995)
8. Agarwal, P.K.: Range searching. In: Goodman, J.E., O'Rourke, J., Tóth, C.D. (eds.) Handbook of Discrete and Computational Geometry, chapter 40. 3rd edition, CRC Press LLC (2017)
9. Agarwal, U., Ramachandran, V.: Fine-grained complexity for sparse graphs. In: Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, pp. 239–252 (2018)
10. Alon, N., Yuster, R., Zwick, U.: Finding and counting given length cycles. Algorithmica **17**(3), 209–223 (1997)
11. Alstrup, S., Brodal, G.S., Rauhe, T.: New data structures for orthogonal range searching. In: 41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, pp. 198–207. IEEE Computer Society, Redondo Beach, California, USA (2000)
12. Backurs, A., Indyk, P.: Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In: Proceedings of the forty-seventh annual ACM symposium on Theory of computing, pp. 51–58. ACM (2015)
13. Backurs, A., Roditty, L., Segal, G., Williams, V.V., Wein, N.: Towards tight approximation bounds for graph diameter and eccentricities. In: Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, pp. 267–280. ACM (2018)
14. Bringmann, K.: Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless seth fails. In: Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on, pp. 661–670. IEEE (2014)
15. Bringmann, K., Fischer, N., Künnemann, M.: A fine-grained analogue of schaefer's theorem in P: dichotomy of exists k-forall-quantified first-order graph properties. In: Shpilka, A. (ed.) 34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA, volume 137 of LIPIcs, pp. 31:1–31:27. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019)
16. Bringmann, K., Künnemann, M.: Quadratic conditional lower bounds for string problems and dynamic time warping. In: Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on, pp. 79–97. IEEE (2015)
17. Carmosino, M.L., Gao, J., Impagliazzo, R., Mihajlin, I., Paturi, R., Schneider, S.: Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In: Sudan, M. (ed.) Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016, pp. 261–270. ACM (2016)
18. Chan, Timothy M.: Orthogonal range searching in moderate dimensions: k-d trees and range trees strike back. Discret. Comput. Geom. **61**(4), 899–922 (2019)
19. Chan, T.M., Larsen, K.G., Patrascu, M.: Orthogonal range searching on the ram, revisited. In: Hurtado, F., van Kreveld, M.J. (eds.) Proceedings of the 27th ACM Symposium on Computational Geometry, Paris, France, June 13-15, 2011, pp. 1–10. ACM (2011)
20. Chan, T.M., Skrepetos, D.: All-pairs shortest paths in geometric intersection graphs. JoCG **10**(1), 27–41 (2019)
21. Chan, T.M., Williams, R.: Deterministic APSP, Orthogonal Vectors, and More: Quickly derandomizing Razborov-Smolensky. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1246–1255. SIAM (2016)
22. Chan, T.M., Zhou, G.: Multidimensional range selection. In: Elbassioni, K.M., Makino, K. (eds.) Algorithms and Computation - 26th International Symposium, ISAAC 2015, Nagoya, Japan, December 9-11, 2015, Proceedings, volume 9472 of Lecture Notes in Computer Science, pp. 83–92. Springer (2015)
23. Chazelle, B.: A functional approach to data structures and its use in multidimensional searching. SIAM J. Comput. **17**(3), 427–462 (1988)

24. Chen, L., Williams, R.: An equivalence class for orthogonal vectors. In: Chan, T.M. (ed.) Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, January 6-9, 2019, pp. 21–40. SIAM, San Diego, California, USA (2019)
25. de Berg, M., Bodlaender, H.L.., Kisfaludi-Bak, S., Marx, D., van der Zanden, T.C..: A framework for eth-tight algorithms and lower bounds in geometric intersection graphs. In: Diakonikolas, I., Kempe,D., Henzinger, M. (eds.) Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, June 25-29, 2018, pp. 574–586. ACM, Los Angeles, CA, USA (2018)
26. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.C.: Orthogonal Range Searching, pp. 95–120. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
27. Gajentaan, A., Overmars, M.H.: On a class of $o(n^2)$ problems in computational geometry. Comput. geom. **5**(3), 165–185 (1995)
28. Gao, J..: On the fine-grained complexity of least weight subsequence in multitrees and bounded treewidth dags. In: Jansen, B.M.P., Telle, J.A. (eds.) 14th International Symposium on Parameterized and Exact Computation, IPEC 2019, September 11-13, 2019, Munich, Germany, volume 148 of LIPIcs, pp. 16:1–16:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019)
29. Gao, J., Impagliazzo, R.: The fine-grained complexity of strengthenings of first-order logic. Electronic Colloquium on Computational Complexity (ECCC), **26**, 9 (2019). URL: https://eccc.weizmann.ac.il/report/2019/009
30. Gao, J., Impagliazzo, R., Kolokolova, A., Williams, R.: Completeness for first-order properties on sparse structures with algorithmic applications. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '17, pp. 2162–2181 (2017)
31. Immerman, N.: Descriptive complexity. Graduate texts in computer science. Springer, New York, NY (1999)
32. Impagliazzo, R., Lovett, S., Paturi, R., Schneider, S.: 0-1 integer linear programming with a linear number of constraints. (2014). arXiv:1401.5512
33. Impagliazzo, R., Paturi, R., Schneider, S.: A satisfiability algorithm for sparse depth-2 threshold circuits. (2012). arXiv:1212.4548
34. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? In: Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on, pp. 653–662. IEEE (1998)
35. JáJá, J.F., Mortensen, C.W., Shi, Q.: Space-efficient and fast algorithms for multidimensional dominance reporting and counting. In: Fleischer, R., Trippen, G. (eds.) Algorithms and Computation, 15th International Symposium, ISAAC 2004, Hong Kong, China, December 20-22, 2004, Proceedings, volume 3341 of Lecture Notes in Computer Science, pp. 558–568. Springer (2004)
36. Korhonen, T.: On multidimensional range queries. (2019). https://laakeri.kapsi.fi/a/rmq.pdf, Accessed: 2021-06-27
37. Künnemann, M., Marx, D.: Finding small satisfying assignments faster than brute force: A fine-grained perspective into boolean constraint satisfaction. In: Saraf, S. (ed.) 35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference), volume 169 of LIPIcs, pp. 27:1–27:28. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020)
38. Künnemann, M., Paturi, R., Schneider, S.: On the Fine-Grained Complexity of One-Dimensional Dynamic Programming. In: 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017), volume 80 of Leibniz International Proceedings in Informatics (LIPIcs), pp. 21:1–21:15 (2017)
39. Lau, J., Ritossa, A.: Algorithms and hardness for multidimensional range updates and queries. In: Lee, J.R. (ed.) 12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference, volume 185 of LIPIcs, pp. 35:1–35:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021)
40. Lincoln, A., Williams, V.V., Williams, R.: Tight hardness for shortest cycles and paths in sparse graphs. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1236–1252. Society for Industrial and Applied Mathematics (2018)
41. Lueker, G.S.: A data structure for orthogonal range queries. In: 19th Annual Symposium on Foundations of Computer Science, 16-18 October 1978, pp. 28–34. IEEE Computer Society, Ann Arbor, Michigan, USA (1978)
42. Marx, D., Sidiropoulos, A.: The limited blessing of low dimensionality: when 1-1/d is the best possible exponent for d-dimensional geometric problems. In: Cheng, S.-W., Devillers, O. (eds.) 30th Annual

Symposium on Computational Geometry, SOCG'14, June 08 - 11, 2014, pp. 67. ACM, Kyoto, Japan (2014)
43. Moeller, D., Paturi, R., Schneider, S.: Subquadratic algorithms for succinct stable matching. In: International Computer Science Symposium in Russia, pp. 294–308. Springer (2016)
44. Nekrich, Yakov: Orthogonal range searching in linear and almost-linear space. Comput. Geom. **42**(4), 342–351 (2009)
45. Patrascu, M., Williams, R.: On the possibility of faster SAT algorithms. In: SODA, volume 10, pp. 1065–1075. SIAM (2010)
46. Willard, D.E..: Predicate-oriented Database Search Algorithms. Center for Research in Computing Technology: Center for Research in Computing Technology. Garland Pub., (1979). URL: https://books.google.de/books?id=iLQmAAAAMAAJ
47. Williams, Ryan: A new algorithm for optimal 2-constraint satisfaction and its implications. Theor. Comput. Sci. **348**(2–3), 357–365 (2005)
48. Williams, R.: Faster decision of first-order graph properties. In: Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), pp. 80. ACM (2014)
49. Williams, V.V., Williams, R.: Subcubic equivalences between path, matrix and triangle problems. In: Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on, pp. 645–654. IEEE (2010)
50. Williams, V.V., Williams, R.: Finding, minimizing, and counting weighted subgraphs. SIAM J. Comput. **42**(3), 831–854 (2013)

Authors and Affiliations

Haozhe An[1] · Mohit Gurumukhani[2] · Russell Impagliazzo[3] · Michael Jaber[4] · Marvin Künnemann[5] · Maria Paula Parga Nina[3]

Haozhe An
haozhe@umd.edu

Russell Impagliazzo
russell@eng.ucsd.edu

Michael Jaber
mjjaber@cs.utexas.edu

Marvin Künnemann
marvin.kuennemann@eth-its.ethz.ch

Maria Paula Parga Nina
mpargani@ucsd.edu

1 University of Maryland, College Park, Maryland, USA

2 Cornell University, Ithaca, USA

3 UC San Diego, San Diego, USA

4 University of Texas at Austin, Austin, USA

5 Institute for Theoretical Studies, ETH Zurich, Switzerland