Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 53 (2016)

Orthogonal Vectors is hard for first-order properties on sparse graphs

Jiawei Gao*†
jiawei@cs.ucsd.edu

Russell Impagliazzo*†
russell@cs.ucsd.edu

Department of Computer Science and Engineering,
University of California, San Diego,
La Jolla, CA. 92093

Abstract

Fine-grained reductions, introduced by Vassilevska-Williams and Williams, preserve any improvement in the known algorithms. These have been used very successfully in relating the exact complexities of a wide range of problems, from NP-complete problems like SAT to important quadratic time solvable problems within P such as Edit Distance. However, until now, there have been few equivalences between problems and in particular, no problems that were complete for natural classes under fine-grained reductions. We give the first such completeness results. We consider the class of first-order graph property problems, viewing the input in adjacency list format (aka "sparse graph representation"). For this class, we show that the sparse Orthogonal Vectors problem is complete under randomized fine-grained reductions. In proving completeness for this problem, we also show that this sparse problem is equivalent to the standard Orthogonal Vectors problem when the number of dimensions is polynomially related to the number of vectors. Finally, we also establish a completeness and hardness result for k-Orthogonal Vectors.

Our results imply that the conjecture "not every first-order graph problem has an improved algorithm" is a useful intermediary between SETH and the conjectured hardness of problems such as Edit Distance. It follows that, if Edit Distance has a substantially subquadratic algorithm, then every first order graph problem has an improved algorithm. On the other hand, if first order graph property problems have improved algorithms, this falsifies SETH (and even some weaker versions of SETH) and gives new circuit lower bounds. We hope that this is the beginning of extending fine-grained complexity to include classes of problems as well as individual problems.

Keywords. orthogonal vectors, fine-grained complexity, first-order logic, model checking, graph properties

*This research is supported by NSF grant CCF-1213151 from the Division of Computing and Communication Foundations. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

†This work was done in part while the author was visiting the Simons Institute for the Theory of Computing.

ISSN 1433-8092

1 Introduction

Most of computational complexity has been aimed at making rather coarse distinctions between problems, separating those that are very expensive in terms of time or other resources from those that are moderate in cost. For example, in traditional complexity we distinguish between problems that are NP-hard, and so likely require exponential time, from those solvable in P. In contrast, "fine-grained complexity" aims at making finer distinctions between problems based on more exact quantitative evaluations of the required resources, such as distinguishing problems requiring $O(n^3)$ time from those solvable in $O(n^2)$ time. While fine-grained complexity is not a new idea (for a variety of approaches see e.g., [SHI90, GO95, DF92, NRS95, JS99, IPZ98], progress in recent years has been extremely rapid and impressive.

However, as the field has grown, it has become much more complex. Underlying progress are many conjectures concerning a vast variety of computational problems, such as ETH [IPZ98], SETH [IP99], the 3-SUM Conjecture [GO95], the Orthogonal Vectors Conjecture [Wil05], the APSP Conjecture [WW10], the Hitting Set Conjecture [AWW15], and conjectures about various versions of matrix multiplication (e.g., [HKNS15]). Unlike for coarse-grained complexity, where $\mathsf{P} \neq \mathsf{NP}$ is widely believed, there is no consensus about the truth or falsity of these conjectures. While each conjecture definitely represents an algorithmic challenge that has withstood much attention from algorithm designers, is is still very possible that several of the conjectures are false and will eventually fall to improved algorithm techniques. For example, many researchers in the area have stated their belief that SETH is in fact false, and have worked on disproving it.

As such conjectures proliferate, it becomes more difficult to discern how believable each conjecture is. There are three types of evidence that have been given for the various conjectures:

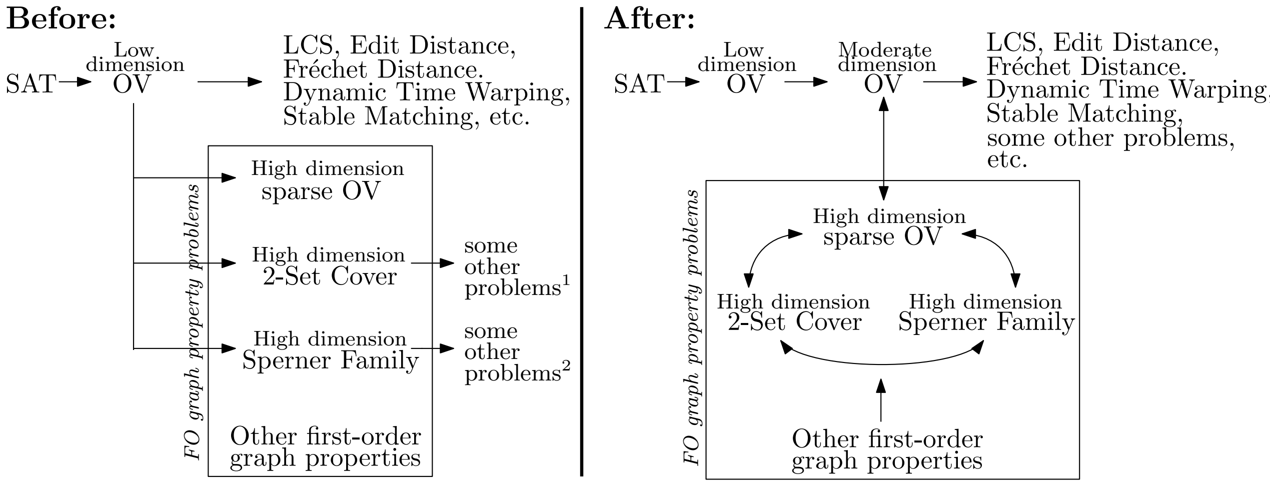
1. The amount of effort that has been put into (unsuccessfully) attempting to discover improved algorithms for the problem in question.
2. That the negation of the conjecture would have interesting complexity-theoretic consequences.
3. That the conjecture in question follows from other, more established, conjectures.

We feel the first two types of evidence are unreliable. The literature is full of unexpected algorithmic breakthroughs on well-studied problems (e.g., Babai's recent graph isomorphism algorithm). The main complexity-theoretic consequences of the failure of these hypotheses is to establish circuit lower bounds within NEXP or a similar class [Wil13, JMV15]. While the possibility of proving such a circuit lower bound via an improved algorithm makes such an algorithm more desirable, it doesn't actually argue for the impossibility of such algorithms, since the circuit bounds in question are almost certainly true. This leaves exploring implications between the various conjectures to tell which are the most likely.

The good news is that many such implications are known. The bad news is that there are relatively few equivalences ([WW10] being an exception), and the known implications seem to make the picture more complex rather than less. (In fact, [CGI$^+$16] presents a technical obstacle that often prevents proofs of such equivalences by direct reductions.)

Looking back at traditional complexity for guidance, we could hope to organize the problems that are the object of these conjectures into *classes* and identify *complete* problems for some of these classes. In this paper, we take a first step towards such a classification. However, the fine-grained world seems to be quite different from that for traditional complexity classes. Traditional complexity classes are typically defined in terms of models of computation and budgets of allowable resources, such as time, memory, non-determinism, randomness or parallelism. Complete problems then capture what is solvable in these models with limited resources available.

This is really useful in unifying two goals of complexity: a *problem-centric* goal of identifying the resources required to solve particular problems, and a *resource-centric* goal of deciding relationships



[1] includes 3-Dominating Set and Bipartite Subset 2-Dominating Set.
[2] includes Graph Dominated Vertex, Maximum Simple Family of Sets, and Subset Graph.

Figure 1: A diagram of reductions. We simplify this picture, and make the reductions to Edit Distance, LCS, etc. more meaningful. (The "other problems" are problems given in [BCH14]. Many of them are also first-order graph properties.)

between the computational power of different models of computation with different resource bounds. If problem Π is complete for the class of problems $C_{M,R}$ solvable in model M with resource limit R, then Π is solvable in model M' with resource limit R' if and only if $C_{M,R}$ is contained in $C_{M',R'}$. Unfortunately, a resource-centric taxonomy seems to be inadequate for fine-grained complexity. Some of the strongest results are obtained by reducing very hard problems (such as SAT, an NP-complete problem) to problems solvable in very low-level complexity classes (such as the AC^0-computable Orthogonal Vectors problem that we will be considering). This is possible because the reductions increase the size of the problems substantially, but is counter-intuitive and makes divisions of conjectures into classes based on resources and model strength difficult. In principle, we could argue that, e.g., a problem is complete under linear-time reductions for the class defined by a larger polynomial-time bound and evoke the Time-Hierarchy Theorem to prove a strong polynomial lower bound on its complexity. However, this does not seem to be the case for any of the natural problems that have arisen so far.

Instead, we here look to *descriptive complexity* as opposed to *resource-oriented complexity*, and consider classes defined in terms of the *logical form* of the problems to be solved. In particular, we consider the class of all *first-order graph properties*, a rich but structured class of problems. As shown in Figure 1, we identify several complete problems for this class (under randomized fine-grained reductions defined in Section 2.1). In fact, these problems are equivalent to a version of the previously studied Orthogonal Vectors problem. It follows from standard arguments that these complete problems are intermediates between SETH and many of the hardness results proved assuming SETH, such as near-quadratic hardness for Edit Distance [BI15], Fréchet Distance [Bri14], LCS and Dynamic Time Warping [ABW15, BK15], and Succinct Stable Matching [MPS15]. Thus, if say Edit Distance were solvable in substantially subquadratic time, not only would an improved algorithm follow for SAT, but improved algorithms would follow for *every* problem in a broad, natural class of graph problems. (And also we would get circuit lower bounds by [Wil13, JMV15].) This seems to us strong evidence in believing that algorithms for these problems cannot be improved. (Our results are similar in spirit but incomparable to those of [AHWW15] , who give implications to stronger forms of SAT algorithms.) Our completeness result also simplifies the tree structure of reductions between many SETH-hard problems given in [BCH14].

In addition to having a very natural and useful complete problem, the class we analyse, first-order graph property problems, is important in itself. While we call the class "graph problems", our class can model any property that involves any types of binary and unary relations, such as set families with the element relation. In this way, this class includes many of the problems considered previously in the fine-grained complexity literature, such as Hitting Set, Orthogonal Vectors, Sperner Family, Diameter 2, Radius 2, k-Independent Set, k-Dominating Set and so on. (Unfortunately, it does not include numerical problems such as k-SUM.) Secondly, this class has been very well-studied in both the complexity and combinatorics literature. For example, the first zero-one law for random graphs was proved for first-order graph properties on finite models ([Fag76]), and Ajtai's lower bound for AC^0 [Ajt83] (proved independently by Furst, Saxe, and Sipser ([FSS84])) was motivated and stated as a result about inexpressibility in first-order logic. First-order properties are also widely used as the uniform version of AC^0 in the complexity literature. Finally, algorithms for model-checking first-order properties are very important to *query evaluation* in databases. The core of the fundamental relational database language *SQL* is equivalent to first-order properties. So understanding the limits of the best possible algorithms for various types of first-order properties illuminates the worst-case performance of any database system in evaluating these queries. (There are some differences between our setting and the typical database setting. Query evaluation processes are not allowed to use arbitrary algorithms, only those representable as series of queries. Also, while many database systems convert higher arity relations to binary relations as a first step, doing so changes the quantifier structure of the query, so our results are not directly applicable to queries with higher arity relations.)

In fine-grained complexity, since we are talking about exact complexities, problem representation is significant. For graph problems, there are two standard representations: adjacency lists (which are a good fit for *sparse* graphs) and adjacency matrices (good for *dense* graphs). The main difference between the two is whether we perform the analysis in terms of the number of edges m or the number of vertices n. For several reasons, we study the adjacency list, or sparse graph, representation. First, many of the problems considered such as Orthogonal Vectors have hard instances that are sparse. Secondly, the complexity of problems in the dense model is somewhat unclear, at least for numbers of quantifiers between 3 and 7 ([Wil14]). Third, the sparse model is more relevant for model checking, since the input to database problems is given as a list of tuples.s

We also build on recent work by Ryan Williams looking at the dense case of first-order graph properties [Wil14] and of Abboud et al. [AWW15] which reduces the Hitting Set problem to the Orthogonal Vectors problem.

1.1 Orthogonal Vectors, and conjectures

The *Orthogonal Vectors problem (OV)* gives a set A of n Boolean vectors of dimension d, and decides if there exist vectors $u, v \in A$ such that u and v are orthogonal, i.e., $u[i] \cdot v[i] = 0$ for all indices $i \in \{1, \dots, d\}$. Another equivalent version is to decide in two sets A and B of Boolean vectors, each of size n, whether there exist $u \in A$ and $v \in B$ so that u and v are orthogonal. A naïve algorithm for OV runs in time $O(n^2 d)$, and the best algorithm runs in $O(n^{2-\Omega(1/\log(d/\log n))})$ [AWY15].

The popular hardness conjectures on OV usually specify dimension d to be between $\omega(\log n)$ and $n^{o(1)}$, so we call them low-dimension OVC. In contrast to the dense model of OV defined above, where the vectors are given explicitly (thus analogous to the adjacency matrix representation of graphs), this paper introduces a sparse version of OV, where the input is a list of vector-index pairs (v, i) for each $v[i] = 1$ (which corresponds to the adjacency list representation of graphs). In sparse

OV, the length of input equals the sum of Hamming weight over all vectors. Here we do not restrict the dimension d of vectors. So we call this problem "sparse high-dimension OV", or "sparse OV" for short.

Based on the size of d, we give three versions of Orthogonal Vector Conjectures. In all three conjectures the complexity is measured in the word RAM model with $O(\log n)$ bit words.

Low-dimension OVC (LDOVC): $\forall \epsilon > 0$, there is no $O(n^{2-\epsilon})$ time randomized algorithm for OV with dimension $d = \omega(\log n)$.

Moderate-dimension OVC (MDOVC): $\forall \epsilon > 0$, there is no $O(n^{2-\epsilon}\mathsf{poly}(d))$ time randomized algorithm that solves OV with dimension d. (Although dimension d is not restricted, we call it "moderate dimension" because such an algorithm only improves on the naive algorithm if $d = n^{O(\epsilon)}$.)

Sparse high-dimension OVC (SHDOVC): $\forall \epsilon > 0$, there is no $O(m^{2-\epsilon})$ time randomized algorithm for sparse OV where m is the total Hamming weight of all the input vectors.

SETH implies LDOVC [Wil05]. Because MDOVC is an extension of LDOVC, it can be implied from the latter. Like LDOVC, MDOVC can also imply the hardness of problems including Edit Distance, LCS, etc. In this paper we will further show MDOVC and SHDOVC are equivalent.

OV can be extended to the k-OV problem for any integer $k \geq 2$, that gives k sets $A_1, \ldots, A_k$ of Boolean vectors, and asks if there exist k different vectors $v_1 \in A_1, \ldots, v_k \in A_k$ so that for all indices i, $\prod_{j=1}^{k} v_j[i] = 0$. We also introduce a sparse version of k-OV similar to sparse OV, where all the ones in the vectors are given in a list. In first-order logic, the sparse k-OV problem can be expressed by $(\exists v_1 \in A_1) \ldots (\exists v_k \in A_k)(\forall i) \left[\bigvee_{j=1}^{k} \neg(v_j[i] = 1)\right]$.

1.2 First-order graph property problems

The problem of deciding whether a structure satisfies a logical formula is called the model checking problem. It is well-studied in finite model theory. In relational databases, first-order model checking plays an important role, because first-order queries capture the expressibility of relational algebra. In contrast to the *combined complexity*, where the database and query are both given as input, the *data complexity* studies the running time when the query is fixed. The combined complexity of first-order queries is PSPACE-complete, but the data complexity is in LOGSPACE [Var82]. Moreover, these problems are also major topics in parameterized complexity theory. [FG06] organizes parameterized first-order model checking problems (many of which are graph problems) into hierarchical classes based on their quantifier structures. Our work will study the model checking problems in a more fine-grained manner.

A graph can be considered as a logical structure with only unary and binary predicates. A first-order graph property problem is to decide whether an input graph satisfies a fixed first-order formula with only unary and binary predicates and no free variables. So it is a special type of model checking problem. Below we list some examples.

- Many classical graph problems are first-order expressible, including[1]:
  1. Graph Diameter-2: $(\forall x_1)(\forall x_2)(\exists x_3) [E(x_1, x_3) \wedge E(x_3, x_2)]$
  2. Graph Radius-2: $(\exists x_1)(\forall x_2)(\exists x_3) [E(x_1, x_3) \wedge E(x_3, x_2)]$
  3. k-Clique: $(\exists x_1) \ldots (\exists x_k) \left[\bigwedge_{i,j \in \{1,\ldots,k\}, i \neq j} E(x_i, x_j)\right]$. More generally, for a fixed graph H of k vertices, deciding if H is a subgraph or induced subgraph of the input graph G (e.g., the k-Independent Set problem) can be expressed in a similar way.

[1] Diameter-2 and radius-2 are not "artificial": easy approximation algorithms for these problems would respectively refute the OV Conjecture and the Hitting Set Conjecture [AWW15].

4. k-Dominating Set: $(\exists x_1)\dots(\exists x_k)(\forall x_{k+1})\left[\bigvee_{i=1}^{k} E(x_i,x_{k+1})\right]$.

- Many non-graph problems defined by first-order formulas with only unary and binary relations can also be considered as graph problems. OV and k-OV, of course, are examples of these problems. If we consider the relation "$\in$" as a binary predicate, we also have:
  1. The Hitting Set problem, where all the sets are given explicitly in a set family $\mathcal{S}$: $(\exists H \in \mathcal{S})(\forall S \in \mathcal{S})(\exists x)\,[(x \in H) \wedge (x \in S)]$. (Other versions of Hitting Set where the sets are not given explicitly, are second-order logic problems. Our definition here is consistent with the version in the Hitting Set Conjecture.)
  2. The k-Set Packing problem, where all the sets are given explicitly in a set family $\mathcal{S}$: $(\exists S_1 \in \mathcal{S})\dots(\exists S_k \in \mathcal{S})(\forall x)\left[\bigvee_{i=1}^{k}\left((x \in S_i) \to \bigwedge_{j\neq i}(x \notin S_j)\right)\right]$.
  3. k-Empty Intersection, k-Set Cover, Set Containment and Sperner Family, defined in Section 2.2.

Let φ be a first-order sentence containing only unary and binary predicates. Assume φ is in prenex normal form with $(k+1) \geq 3$ distinct quantified variables and no free variables, i.e., $\varphi = (Q_1x_1)\dots(Q_{k+1}x_{k+1})\,\psi(x_1,\dots,x_{k+1})$, in which $Q_i \in \{\exists,\forall\}$, and ψ is quantifier-free. A first-order graph property problem (which is a model checking problem) denoted by MC_φ, is to decide whether an input graph $G=(V,E)$ satisfies φ (i.e., whether $G \models \varphi$), where $x_1,\dots,x_{k+1} \in V$. Each binary predicate R in φ corresponds to a subset E_R of edges, such that $R(x_i,x_j) = \mathit{true}$ iff $(x_i,x_j) \in E_R$. Each unary relation in φ can also be considered as a set of arity-one edges (or self-loops). So for simplicity we refer to both unary and binary relations as "edges". The input graph G is given by a list of all unary and binary edges. Let n be the total number of vertices, and m be the number of edges in G. We assume that $m \geq n$ and that every vertex is in an edge. So the input length is m, and thus we can replace n with m in all complexity upper bounds. We also assume $m = n^{1+o(1)}$ (the graph is always sparse). This assumption is without loss of generality, by the argument in Section 3.1.

We use $MC(k)$ for the class of graph property problems MC_φ where φ has k quantifiers (thus k variables, since φ is in prenex normal form). We also use the notations of form $MC(Q_1\dots Q_{k+1})$ to represent the class of problems MC_φ where $\varphi = (Q_1x_1)\dots(Q_{k+1}x_{k+1})\,\psi(x_1,\dots,x_{k+1})$. Besides, we will use $\exists^c$ and $\forall^c$ to represent c consecutive quantifiers. For example, $MC(\exists^k\forall) = \{MC_\varphi \mid \varphi = (\exists x_1)\dots(\exists x_k)(\forall x_{k+1})\,\psi(x_1,\dots,x_{k+1})\}$.

An obvious fact is that MC_φ and $MC_{\neg\varphi}$ are reducible to each other, by negating the answer.

Finally we state a conjecture on the hardness of the first-order graph property problems. Again we measure the complexity in the word RAM model with $O(\log n)$ bit words.

First-order graph property conjecture (FOC): There is some integer $k \geq 2$, so that $\forall\epsilon > 0$, some problem in $MC(k+1)$ cannot be solved by any randomized algorithm in $O(m^{k-\epsilon})$ time.

1.3 Main Results

First, we show that conjectures for OV defined on dense and sparse models are equivalent under randomized reductions, which means MDOVC is true iff SHDOVC is true. (The lemma is implied by Lemma 4.2 in Section 4.1.)

Lemma 1.1. *For any integer $k \geq 2$, there exist $\delta, \epsilon > 0$ and an $O(n^{k-\epsilon})$ time randomized algorithm solving k-OV with dimension $d = n^\delta$, if and only if there is some $\epsilon' > 0$ and an $O(m^{k-\epsilon'})$ time randomized algorithm solving sparse k-OV with m being the total Hamming weight of all the input vectors.*

Our main result establishes an equivalence of MDOVC and FOC, showing the completeness of sparse OV, and hardness of dense OV, for the class of first-order graph property problems.

Theorem 1. *The following two propositions are equivalent:*

(A) There exist $\delta, \epsilon > 0$ so that OV of dimension $d = n^\delta$ can be solved in randomized time $O(n^{2-\epsilon})$. (i.e., MDOVC is false)

(B) For any integer $k \geq 2$, for any first-order property L with unary and binary relations expressible with $k+1$ quantifiers in prenex normal form, there exists $\epsilon > 0$ so that L can be decided in randomized time $O(m^{k-\epsilon})$. (i.e., FOC is false)

Besides, this paper will also prove a hardness and completeness result for k-OV, connecting one combinatorial problem to a large and natural class of logical problems. Using the notion of fine-grained reductions, the following theorem indicates that sparse k-OV (and therefore also dense k-OV) is complete for $MC(\exists^k\forall)$ (and its negation form $MC(\forall^k\exists)$), and hard for $MC(\forall\exists^{k-1}\forall)$ (and its negation form $MC(\exists\forall^{k-1}\exists)$) under randomized fine-grained reductions.

Theorem 2. *If sparse k-OV with total Hamming weight m can be solved in randomized $O(m^{k-\epsilon})$ time for some $\epsilon > 0$, then all the problems in $MC(\exists^k\forall)$, $MC(\forall^k\exists)$, $MC(\forall\exists^{k-1}\forall)$ and $MC(\exists\forall^{k-1}\exists)$ are solvable in randomized time $O(m^{k-\epsilon'})$ for some $\epsilon' > 0$.*

$MC(\exists^k\forall)$ and $MC(\forall^k\exists)$ are interesting sub-classes of $MC(k+1)$: If Nondeterministic SETH is true, then all the SETH-hard problems in $MC(k+1)$ are contained in $MC(\exists^k\forall)$ or $MC(\forall^k\exists)$ ([CGI+16]).

We will also show that the 2-Set Cover problem and the Sperner Family problem, both in $MC(\exists\exists\forall)$, are equivalent to sparse OV under randomized reductions, and thus hard for first-order graph property problems.

1.4 Organization of this paper

In Section 2, we define the fine-grained reductions, and present the high-level ideas for techniques of reducing from a first-order graph property problem to OV. Section 3 outlines the proofs for Theorem 1 and Theorem 2. We present the reduction from $MC(\exists^k\forall)$ to k-OV in Section 4. And then we present the reduction from $MC(\forall\exists^{k-1}\forall)$ to k-OV in Section 5. Finally in Section 6 we talk about open problems. Appendix A gives a baseline algorithm for $MC(k+1)$ with time complexity $O(n^{k-1}m)$. Appendix B solves the easy cases in $MC(k+1)$ by giving $O(m^{k-1/2})$ algorithms for them.

2 Preliminaries

2.1 Fine-grained reductions

To formalize the reductions, we use the notion *fine-grained reductions*, which was introduced by Vassilevska-Williams [Wil]. In these reductions, we carefully preserve the conjectured time complexities of different problems. Assume L_1 and L_2 are languages and T_1 and T_2 are their conjectured running time lower bounds respectively.

Definition 2.1 (Fine-grained Turing reduction ($\leq_{FGT}$))**.** $(L_1, T_1) \leq_{FGT} (L_2, T_2)$ if for any $\epsilon > 0$, there exists $\epsilon' > 0$, and an algorithm running in time $T_1(n)^{1-\epsilon'}$ on input of length n. The algorithm makes q calls to oracle of L_2 with query lengths $n_1, \ldots, n_q$, such that $\sum_{i=1}^{q}(T_2(n_i))^{1-\epsilon} \leq (T_1(n))^{1-\epsilon'}$.

Then, if L_2 has an algorithm substantially faster than T_2, L_1 can be solved substantially faster than T_1. In almost all fine-grained reductions, $T_1 \geq T_2$, i.e., we usually reduce from harder problems to easier problems, which may seem counter-intuitive. A harder problem L_1 can be reduced to a easier problem L_2 with $T_1 > T_2$ in two ways.

1. The reduction makes multiple calls to an algorithm solving L_2.
2. The reduction blows up the size of the L_2 instance.[2] (e.g., the reduction from CNF-SAT to OV is an example of this technique.)

All the reductions from higher complexity to lower complexity problems in this paper belong to the first type.

Note that in the case when $T_1 = T_2$, we cannot blowup the size of the problem instances, or the query lengths by even a small polynomial factor. This is an important point, thus we emphasize this case by defining fine-grained mapping reduction from L_1 to L_2 on running time $T = T_1 = T_2$.

Definition 2.2 (Fine-grained mapping reduction ($\leq_{FGM}$)). $L_1 \leq^T_{FGM} L_2$ if for any $\delta > 0$, there exists $\epsilon' > 0$, and algorithm running in time $T(n)^{1-\epsilon'}$ on input x of length n, computing $f(x)$ of length $O(n^{1+\delta})$, so that $x \in L_1$ iff $f(x) \in L_2$.

Then if $L_1 \leq^T_{FGM} L_2$ and L_2 is solvable in time $O(T(n)^{1-\epsilon})$ for some $\epsilon > 0$, then we pick $\delta < \epsilon$ so that L_1 is solvable in time $O(T(n)^{1-\epsilon'} + T(n^{(1+\delta)(1-\epsilon)})) = O(T(n)^{1-\epsilon'} + T(n)^{1-(\epsilon-\delta)})$. This is why we need to be able to create instances whose size is as small as n^δ for arbitrarily small $\delta > 0$. We can also similarly define randomized fine-grained reductions $\leq_{rFGT}$ and $\leq_{rFGM}$, where the reduction algorithms are randomized.

2.2 Sparsity and co-sparsity

This section gives an intuitive and high-level view about the techniques of reducing a first-order graph property problem to OV, for the proof of Theorem 1 and Theorem 2. Because of Lemma 1.1, in the remainder of this paper, unless specified, we will use "OV" and "k-OV" to refer to sparse versions of these problems. The sparse k-OV problem can be reformulated as the k-Empty Intersection (k-EI) problem, where sets correspond to vectors and elements correspond to dimensions:

Problem: k-Empty Intersection (k-EI)(Equivalent to k-OV.)

Input: A universe U of size n_u, and k families of sets $\mathcal{S}_1 \dots \mathcal{S}_k$ on U, of size $n_1, \dots n_k$.
Output: Whether there exist $S_1 \in \mathcal{S}_1, \dots, S_k \in \mathcal{S}_k$ such that $\bigcap_{i=1}^k S_i = \emptyset$.
Logical expression: $\varphi = (\exists S_1 \in \mathcal{S}_1) \dots (\exists S_k \in \mathcal{S}_k)(\forall u \in U) \left[\bigvee_{i=1}^k \neg(u \in S_i)\right]$.

Next, we introduce two similar problems that act as important intermediate problems in our reduction process.

Problem: Set Containment (Equivalent[3] to Sperner Family.)

Input: A universe U of size n_u, and two families of sets $\mathcal{S}_1, \mathcal{S}_2$ on U, of size n_1, n_2.
Output: Whether there exist $S_1 \in \mathcal{S}_1, S_2 \in \mathcal{S}_2$ such that $S_1 \subseteq S_2$.
Logical expression: $\varphi = (\exists S_1 \in \mathcal{S}_1)(\exists S_2 \in \mathcal{S}_2)(\forall u \in U) [(\neg(u \in S_1)) \vee (u \in S_2)]$.

Problem: k-Set Cover (Equivalent to k-Dominating Set.)

Input: A universe U of size n_u, and k families of sets $\mathcal{S}_1 \dots \mathcal{S}_k$ on U, of size $n_1, \dots n_k$.
Output: Whether there exist $S_1 \in \mathcal{S}_1, \dots, S_k \in \mathcal{S}_k$ such that $\bigcup_{i=1}^k S_i = U$.
Logical expression: $\varphi = (\exists S_1 \in \mathcal{S}_1) \dots (\exists S_k \in \mathcal{S}_k)(\forall u \in U) \left[\bigvee_{i=1}^k (u \in S_i)\right]$.

[2]Actually it is harder to fine-grained reduce from a problem with lower time complexity to a problem with higher time complexity (e.g., prove that $(MC(k), m^{k-1}) \leq_{FGT} (MC(k+1), m^k))$, because this direction often needs creating instances with size much smaller than the original instance size.

[3]Equivalent under linear-time reductions. It is the same for the k-Set Cover problem below.

All these problems are first-order graph property problems: we can use unary predicates to partition the vertex set into $(\mathcal{S}_1, \dots, \mathcal{S}_k, U)$, and consider the relation "$\in$" as a binary predicate. We let n (corresponding to the number of nodes in the input graph) be the sum of $n_1, \dots, n_k$ and n_u, and let the input size m (corresponding to the number of edges in the input graph) be the sum of all sets' sizes in all set families. We call 2-Set Cover, Set Containment and OV(or equivalently 2-EI), the *Basic Problems*, which will be formally defined and generalized in Section 4.1. [BCH14] proved that when $k = 2$, these Basic Problems require time $m^{2-o(1)}$ under SETH, and that if the size of universe U is polylogarithmic to the input size, then the three problems are equivalent under subquadratic-time reductions. The main idea of the reductions between these problems is to complement all sets in $\mathcal{S}_1$, or $\mathcal{S}_2$, or both. It is easy to see that $S_1 \cap S_2 = \emptyset \iff {S_1}^{\complement} \cup {S_2}^{\complement} = U \iff S_1 \subseteq {S_2}^{\complement} \iff S_2 \subseteq {S_1}^{\complement}$. Therefore, if we could complement the sets, we can easily prove the equivalence between the three Basic Problems. However we cannot do this when n_u is large.

For a sparse binary relation like $(u \in S_1)$, we say its complement, like $(u \notin S_1)$, is *co-sparse*. Suppose we want to enumerate all tuples (S_1, u) s.t. $u \in S_1$, we can go through all relations (aka edges) between U and $\mathcal{S}_1$. So we can do this in linear time. On the contrary, if we want to enumerate all pairs (S_1, u) s.t. $u \notin S_1$, we cannot do this in linear time, because we cannot touch the pairs by touching edges between them. What is even worse, when n_u is as large as n, the number of such pairs can reach m^2. When $k = 2$, a fine-grained mapping reduction between m^2-time problems allows neither quadratic time reductions, nor quadratic size problem instances. Essentially, a major technical obstacle in our reductions is to efficiently deal with co-sparsity.

Switching between sparsity and co-sparsity. Because of the above argument, it is hard to directly reduce between the Basic Problems, so instead we reduce each problem to a highly-asymmetric instance of the same problem, where sparse relations are easily complemented to relations that are also sparse. Observe that when the size of U is m^δ for some $\delta < 1$, complementing all sets can be done in $O(m^{1+\delta})$, which is substantially faster than $O(m^2)$. The new instance created also has size $O(m^{1+\delta})$. If we can do this for arbitrarily small $\delta > 0$, we can make it a fine-grained mapping reduction. Using this technique which we call *universe-shrinking self-reduction*, we can show that OV, 2-Set Cover and Set Containment are equivalent under $\leq_{FGM}^{m^2}$.

Claim 2.1. *If any one of OV, 2-Set Cover and Set Containment (or Sperner Family) has subquadratic time randomized algorithms, then the other two are also solvable in randomized subquadratic time. Thus the three problems are all hard for $MC(k)$ with $k \geq 3$.*

This claim itself is an interesting result: in [BCH14], conditional lower bounds for many problems stem from the above three problems, forming a tree of reductions. By the equivalence result, the root of the tree can be replaced by the quadratic-time hardness conjecture on any of the three problems, thus the reduction tree is simplified.

The above claim is a special case of Lemma 4.1. In Section 4 we will prove a more general version of equivalence. Note that the universe-shrinking self-reduction is the only randomized step. All the other reductions in this paper are deterministic.

Dealing with co-sparsity. Having been able to reduce between the three Basic Problems, what should we do for general problems with arbitrary formulas? The detailed processes are complicated, so here we talk about a high-level idea in not only reductions but also algorithm design throughout the paper.

Our algorithms often need to iterate over all pairs (x_i, x_j) satisfying some conditions, so as to get the pairs, or to count the number of them. These "conditions" we need are first-order. So these algorithms can be considered as *query processing.* A set of pairs (x_i, x_j) can be considered as the result of a *first-order query* defined by an intermediate formula φ' on the graph G (or some intermediate structures). What our reduction algorithms usually do is to generate such queries, evaluate the queries, and use the results in the future process.

For any such query, there are three cases. If the result of the query is a sparse relation like $[\neg R_1(x_1, x_2) \wedge R_2(x_1, x_2)]$, we can iterate over them (say, first enumerate all edges in E_{R_2} so that $R_2(x_1, x_2)$ is true, and then check if $R_1(x_1, x_2)$ is false). Then, we can do further operations on these (x_1, x_2) tuples resulted from the query. When the result of the query is a co-sparse relation like $[\neg R_1(x_1, x_2) \wedge \neg R_2(x_1, x_2)]$, we cannot directly iterate over them. So we work on its complement (which is sparse, instead of co-sparse), but then do some further processing to filter those pairs out from future use (say, work on all edges in $E_{R_1} \cup E_{R_2}$ so that $R_1(x_1, x_2)$ is true or $R_2(x_1, x_2)$ is true, then exclude those pairs from future use). Sometimes, the result of a query is neither sparse nor co-sparse, but we will see it is always a combination of sparse and co-sparse relations. Thus we need to distinguish them and deal with the sparse and co-sparse parts separately, which will be explained next.

Separating sparse and co-sparse relations. We exemplify this technique by considering the query $[\neg R_1(x_1, x_2) \vee \neg R_2(x_1, x_2)]$. For a pair (x_1, x_2), to make the formula true, predicates R_1, R_2 can be assigned values from $\{(\mathit{True}, \mathit{False}), (\mathit{False}, \mathit{True}), (\mathit{False}, \mathit{False})\}$. In the first two cases, the pairs (x_1, x_2) satisfying $[R_1(x_1, x_2) \wedge \neg R_2(x_1, x_2)]$ and $[\neg R_1(x_1, x_2) \wedge R_2(x_1, x_2)]$ are sparse, while in the last case, the pairs satisfying $[\neg R_1(x_1, x_2) \wedge \neg R_2(x_1, x_2)]$ are co-sparse. So if we want to work on the tuples satisfying this query, we work on tuples satisfying the first two cases directly by enumerating edges, and then work on the tuples *not* satisfying the third case (i.e., the tuples where either $R_1(x_1, x_2)$ or $R_2(x_1, x_2)$ is true), in order to exclude them from future use.

In general, a query can be written as a DNF, where the result of each term is a conjunction of predicates and negated predicates, and therefore either sparse or co-sparse. Then we can deal with the sparse and co-sparse cases separately. We will use this technique for constructing the Hybrid Problem in Section 4.2 (where the "future use" refers to "constructing gadgets from these pairs"), and for the baseline algorithm presented in Appendix A (where the "future use" refers to "counting the number of these pairs").

3 Proof Overview

3.1 $O(n^{k-1}m)$ Baseline algorithm

We will present a $O(n^{k-1}m)$ baseline algorithm solving $MC(k+1)$ for $k \geq 1$, in Appendix A. So we can solve any $(k+1)$-quantifier problem in time $O(m^k)$, which matches our conjectured lower bound. A central step used in the algorithm is the following lemma, called the *quantifier-eliminating downward reduction*, that will be proved in Appendix A.

Lemma 3.1 (Quantifier-eliminating downward reduction for $MC(k+1)$)**.** *Let the running time of $MC(k+1)$ on graphs of n vertices and m edges be $T_k(n, m)$. We have the recurrence*

$$T_k(n, m) \leq n \cdot T_{k-1}(n, O(m)) + O(m), \ \textit{for } k \geq 2.$$
$$T_1(n, m) = O(m).$$

By this lemma, for any problem $L_1 \in MC(k+1)$, there exists a problem $L_2 \in MC(k)$ such that $(L_1, m^k) \leq_{FGT} (L_2, m^{k-1})$. (See Appendix A.)

For problems in $MC(2)$, the algorithm runs in $O(m)$ time, and cannot be further improved. Therefore this paper considers two-quantifier problems as trivial cases, and only talks about problems with at least three quantifiers.

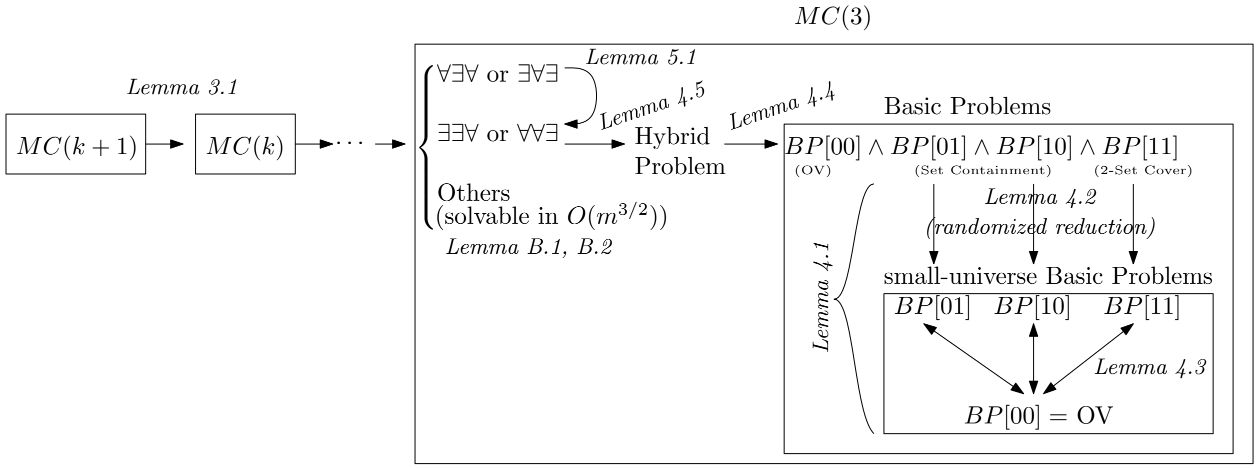


Figure 2: Overview of the reduction process for Theorem 1.

Also, in our definition of first-order graph property problems, it is safe to assume $m \leq n^{1+\epsilon}$ for any $\epsilon > 0$, for otherwise we can run the baseline algorithm in $O(n^{k-1}m)$ time to beat m^k time.

3.2 Completeness of OV

Following is the outline of reduction from a problem in $MC(k+1)$ to OV for any integer $k \geq 2$, thus proving the direction from (A) to (B) in Theorem 1. The direction from (B) to (A) is straightforward, because sparse OV is in $MC(3)$.

1. Using the quantifier-eliminating downward reduction in Lemma 3.1, reduce from the $(k+1)$-quantifier problem down to a 3-quantifier problem.
2. Based on the exact quantifier structure,
   - $MC(\exists\exists\exists)$, $MC(\forall\forall\forall)$, $MC(\forall\exists\exists)$ and $MC(\exists\forall\forall)$ are solvable in $O(m^{3/2})$, using algorithms in Appendix B.
   - For $MC(\forall\exists\forall)$ or its negation $MC(\exists\forall\exists)$, reduce the problem to $MC(\exists\exists\forall)$ using Lemma 5.1, and then reduce it to the Hybrid Problem using Lemma 4.5.
   - For $MC(\exists\exists\forall)$ or its negation $MC(\forall\forall\exists)$, reduce the problem to the Hybrid Problem, using Lemma 4.5.
3. Reduce from the Hybrid Problem to a combination of 4 Basic Problems, using Lemma 4.4.
4. Reduce all Basic Problems to OV, using Lemma 4.1: First do universe-shrinking self-reductions on each Basic Problem (Lemma 4.2), and then complement the sets and get OV (Lemma 4.3).

Figure 2 shows a diagram of the above reduction process.

Moreover, Lemmas 5.1, 4.5, 4.4 and 4.1 also work for any constant $k \geq 2$. So for a $MC(\exists^k\forall)$ or $MC(\forall\exists^{k-1}\forall)$ problem, we can reduce it to k-OV as follows:

1. If the problems belongs to $MC(\forall\exists^{k-1}\forall)$, reduce it to $MC(\exists^k\forall)$ using Lemma 5.1.
2. Reduce the $MC(\exists^k\forall)$ problem to the Hybrid problem, using Lemma 4.5.
3. Reduce from the Hybrid Problem to a combination of 2^k Basic Problems, using Lemma 4.4.
4. Reduce all Basic Problems to k-OV, using Lemma 4.1

Thus we have proved Theorem 2.

4 Completeness of k-OV in $MC(\exists^k\forall)$

This section will prove the completeness of k-OV in $MC(\exists^k\forall)$ problems. First, we introduce a class of Basic Problems, and prove these problems are equivalent to k-OV under $\leq_{FGM}^{m^k}$. Then, we show that any problem in $MC(\exists^k\forall)$ can be reduced to a combination of Basic Problems (aka. the Hybrid Problem).

4.1 How to complement a sparse relation: Basic Problems, and reductions between them

In this section we define the Basic Problems, which have similar logical expressions to k-OV (or k-EI), k-Set Cover and Set Containment problems. We will prove that these problems are fine-grained reducible to each other.

Let $k \geq 2$. We introduce 2^k Basic Problems labeled by k-bit binary strings from 0^k to 1^k. The input of these problems is the same as that of k-EI defined in Section 2.2: k set families $\mathcal{S}_1 \dots \mathcal{S}_k$ of size $n_1, \dots, n_k$ on a universe U of size n_u. We define 2^k quantifier-free formulas $\psi_{0^k}, \dots, \psi_{1^k}$ such that

$$\psi_\ell = \left(\bigvee_{i \in \{1,\dots,k\}, \ell[i]=0} (\neg(u \in S_i))\right) \vee \left(\bigvee_{i \in \{1,\dots,k\}, \ell[i]=1} (u \in S_i)\right).$$

Here, $\ell[i]$, the i-th bit of label ℓ, specifies whether u is in each S_i or not, in the i-th term of ψ_ℓ.

For each ℓ, let $\varphi_\ell = (\exists S_1 \in \mathcal{S}_1) \dots (\exists S_k \in \mathcal{S}_k)(\forall u \in U)\psi_\ell$. For simplicity, we will omit the domains of the variables in these formulas. We call $MC_{\varphi_{0^k}}, \dots, MC_{\varphi_{1^k}}$ the Basic Problems. We refer to the Basic Problem MC_{φ_ℓ} as $BP[\ell]$. These problems are special cases of first-order model checking on graphs, where sets and elements correspond to vertices, and membership relations correspond to edges. Note that $BP[0^k]$ is k-EI, and $BP[1^k]$ is k-Set Cover. When $k = 2$, $BP[01]$ and $BP[10]$ are Set Containment problems. For a k-tuple $(S_1 \in \mathcal{S}_1, \dots, S_k \in \mathcal{S}_k)$ satisfying $(\forall u)\psi_\ell$, we call it a *solution* of the corresponding Basic Problem $BP[\ell]$.

We present a randomized fine-grained mapping reduction between any two Basic Problems, thus proving the following lemma, which is a generalized version of Claim 2.1.

Lemma 4.1. *For any $\ell_1, \ell_2 \in \{0,1\}^k$, there is a randomized fine-grained mapping reduction $BP[\ell_1] \leq_{rFGM}^{m^k} BP[\ell_2]$.*

For problems $BP[\ell_1]$ and $BP[\ell_2]$ where ℓ_1 and ℓ_2 only differ in the i-th bit, if we are allowed to complement all sets in $\mathcal{S}_i$, we can easily reduce between them. Similarly, if ℓ_1 and ℓ_2 differ in more than one bit, we can complement all the sets in corresponding set families. However, complementing the sets in $\mathcal{S}_i$ takes time $O(n_i n_u)$, which might be as large as m^2. To solve this, we self-reduce $BP[\ell_1]$ on the universe U to the same problem on a smaller universe U', and then complement sets on U'. For any given δ, if the size of U' is $n'_u = O(m^\delta)$, then complementing all sets in $\mathcal{S}_i$ only takes time and space $m \cdot O(m^\delta) = O(m^{1+\delta})$.

Lemma 4.2 (Universe-shrinking self-reductions of Basic Problems)**.** *Let label ℓ be any binary string in $\{0,1\}^k$. For any $\epsilon > 0$, given a $BP[\ell]$ instance I of size m and universe U of size n_u, we can either solve it in time $O(m^{k-\epsilon})$, or use time $O(m^{k-\epsilon})$ to create a $BP[\ell]$ instance I' of size $O(m^{1+\epsilon})$ on universe U' whose size is $n'_u = O(m^{5\epsilon})$, so that $I \in BP[\ell]$ iff $I' \in BP[\ell]$ with error probability bounded by $O(m^{-\epsilon})$.*

Note that the self-reduction of k-OV actually reduces the sparse OV to the dense and low-dimension version of OV, implying Lemma 1.1.

We will present the randomized self-reductions for problems $BP[\ell]$ s.t. $\ell \neq 1^k$ in Section 4.1.1. For $BP[1^k]$, we will prove that it is either easy to solve or easy to complement in Section 4.1.2.

After shrinking the universe, we complement the sets to reduce between two Basic Problems $BP[\ell_1]$ and $BP[\ell_2]$ according to the following lemma.

Lemma 4.3 (Reduction between different Basic Problems). *For two different labels $\ell_1, \ell_2 \in \{0,1\}^k$, given set families $\mathcal{S}_1, \ldots, \mathcal{S}_k$, let $\mathcal{S}'_1, \ldots, \mathcal{S}'_k$ be defined such that*

$$\mathcal{S}'_i = \begin{cases} \left\{ S_i{}^{\complement} \mid S_i \in \mathcal{S}_i \right\}, & \text{if } \ell_1[i] \neq \ell_2[i] \\ \mathcal{S}_i, & \text{otherwise} \end{cases}, \text{for } i \in \{1, \ldots, k\}$$

then, $(\exists S_1 \in \mathcal{S}_1) \ldots (\exists S_k \in \mathcal{S}_k)(\forall u)\psi_{\ell_1}$ *iff* $(\exists S'_1 \in \mathcal{S}'_1) \ldots (\exists S'_k \in \mathcal{S}'_k)(\forall u)\psi_{\ell_2}$.

The proof of correctness is straightforward. For any $\epsilon > 0$, after the universe-shrinking self-reduction by Lemma 4.2, the new universe size n'_u has become $O(m^{5\epsilon})$. So the time complexity in this step is bounded by $O(m^{1+5\epsilon})$, which is significantly less than m^k even if $k = 2$.

Let new instance size be m'. We need to show that when we apply an algorithm better than $(m')^k$ algorithm on the constructed instance, we get an algorithm better than than m^k, i.e., for any δ there is a γ, so that $(m')^{k-\delta} < m^{k-\gamma}$. Since $m' = m^{1+5\epsilon}$ for an arbitrarily small constant ϵ, this will be true if we pick $\epsilon = \delta/10k$.

Finally, by the two-step fine-grained mapping reductions given by Lemma 4.2 and Lemma 4.3, we have a fine-grained mapping reduction between any two Basic Problems, completing the proof for Lemma 4.1.

When $k = 2$, Orthogonal Vectors ($BP[00]$), Set Containment ($BP[01]$ and $BP[10]$) and 2-Set Cover ($BP[11]$) are reducible to each other in subquadratic time. Thus Claim 2.1 follows.

4.1.1 Randomized universe-shrinking self-reduction of $BP[\ell]$ where $\ell \neq 1^k$

The main idea is to divide the sets into large and small ones. For large sets, there are not too many of them in the sparse structure, so we can work on them directly. For small sets, we use a Bloom Filter mapping each element in U to some elements in U' at random, and then for each set on universe U, we compute the corresponding set on universe U'. Next we can decide the same problem on these newly computed sets, instead of sets on U. ([CIP02] used a similar technique in reducing from Orthogonal Range Search to the Subset Query problem.) Because the sets are small, it is unlikely that some elements in two different sets on U are mapped to the same element on U', so the error probability of the reduction algorithm is small.

Step 1: Large sets. Let $d = m^\epsilon$. For sets of size at least d, we directly check if they are in any solutions. There are at most $O(m/d) = O(m^{1-\epsilon})$ of such large sets. In the outer loop, we enumerate all large sets in $\mathcal{S}_1, \ldots, \mathcal{S}_k$. If their sizes are pre-computed, we can do the enumeration in $O(m^{1-\epsilon})$. Assume the current large set is $S_i \in \mathcal{S}_i$. Because variables quantified by $\exists$ are interchangeable, we can interchange the order of variables, and let S_i be the outermost quantified variable S_1. On each such S_i (or S_1 after interchanging), we create new formula ψ_{S_1} on variables $S_2, \ldots, S_k, u$ from formula ψ, by replacing each occurrence of unary predicate on S_1 with a constant, and replacing each occurrence of binary predicate $R(S_1, S_j)$ (or $R(S_j, S_1)$) with unary predicate $R'(S_j)$ whose value equals $R(S_1, S_j)$ (or $R(S_j, S_1)$). Then, we decide if the graph induced by $\mathcal{S}_2, \ldots, \mathcal{S}_k$ and U satisfies $(\exists S_2) \ldots (\exists S_k)(\forall u)\psi_{S_1}$, using the baseline algorithm, which takes time $O(m^{k-1})$ for each such large set S_1. Thus the overall running time is $O(m^{1-\epsilon}) \cdot O(m^{k-1}) = O(m^{k-\epsilon})$. If no solution is found in this step, proceed to Step 2.

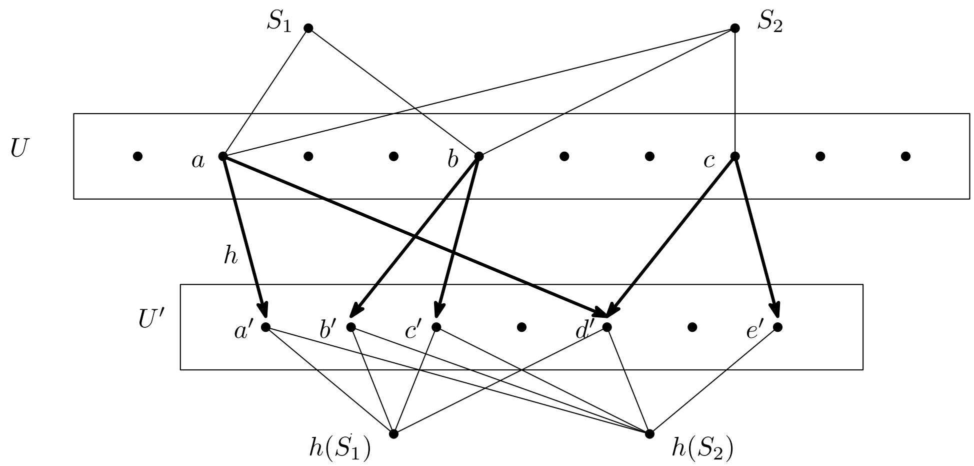


Figure 3: The universe-shrinking process. $S_1 = \{a, b\}$ and $S_2 = \{a, b, c\}$. After the mapping h, the new sets are $h(S_1) = \{a', b', c', d'\}$ and $h(S_2) = \{a', b', c', d', e'\}$.

Step 2: Small sets. Now we can exclude all the sets of size at least d. For sets of size smaller than d, we do the self-reduction to universe U' of size $n'_u = m^{5\epsilon}$. Let $t = m^\epsilon$, and let $h : U \to U'^t$ be a function that independently maps each element $u \in U$ to t elements in U' at random. On set $S \subseteq U$, we overload the notation h by defining $h(S) = \bigcup_{u \in S} h(u)$. For all set families $\mathcal{S}_i$, we compute new sets $h(S_i)$ for all $S_i \in \mathcal{S}_i$. Then, we decide whether the new sets satisfy the following sentence, which is another $BP[\ell]$ problem:

$$(\exists S_1) \ldots (\exists S_k)(\forall u) \left[\left(\bigvee_{i \in \{1,\ldots,k\}, \ell[i]=0} (\neg(u \in h(S_i))) \right) \vee \left(\bigvee_{i \in \{1,\ldots,k\}, \ell[i]=1} (u \in h(S_i)) \right) \right]$$

The size of the new instance is $O(nt) = O(m^{1+\epsilon})$, and the running time of the self-reduction is also $O(nt) = O(m^{1+\epsilon})$. So it is a fine-grained mapping reduction for any $k \geq 2$.

Figure 3 illustrates an example of the universe-shrinking self reduction for problem $BP[01]$, where we look for S_1, S_2 so that $S_1 \subseteq S_2$. If they exist, then after the self-reduction, it is always true that $h(S_1) \subseteq h(S_2)$. Still, it might happen that some $S_1 \not\subseteq S_2$ but $h(S_1) \subseteq h(S_2)$. In this case, a false positive occurs. In problem $BP[00]$ where we decide whether there exist S_i and S_j so that they are disjoint, a false negative may occur when there are two disjoint sets but some elements in $S_1 \cap S_2$ are mapped to the same element in U'. Next we will analyze the error probability of this reduction.

Analysis. Because variables quantified by $\exists$ are interchangeable, w.l.o.g. for ℓ containing i ($i \geq 1$) zeros and $k - i$ ones, we can assume $BP[\ell]$ is defined by

$$(\exists S_1) \ldots (\exists S_k)(\forall u) \left[\left(\bigvee_{j=1}^{i} (u \notin S_j) \right) \vee \left(\bigvee_{j=i+1}^{k} (u \in S_j) \right) \right],$$

or equivalently,

$$(\exists S_1) \ldots (\exists S_k) \left[\left(\bigcap_{j=1}^{i} S_j \right) \subseteq \left(\bigcup_{j=i+1}^{k} S_j \right) \right].$$

Let sets $A = \bigcap_{j=1}^{i} S_j$ and $B = \bigcup_{j=i+1}^{k} S_j$. Then the problem is to decide whether there exists $(S_1, \ldots, S_k)$ so that $A \subseteq B$. After the self-reduction, we let sets $A' = \bigcap_{j=1}^{i} h(S_j)$ and $B' = \bigcup_{j=i+1}^{k} h(S_j)$, and decide if there exists $(S_1, \ldots, S_k)$ such that $A' \subseteq B'$.

1. False positive. A false positive occurs when

$$\forall(S_1,\ldots,S_k), A \not\subseteq B, \text{ but } \exists(S_1,\ldots,S_k), A' \subseteq B'.$$

For a fixed tuple $(S_1,\ldots,S_k)$ such that $A \not\subseteq B$, an error occurs when $\exists u \in A-B$ such that $h(u) \subseteq B'$. The size of B' is at most kdt. So the error probability $\Pr[h(u) \subseteq B'] \leq (kdt/n'_u)^t = (km^\epsilon m^\epsilon/m^{5\epsilon})^t < m^{-\epsilon t}$. The size of $A-B$ is bounded by kd, so the probability $\Pr[\exists u \in A-B, h(u) \subseteq B'] \leq kd \cdot m^{-\epsilon t}$. There are $O(m^k)$ tuples of $(S_1,\ldots,S_k)$, so the total error probability is at most $O(m^k) \cdot kd \cdot m^{-\epsilon t} = O(m^{k+\epsilon-\epsilon m^\epsilon})$, which is exponentially small.

2. False negative. A false negative occurs when

$$\exists(S_1,\ldots,S_k), A \subseteq B, \text{ but } \forall(S_1,\ldots,S_k), A' \not\subseteq B'.$$

Fix any tuple $(S_1,...,S_k)$ that satisfies $A \subseteq B$ in the original instance, and consider the distribution on the corresponding $h(S_1),..,h(S_k)$. By definition, $B' = \bigcup_{u\in B} h(u)$, and so contains $\bigcup_{u\in A} h(u)$. So if $A' \subseteq \bigcup_{u\in A} h(u)$, we will have $A' \subseteq B'$, and there will not be a false negative. If not, then there is some $u' \in A' = \bigcap_{j=1}^{i} h(S_j)$, such that $u' \notin \bigcup_{u\in A} h(u)$. Then for each $j \in \{1,\ldots,i\}$, in each S_j there is a $u_j \in S_j$ with $u' \in h(u_j)$, but not all u_j are identical. (Otherwise the $u_j \in A$, so $u' \in h(u_j) \subseteq \bigcup_{u\in A} h(u)$, contradicting $u' \notin \bigcup_{u\in A} h(u)$). In particular, this means that for some j_1, j_2, there are $u_{j_1} \in S_{j_1}, u_{j_2} \in S_{j_2}$, such that $h(u_{j_1}) \cap h(u_{j_2}) \neq \emptyset$. So the error probability is bounded by $k^2 \cdot \Pr[\exists(u_1 \in S_{j_1}, u_2 \in S_{j_2}), h(u_1) \cap h(u_2) \neq \emptyset]$. Because $|S_{j_1}|$ and $|S_{j_2}|$ are at most d, by Birthday Paradox, the probability is at most $O(k^2d^2t^2/n'_u) = O(m^{-\epsilon})$. This is the upper bound of the error probability for the fixed $(S_1,\ldots,S_k)$ tuple. Then, the probability of the event "$\forall(S_1,\ldots,S_k), A' \not\subseteq B'$" is even smaller.

4.1.2 Deterministic universe-shrinking self-reduction of $BP[1^k]$

$BP[1^k]$ is the k-Set Cover problem, which decides whether there exist k sets covering the universe U. It is special in the Basic Problems: when n_u is small, the sets are easy to complement; when n_u is large, the problem is easy to solve.

Case 1: Large universe. If $n_u > m^\epsilon$, then in a solution of this problem, at least one set has size at least n_u/k. There are at most $m/(k/n_u) = O(m^{1-\epsilon})$ such large sets, thus they can be listed in time $O(m^{1-\epsilon})$, after pre-computation on the sizes of all sets. Our algorithm exhaustively searches all such large sets. And then, similarly to "Step 1" in Section 4.1.1, for each of the large sets, we run the baseline algorithm to find the remaining $k-1$ sets in the k-set cover, which takes time $O(m^{k-1})$. So the overall running time is $O(m^{1-\epsilon}) \cdot O(m^{k-1}) = O(m^{k-\epsilon})$.

Case 2: Small universe. If $n_u \leq m^\epsilon$, then we do not need a universe-shrinking self-reduction, because the universe is already small enough.

4.2 Hybrid Problem

Next we reduce general $MC(\exists^k\forall)$ problems to an intermediate problem called the Hybrid Problem, which is a combination of 2^k Basic Problems. Then by reducing from the Hybrid Problem to Basic Problems, we can set up a connection between $MC(\exists^k\forall)$ and OV.

Let $k \geq 2$. The input to the Hybrid Problem includes four parts:

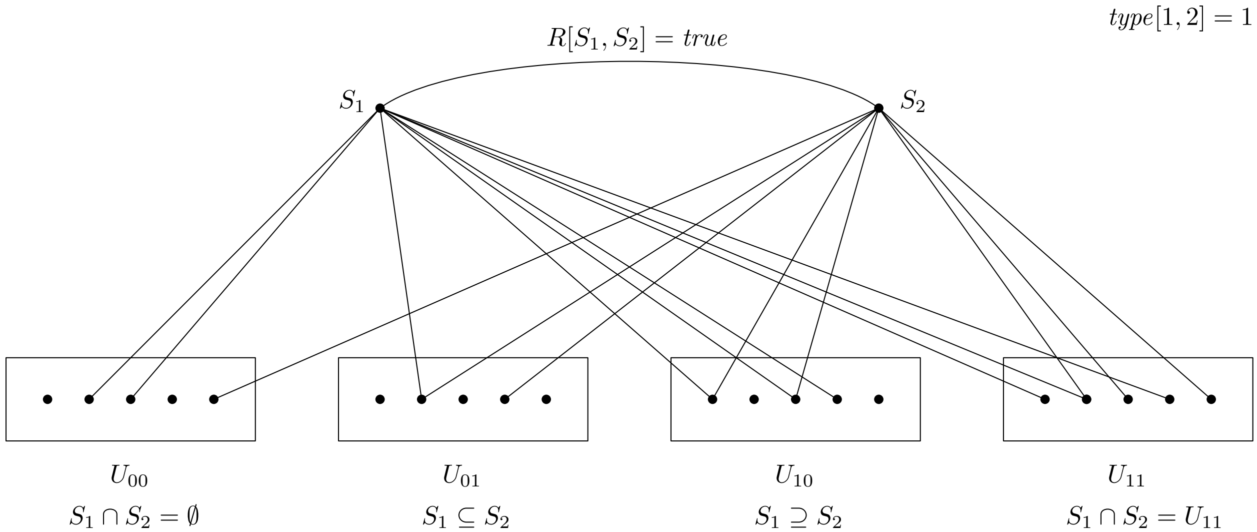


Figure 4: An example of a solution to a Hybrid Problem instance, when $k = 2$. In sub-universes $U_{00}, U_{01}, U_{10}, U_{11}$, sets S_1 and S_2 are solutions of $BP[00]$(2-Empty Intersection), $BP[01]$(Set Containment), $BP[10]$(Set Containment in the reversed direction) and $BP[11]$(2-Set Cover), respectively. And $\mathit{type}[1,2] = 1$ specifies that the predicate R on (S_1, S_2) must be true.

1. Set families $\mathcal{S}_1 \ldots \mathcal{S}_k$ defined on universe U, where U is partitioned into 2^k disjoint sub-universes: $U = \bigcup_{\ell \in \{0,1\}^k} U_\ell$.
2. A binary predicate R defined on pairs of sets from any two distinct set families. R is a symmetric relation ($R(S_i, S_j)$ iff $R(S_j, S_i)$).
3. *type* is binary string of length $\binom{k}{2}$, indexed by two integers $[i, j]$, s.t. $i, j \in \{1, \ldots, k\}$ and $i < j$.

The goal of the problem is to decide if there exist $S_1 \in \mathcal{S}_1, \ldots, S_k \in \mathcal{S}_k$ such that both of the following constraints are true:
(A) For each $\ell \in \{0,1\}^k$, $(S_1, \ldots S_k)$ is a solution of $BP[\ell]$ defined on sub-universe U_ℓ.
(B) For all pairs of indices $i, j \in \{1, \ldots, k\}, i < j$, we have that $R(S_i, S_j) = \mathit{true}$ iff $\mathit{type}[i, j] = 1$.

We let n be the sum of $|\mathcal{S}_1|, \ldots, |\mathcal{S}_k|$ and U, and let m be the number of all unary and binary relations. The Hybrid Problem is a first-order graph property problem with additional constraints. As usual, we assume all relations in the Hybrid Problem are sparse ($m \leq n^{1+o(1)}$). Figure 4 shows a solution to a Hybrid Problem instance when $k = 2$.

Intuition behind the Hybrid Problem. We mentioned in Section 2.2 that any first-order query containing two variables can be written in a “normal form”, which is a combination of sparse and co-sparse relations. The Hybrid Problem is designed for separating sparse relations from co-sparse ones, for *all* pairs of variables in formula φ.

The relation between the pair of variables (x_i, x_{k+1}) where $1 \leq i \leq k$ can be either sparse or co-sparse. Because there are k of such variables x_i, there are 2^k cases for a combination $((x_1, x_{k+1}), \ldots, (x_k, x_{k+1}))$. These cases correspond to the 2^k Basic Problems. In each Basic Problem, we deal with one of the 2^k cases.

For a relations between the pair of variables (x_i, x_j) where $1 \leq i < j \leq k$, it also can be either sparse or co-sparse. We use $\mathit{type}[i, j]$ to distinguish the two cases: when it is set to 1, we expect a sparse relation for (x_i, x_j), otherwise we expect a co-sparse relation.

4.2.1 Reduction to Basic Problems

Lemma 4.4. *(Hybrid Problem, m^k) $\leq_{rFGT}$ (OV, m^k).*

Given an instance of the Hybrid Problem, we can do the following modification in time $O(m)$. For each pair of indices i, j where $1 \leq i < j \leq k$, we construct auxiliary elements depending on the value of *type*$[i, j]$.

Case 1: If *type*$[i, j] = 0$, then in a solution to the Hybrid Problem, S_i and S_j should not have an edge $R(S_i, S_j)$ between them. Let ℓ be the length-k binary string where the i-th and j-th bits are zeros and all other bits are ones. For each edge $R(S_i, S_j)$, we add an extra element u_{ij} in U_ℓ and let $u_{ij} \in S_i$, $u_{ij} \in S_j$. Thus, S_i and S_j can both appear in the solution only when $(u_{ij} \notin S_i) \vee (u_{ij} \notin S_j)$, and it holds iff $R(S_i, S_j) = \mathit{false}$.

Case 2: If *type*$[i, j] = 1$, then in a solution to the Hybrid Problem, S_i and S_j should have an edge $R(S_i, S_j)$ between them. Let ℓ be the length-k binary string where the j-th bit is zero and all other bits are ones. For each $S_j \in \mathcal{S}_j$, we add an extra element u_j in U_ℓ and let $u_j \in S_j$. For each edge $R(S_i, S_j)$, we let $u_j \in S_i$. Thus, S_i and S_j can both appear in the solution only when $(u_j \notin S_j) \vee (u_j \in S_i)$, and it holds iff $R(S_i, S_j) = \mathit{true}$.

After the above construction, we can drop the constraint (B) of the Hybrid Problem. We will ignore the relation R and *type* in the Hybrid Problem. The problem now is to decide whether there exists tuple $(S_1, \ldots, S_k)$ being a solution to all 2^k Basic Problems. Then we can use Lemma 4.1 to reduce all these Basic Problems to $BP[0^k]$. Let $U_\ell{}'$ be the sub-universe of the $BP[0^k]$ instance reduced from the $BP[\ell]$ sub-problem. $(S_1, \ldots, S_k)$ is a solution to all Basic Problems iff their intersection is empty on every sub-universe U'_ℓ, iff their intersection is empty on universe $\bigcup_{\ell \in \{0,1\}^k} U'_\ell$, i.e., it is a solution of a $BP[0^k]$ instance.

Multiplying the error probability in the reductions between Basic Problems by 2^k, which is a constant number, and then taking a union bound, we get similar bounds of error probability for the Hybrid Problem.

4.2.2 Assumptions on the input graph

In the remainder of this paper, we will work on generalized input graph G. We adopt the following conventions.

We use letter φ to represent formulas with quantifiers, and letter ψ for quantifier-free formulas. Unlike in database theory where "relations" refers to tables and "tuples" refers to rows in tables, we say "relations" to mean the rows, i.e., edges in graphs that correspond to binary predicates in φ. We use the word "tuple" (or "pair" for binary tuples) for any possible combinations of variables or vertices. To avoid ambiguity, we let $x_1, \ldots, x_{k+1}$ stand for variables in φ, and let $v_1, \ldots, v_{k+1}$ be the concrete values assigned to the variables (i.e., vertices). Let $x_i \leftarrow v_i$ denote that variable x_i is assigned the value v_i.

Without loss of generality we make the following assumptions about the input graph:

- G is a $(k+1)$-partite directed graph, whose vertex set is partitioned into $V_1, \ldots, V_{k+1}$, of sizes $n_1, \ldots, n_{k+1}$ respectively. For each $i \in \{1, \ldots, k+1\}$, V_i is the set of candidate values for x_i. In other words, we want to decide whether $(\exists x_1 \in V_1) \ldots (\exists x_k \in V_{k+1}) \psi(x_1, \ldots, x_{k+1})$. This assumption can be achieved by creating nodes and adding unary predicates, which can be done in time linear to m.

- There is a data structure where we can both check the existence of an edge (whether a relation holds) in constant time, and enumerate the incident edges of a vertex in time proportional to its degree (e.g., a hash table of edges together with a linked list of edges for each vertex).
- If any predicate occurs multiple times with different argument lists, we rename it to different predicates, and split the corresponding set of edges. For example, we can replace a subformula $[(\neg R(x_1, x_2) \wedge R(x_2, x_3)) \vee R(x_1, x_2)]$ by $[(\neg R_1(x_1, x_2) \wedge R_2(x_2, x_3)) \vee R_1(x_1, x_2)]$, and then move the E_R edges on (V_1, V_2) to edge set E_{R_1}, and the edges on (V_2, V_3) to E_{R_2}. This modification can be done in linear time.

4.2.3 Turing reduction from general $MC(\exists^k\forall)$ problems to the Hybrid Problem

Lemma 4.5. *For any integer $k \geq 2$, any problem in $MC(\exists^k\forall)$ is linear-time Turing reducible to the Hybrid Problem.*

Consider the problem MC_φ where $\varphi = (\exists x_1)\ldots(\exists x_k)(\forall x_{k+1})\psi(x_1, \ldots, x_{k+1})$. Let P_{k+1} be the set of unary and binary predicates in ψ that involve variable x_{k+1}, and let $P_{\overline{k+1}}$ denote the set of the other predicates not including x_{k+1}. We give all predicates in $P_{\overline{k+1}}$ a canonical order. A *partial interpretation* α for $P_{\overline{k+1}}$ is a binary string of length $|P_{\overline{k+1}}|$, that encodes the truth values assigned to all predicates in $P_{\overline{k+1}}$. For each i s.t. $1 \leq i \leq |P_{\overline{k+1}}|$, if the i-th predicate in $P_{\overline{k+1}}$ is assigned to *true*, then we set the i-th bit of α to one, otherwise we set it to zero. For a tuple $(v_1, \ldots, v_k)$, we say it *implies* α (denoted by $(v_1, \ldots, v_k) \models \alpha$) iff when $(x_1 \leftarrow v_1, \ldots, x_k \leftarrow v_k)$. the evaluations of all predicates in $P_{\overline{k+1}}$ are the same as the values specified by α.

For each $\alpha \in \{0,1\}^{P_{\overline{k+1}}}$, we create a distinct Hybrid Problem instance H_α. If any of the Hybrid Problems accepts, we accept. Let $\psi|_\alpha(x_1, \ldots, x_{k+1})$ be ψ after replacing all occurrences of predicates in $P_{\overline{k+1}}$ by their corresponding truth values specified by α. The following steps show how to create H_α from α and $\psi|_\alpha(x_1, \ldots, x_{k+1})$.

Step 1: Construction of sets. We introduce *colors*, which are partial interpretations defined on some specific subsets of the predicates concerning variable x_{k+1}. We call them "colors" because they can be considered as a kind of labels on (v_i, v_{k+1}) pairs. For each $i \in \{1, \ldots, k\}$, we give all the unary and binary predicates defined on (x_i, x_{k+1}) (including those on (x_{k+1}, x_i)) a canonical order. We use P_i to denote the set of these predicates for each i. Let a color be a partial interpretation for P_i, which is a binary string of length $|P_i|$, encoding the truth values assigned to all predicates in P_i. For each j s.t. $1 \leq j \leq |P_i|$, if the j-th predicate in P_i is assigned to *true*, then we set the j-th bit of the color to one, otherwise we set it to zero. For a color $c_i \in \{0,1\}^{|P_i|}$, we say $(v_i, v_{k+1}) \models c_i$ iff when $x_i \leftarrow v_i$ and $x_{k+1} \leftarrow v_{k+1}$, the values of all predicates in P_i are the same as the corresponding bits of c_i. We refer to the colors where all bits are zeros as the *background colors*. These colors are special because they correspond to interpretations where all predicates in P_i are false, i.e., we cannot directly go through all pairs (v_i, v_{k+1}) where $(v_i, v_{k+1}) \models 0^{|P_i|}$, since this is a co-sparse relation. So we need to deal with these pairs separately.

For a vertex combination $(v_1, \ldots, v_{k+1})$ where $(v_i, v_{k+1}) \models c_i$ on all $1 \leq i \leq k$, the k-color-tuple $(c_1, \ldots, c_k)$ form a *color combination*, which corresponds to truth values assigned to all the predicates in P_{k+1}.

For each $v_i \in V_i$ where $1 \leq i \leq k$, we create set S_{v_i} in the set family $\mathcal{S}_i$. For each $v_{k+1} \in V_{k+1}$, and each color combination $(c_1, \ldots, c_k)$ s.t. $c_i \in \{0,1\}^{|P_i|}$ and the values of all predicates specified by $(c_1, \ldots, c_k)$ make $\psi|_\alpha$ evaluate to *false* (in which case we say $(c_1, \ldots, c_k)$ does

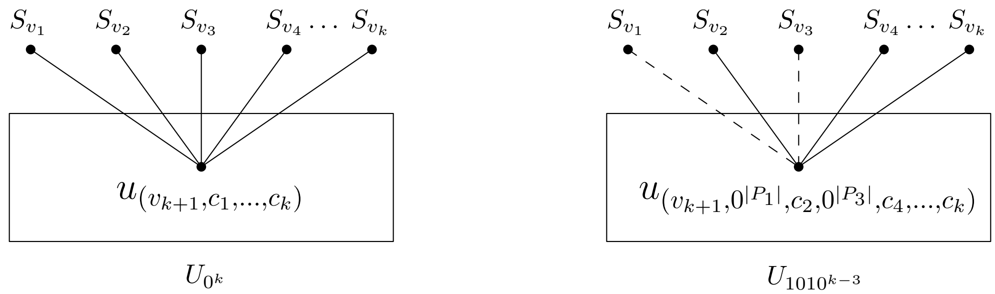


Figure 5: The formula is satisfied iff there exists $(S_{v_1}, S_{v_2}, \dots, S_{v_k})$ so that there does not exist such an element u in any of the sub-universes: the left figure illustrates the case where none of $c_1, \dots, c_k$ is a background color. The right is the case where only c_1 and c_3 are background colors. (The dashed lines stand for non-existing edges.)

not satisfy $\psi|_\alpha$), we create an element $u_{(v_{k+1},c_1,\dots,c_k)}$ in U. We call a string $C \in \{0,1\}^k$ an *encoding* of a color combination $(c_1, \dots, c_k)$ when on all indices $i \in \{1, \dots, k\}$, $C[i] = 1$ iff $c_i = 0^{|P_i|}$. We put each element $u_{(v_{k+1},c_1,\dots,c_k)}$ in the sub-universe U_C iff C is an encoding of $(c_1, \dots, c_k)$.

Next we will construct the sets. For each $v_i \in V_i$, let S_{v_i} be

$$S_{v_i} = \{u_{(v_{k+1},c_1,\dots,c_k)} \mid (c_1, \dots, c_k) \text{ does not satisfy } \psi|_\alpha, \text{ and} \\ ((c_i \neq 0^{|P_i|}, (v_i, v_{k+1}) \models c_i), \text{ or } (c_i = 0^{|P_i|}, (v_i, v_{k+1}) \not\models c_i = 0^{|P_i|}))\}.$$

To construct such sets, for each edge on (x_i, x_{k+1}) (and (x_{k+1}, x_i)), we do the following. Assume the current vertex pair is (v_i, v_{k+1}).

1. First, let set S_{v_i} contain all elements $u_{(v_{k+1},c_1,\dots,c_k)}$ in U where c_i is a fixed color such that $(v_i, v_{k+1}) \models c_i$, and the other colors c_j can be any string in $\{0,1\}^{|P_j|}$.
2. Next, let set S_{v_i} contain all elements $u_{(v_{k+1},c_1,\dots,c_k)}$ in U where $c_i = 0^{|P_i|}$ (here $(v_i, v_{k+1}) \not\models c_i = 0^{|P_i|}$ because there is some edge connecting v_i and v_{k+1}, meaning at least one bit in c_i is 1), and the other colors c_j can be any string in $\{0,1\}^{|P_j|}$.

In other words, in the sub-universe labeled by 0^k, which is made up of elements $u_{(v_{k+1},c_1,\dots,c_k)}$ such that none of the c_i equals $0^{|P_i|}$, and that $(c_1, \dots, c_k)$ does not satisfy $\psi|_\alpha$, a set S_{v_i} contains an element $u_{(v_{k+1},c_1,\dots,c_k)}$ iff $(v_i, v_{k+1}) \models c_i$. On the other hand, in the sub-universe labeled by C where the i-th bit of C is 1, which is made up of elements $u_{(v_{k+1},c_1,\dots,c_k)}$ such that $c_i = 0^{|P_i|}$ and that $(c_1, \dots, c_k)$ does not satisfy $\psi|_\alpha$, a set S_{v_i} contains an element $u_{(v_{k+1},c_1,\dots,c_k)}$ iff $(v_i, v_{k+1}) \not\models c_i = 0^{|P_i|}$.

Analysis. Now we show the above construction achieves constraint (A) in the definition of the Hybrid Problem.

- Assume that $(v_1, \dots, v_k)$ does not satisfy $(\forall v_{k+1})\psi|_\alpha(v_1, \dots, v_{k+1})$, i.e., there exists some $v_{k+1} \in V_{k+1}$ such that $\psi|_\alpha(v_1, \dots, v_{k+1})$ is false. Then consider the specific color combination $(c_1, \dots, c_k)$ where on each i, $(v_i, v_{k+1}) \models c_i$. So $(c_1, \dots, c_k)$ does not satisfy $\psi|_\alpha(x_1, \dots, x_{k+1})$. Thus there exists an element $u_{(v_{k+1},c_1,\dots,c_k)}$ in U.

  If none of the colors in combination $(c_1, \dots, c_k)$ is the background color, then the encoding of $(c_1, \dots, c_k)$ is the string 0^k. Thus, the element $u_{(v_{k+1},c_1,\dots,c_k)}$ is in sub-universe U_{0^k}. By our construction, $u_{(v_{k+1},c_1,\dots,c_k)}$ is contained in all of $S_{v_1}, \dots, S_{v_k}$, as shown on the left side of Figure 5. This is because for when we went through all the edges, at the edge between (v_i, v_{k+1}), we put $u_{(v_{k+1},c_1,\ldots,c_k)}$ in S_{v_i}, since none of the colors is background. Thus $(\exists u \in U_{0^k})\left[\bigwedge_{i=1}^{k}(u \in S_{v_i})\right]$, so it is not the case that $(\forall u \in U_{0^k})\left[\bigvee_{i=1}^{k}\neg(u \in S_{v_i})\right]$, which means $S_{v_1},\ldots,S_{v_k}$ is not a solution of $BP[0^k]$ on sub-universe U_{0^k}.

If some of the colors c_i in the color combination $(c_1,\ldots,c_k)$ equal the background color $0^{|P_i|}$, then in the encoding C of $(c_1,\ldots,c_k)$, $C[i]=1$. Thus, the element $u_{(v_{k+1},c_1,\ldots,c_k)}$ is in the sub-universe U_C. By our construction, $u_{(v_{k+1},c_1,\ldots,c_k)}$ is contained in sets S_{v_i} for all indices i where c_i is not the background color $0^{|P_i|}$, and is not contained in sets S_{v_j} for all indices j where c_j is the background color $0^{|P_j|}$. The latter case is because for each index j where c_j is the background color, there is no edge connecting the pair of vertices (v_j, v_{k+1}). So we did not put $u_{(v_{k+1},c_1,\ldots,c_k)}$ in S_{v_j}. (The right side of Figure 5 demonstrates the example where c_1 and c_3 are the background colors while other colors are not.) Thus

$$(\exists u \in U_C)\left[\bigwedge_{i\in\{1,\ldots,k\},C[i]=0}(u \in S_{v_i}) \wedge \bigwedge_{i\in\{1,\ldots,k\},C[i]=1}(\neg(u \in S_{v_i}))\right],$$

so it is not the case that

$$(\forall u \in U_C)\left[\bigvee_{i\in\{1,\ldots,k\},C[i]=0}(\neg(u \in S_{v_i})) \vee \bigvee_{i\in\{1,\ldots,k\},C[i]=1}(u \in S_{v_i})\right],$$

which means $S_{v_1},\ldots,S_{v_k}$ is not a solution of $BP[C]$ on sub-universe U_C.

- On the other hand, assume that $(v_1,\ldots,v_k)$ satisfies $(\forall v_{k+1})\psi|_\alpha(v_1,\ldots,v_{k+1})$. We claim that for *all* $\ell \in \{0,1\}^k$, $(S_{v_1},\ldots,S_{v_k})$ is a solution to Basic Problem $BP[\ell]$.

  Consider the sub-universe U_C for each $C \in \{0,1\}^k$. If $C = 0^k$, i.e., the sub-universe is U_{0^k} corresponding to $BP[0^k]$, then none of the elements $u_{(v_{k+1},c_1,\ldots,c_k)}$ in U_{0^k} contains any background color among its $c_1,\ldots,c_k$. For the sake of contradiction, suppose there exists an element $u_{(v_{k+1},c_1,\ldots,c_k)}$ that is contained in all sets $S_{v_1},\ldots,S_{v_k}$. So by our construction of sets, for each $i \in \{1,\ldots,k\}$, $(v_i,v_{k+1}) \models c_i$. Recall that the color combination $(c_1,\ldots,c_k)$ in any element $u_{(v_{k+1},c_1,\ldots,c_k)}$ does not satisfy $\psi|_\alpha$. Then this means the vertex v_{k+1} does not satisfy $\psi|_\alpha(v_1,\ldots,v_k,v_{k+1})$, which leads to a contradiction.

  Thus on $(S_{v_1},\ldots,S_{v_k})$, it is not the case that $(\exists u \in U_{0^k})\left[\bigwedge_{i=1}^{k}(u \in S_{v_i})\right]$, implying $(S_{v_1},\ldots,S_{v_k})$ satisfies $(\forall u \in U_{0^k})\left[\bigvee_{i=1}^{k}\neg(u \in S_{v_i})\right]$. So it is a solution of the Basic Problem $BP[0^k]$ on sub-universe U_{0^k}.

  If $C \neq 0^k$, for the sake of contradiction, suppose there exists an element $u_{(v_{k+1},c_1,\ldots,c_k)}$ such that among $S_{v_1},\ldots,S_{v_k}$, it is contained in set S_{v_i} iff $C[i]=0$. Then by our construction of sets, this means for all i such that $C[i]=0$, $(v_i,v_{k+1}) \models c_i$; while for all i such that $C[i] \neq 0$, $(v_i,v_{k+1}) \models 0^{|P_i|} = c_i$. Combining the two statements, for all i, $(v_i,v_{k+1}) \models c_i$. Recall again that the color combination $(c_1,\ldots,c_k)$ in any element $u_{(v_{k+1},c_1,\ldots,c_k)}$ does not satisfy $\psi|_\alpha$. This implies the vertex v_{k+1} does not satisfy $\psi|_\alpha(v_1,\ldots,v_{k+1})$, which leads to a contradiction.

  Thus on $(S_{v_1},\ldots,S_{v_k})$, it is not the case that

  $$(\exists u \in U_C)\left[\bigwedge_{i\in\{1,\ldots,k\},C[i]=0}(u \in S_{v_i}) \wedge \bigwedge_{i\in\{1,\ldots,k\},C[i]=1}(\neg(u \in S_{v_i}))\right],$$

  implying $(S_{v_1},\ldots,S_{v_k})$ satisfies

  $$(\forall u \in U_C)\left[\bigvee_{i\in\{1,\ldots,k\},C[i]=0}(\neg(u \in S_{v_i})) \vee \bigvee_{i\in\{1,\ldots,k\},C[i]=1}(u \in S_{v_i})\right].$$

So it is a solution of the Basic Problem $BP[C]$ on sub-universe U_C.

In summary, there exists tuple $(v_1, \ldots, v_k)$ such that $(\forall v_{k+1})\psi|_\alpha(v_1, \ldots, v_k, v_{k+1})$ holds true, iff there exist sets $(S_{v_1}, \ldots, S_{v_k})$ such that for all $\ell \in \{0,1\}^k$, $(S_{v_1}, \ldots, S_{v_k})$ is a solution of Basic Problem $BP[\ell]$ on sub-universe U_ℓ. Thus our reduction satisfies constraint (A) of the Hybrid Problem.

Step 2: Construction of relation R and string *type*. Next, we consider the predicates in $P_{\overline{k+1}}$, which are predicates unrelated to variable x_{k+1}. We create edges for predicate R according to the current partial interpretation α.

For a pair of vertices $v_i \in V_i$ and $v_j \in V_j$ where $1 \leq i < j \leq k$, we say (v_i, v_j) agrees with α if the evaluations of all predicates on (x_i, x_j) (including (x_j, x_i)) when $x_i \leftarrow v_i, x_j \leftarrow v_j$, is the same as the truth values of corresponding predicates specified by α.

Case 1: At least one predicate on (x_i, x_j) in α is true. (i.e., (x_i, x_j) is in a sparse relation) For all edges (v_i, v_j) (including (v_j, v_i)) where $v_i \in V_i$ and $v_j \in V_j$ and $i < j \leq k$, if (v_i, v_j) agrees with α, then we create edge $R(S_{v_i}, S_{v_j})$. Finally we make $type[i,j] = 1$ in the Hybrid Problem H_α.

Case 2: All predicates on (x_i, x_j) in α are false. (i.e., (x_i, x_j) is in a co-sparse relation) For all edges (v_i, v_j) (including (v_j, v_i)) where $v_i \in V_i$ and $v_j \in V_j$ and $i < j \leq k$, if (v_i, v_j) does not agree with α, then we create edge $R(S_{v_i}, S_{v_j})$. Finally we make $type[i,j] = 0$ in the Hybrid Problem H_α.

Analysis. We prove that (v_i, v_j) can appear in the solution of H_α only if when it agrees with α. If (v_i, v_j) does not agree with α, we should not let them be in any solution of H_α. This is done by the relation R and the string *type*.

Consider the two cases. If in α some predicates on (x_i, x_j) are true (i.e., tuples that agree with α is sparse), then in any (v_i, v_j) that agrees with α, there must be an edge in G connecting v_i and v_j. So we can add an edge (defined by relation R) on the corresponding sets S_{v_i}, S_{v_j} and require there must be such an edge in the solution (i.e., *type* being 1).

On the other hand, if all predicates on (x_i, x_j) in α are false (i.e., tuples agreeing with α is co-sparse), then in any (v_i, v_j) that agrees with α, there should not be any edge connecting v_i and v_j. In this case we turn to consider the tuples (v_i, v_j) that do not agree with α (which is a sparse relation, instead of co-sparse). We create edges on the corresponding sets S_{v_i}, S_{v_j} and require there must *not* be such an edge in the solution (i.e., *type* being 0).

Therefore, a tuple $(v_1, \ldots, v_k)$ implies α iff for all $i, j \in \{1, \ldots, k\}, i < j$, the truth value of relation $R(S_{v_i}, S_{v_j})$ equals whether $type[i,j] = 1$. Thus our reduction satisfies constraint (B) of the Hybrid Problem.

From the analyses of the two steps, we have justified that: there exists $(v_1, \ldots, v_k)$ so that $(v_1, \ldots, v_k) \models \alpha$, and $\psi|_\alpha$ holds for all $v_{k+1} \in V_{k+1}$, iff there exists $(S_{v_1}, \ldots, S_{v_k})$ being a solution to the Hybrid Problem H_α. Thus, if for any $\alpha \in \{0,1\}^{P_{\overline{k+1}}}$, the Hybrid Problem H_α accepts, then there exists a solution $(v_1, \ldots, v_k)$ so that $\psi(v_1, \ldots, v_k, v_{k+1})$ holds for all $v_{k+1} \in V_{k+1}$. Otherwise there does not exist such a solution. The argument proves the following claim.

Claim 4.1. *The two propositions are equivalent:*
(1) MC_φ has a solution $x_1 \leftarrow v_1, \ldots, x_k \leftarrow v_k$ such that $(\forall v_{k+1} \in V_{k+1})\psi(v_1, \ldots, v_{k+1})$ is satisfied.

(2) *There exists an* $\alpha \in \{0,1\}^{P_{\overline{k+1}}}$ *so that* $(S_{v_1}, \ldots, S_{v_k}) \models \alpha$*, and* $S_{v_1}, \ldots, S_{v_k}$ *is a solution to the Hybrid Problem* H_α*.*

The running time of the whole reduction process is linear in the total number of edges in the graph, because the number of predicates is constant. Thus Lemma 4.5 follows.

5 Hardness of k-OV for $MC(\forall\exists^{k-1}\forall)$

In this section we extend the reduction from Hitting Set to Orthogonal Vectors in [AWW15] to sparse structures, giving a fine-grained Turing reduction from any $MC(\forall\exists^{k-1}\forall)$ problem to a $MC(\exists^k\forall)$ problem, establishing the hardness of k-OV for these problems.

Lemma 5.1. *For* $k \geq 2$*, let* $\varphi' = (\exists x_2) \ldots (\exists x_k)(\forall x_{k+1})\psi(x_1, \ldots, x_{k+1})$*. There is a fine-grained Turing reduction*

$$(MC_{(\forall x_1)\varphi'}, m^k) \leq_{FGT} (MC_{(\exists x_1)\varphi'}, m^k).$$

We continue to use the conventions and assumptions in Section 4.2.2. First, we show that in problem $MC_{(\exists x_1)\varphi'}$, if graph G satisfies $(\exists x_1)\varphi'$, then we can find a satisfying value v_1 for variable x_1 by binary search. We divide the set V_1 into two halves, take each half of V_1 and query whether $(\exists x_1)\varphi'$ holds true on the graph induced by this half of V_1 together with the original sets $V_2, \ldots, V_{k+1}$. If any half of V_1 works, then we can shrink the set of candidate values for x_1 by a half, and then recursively query again, until there is only one vertex v_1 left. So it takes $O(\log |V_1|)$ calls to find a v_1 in some solution. This means as long as there is a solution for $MC_{\exists x_1 \varphi'}$, we can find a satisfying v_1 efficiently, with $O(\log m)$ queries to the decision problem.

Step 1: Large degree vertices. Let $t = m^{(k-1)/k}$. We deal with vertices in $V_1 \ldots V_k$ with degree greater than t. There are at most $m/t = m^{1/k}$ such vertices. After pre-computing the sizes of all the sets, these large sets can be listed in time $O(m^{1/k})$.

Step 1-1: Large degree vertices in V_1. For each vertex $v_1 \in V_1$ with degree at least t, we create a formula ψ_{v_1} on variables $x_2, \ldots, x_{k+1}$ from formula ψ, by replacing occurrences of unary predicates in ψ on x_1 by constants, and replacing occurrences of binary predicates involving x_1 by unary predicates on the other variables. Then we check if the graph induced by $V_2, \ldots, V_{k+1}$ satisfies $(\exists x_2) \ldots (\exists x_k)(\forall x_{k+1})\psi_{v_1}(x_2, \ldots, x_{k+1})$ by running the baseline algorithm in time $O(m^{k-1})$. If the new formula is satisfied, then we mark v_1 as "good". The total time complexity is $O(m^{1/k}) \cdot O(m^{k-1}) = O(m^{k-1+1/k})$.

Step 1-2: Large degree vertices in $V_2, \ldots, V_k$. Now we exhaustively search over all vertices $v_1 \in V_1$ with degree less than t in the outermost loop. For each such v_1, we find out all vertices $v_i \in V_i$ for $2 \leq i \leq k$, with degree at least t. Again, there are at most $O(m^{1/k})$ of them.

Case 1: $k > 2$**.** Because variables x_2 through x_k are all quantified by $\exists$, we interchange their order so that the variable x_i becomes the second-outermost variable x_2 (and thus the current v_i becomes v_2). Next, for each v_1 and v_2 we construct a new formula $\psi_{(v_1,v_2)}$ on variables $x_3, \ldots, x_{k+1}$, by regarding x_1 and x_2 as fixed values v_1 and v_2, and then modify ψ into $\psi_{(v_1,v_2)}$ similarly to the previous step. Again, we run the baseline algorithm to check whether the graph induced by the current $V_3, \ldots, V_{k+1}$ satisfies $(\exists x_3) \ldots (\exists x_{k+1})\psi_{(v_1,v_2)}(x_3, \ldots, x_{k+1})$, using time $O(m^{k-2})$. If the formula is satisfied, we mark the current v_1 as "good". The total time complexity is $O(m \cdot m^{1/k}) \cdot (m^{k-2}) = O(m^{k-1+1/k})$.

Case 2: $k = 2$**.** For each vertex v_2, we mark all the v_1's satisfying $\forall x_3 \psi(x_1, x_2, x_3)$ as "good". This can be done in $O(m)$ using the algorithm for the base case of the baseline algorithm, by treating the current v_2 as constant. So this process runs in time $O(m^{1/k}) \cdot O(m) = O(m^{3/2})$.

If not all vertices in V_1 with degree at least t are marked "good", we reject. Otherwise proceed to Step 2.

Step 2: Small degree vertices. First we exclude all the large vertices from the graph. Then for the "good" vertices found in the previous step, we also exclude them from V_1.

Now all vertices have degree at most t. In each of $V_1, \ldots, V_k$, we pack their vertices into groups where in each group the total degree of vertices is at most t. Then the total number of groups is bounded by $O(m/t)$.

For each k-tuple of groups $(G_1, \ldots, G_k)$ where $G_1 \subseteq V_1, \ldots, G_k \subseteq V_k$, we query the oracle deciding $MC_{(\exists x_1)\varphi'}$ whether it accepts on the subgraph induced by vertices in $G_1, \ldots, G_k$. If so, then we find a vertex v_1 in V_1 so that when $x_1 \leftarrow v_1$, the current subgraph satisfies φ'. We remove this v_1 from V_1. Then we repeat this process to find new satisfying v_1's in V_1, and remove these v_1's from V_1. When V_1 is empty, or when no new solution is found after all group combinations are exhausted, the algorithm terminates. If in the end V_1 is empty, then all $v_1 \in V_1$ are in solutions of $MC_{\exists x_1 \varphi'}$, so we accept. Otherwise we reject.

Each query to $MC_{\exists x_1 \varphi'}$ has size $m' = O(kt) = O(t)$. Because the number of different k-tuples of groups is $O(m/t)^k = O((m/t)^k)$, the number of queries made is $O((m/t)^k + |V_1|) \cdot O(\log m) = O((m^{1/k})^k + |V_1|) \cdot O(\log m) = O(m \log m)$ times. If $MC_{\exists x_1 \varphi'}$ on input size m' is solvable in time $O(m'^{k-\epsilon})$, then the running time for $MC_{\forall x_1 \varphi'}$ is $O(m \log m) \cdot O(m'^{k-\epsilon}) = O(m^{1+((k-1)/k)(k-\epsilon)} \log m) = O(m^{k-(1-1/k)\epsilon} \log m)$. The exponent of m is less than k. Thus this is a fine-grained Turing reduction. Lemma 5.1 follows.

6 Open Problems

One obvious open problem is to derandomize our universe-shrinking self-reductions, or show that this is not possible. One delicate point is that we cannot increase the running times by even a small polynomial factor.

Our results raise the possibility that many other classes have complete problems under fine-grained reducibility, and that this will be a general method for establishing the plausibility of conjectures on the fine-grained complexity of problems. There are some obvious candidates for such classes. We could drop the restriction that all relations are binary or unary, and look at first-order "hypergraph" properties. While it is possible to reduce such problems to first-order graph properties, and even in a way that preserves the number of edges up to constant factors, doing so usually introduces more quantifiers and variables, and so is not in general a fine-grained reduction. We could also stratify the first-order formulas by *variable* complexity, the number of distinct variable names in a formula, rather than number of quantifiers. (Variable complexity arises naturally in database theory, because the variable complexity determines the arity of some relation in any way of expressing the query as a sequence of sub-queries.) First-order logic is rather limited, so we could look at augmentations that increase its reach, such as allowing a total ordering on elements, or allowing the logic to take transitive closures of relations (e.g., to talk about the reachability relation in a sparse directed graph), or more generally, introduce monotone fixed point operations.

We'd like to find more reductions between and equivalences among the problems that are proven hard under some conjecture. For example, Edit Distance, Fréchet Distance, and Longest Common Subsequence are all almost quadratically hard assuming SETH. Are there any reductions between these problems? Are they all equivalent as far as having subquadratic algorithms? All of these problems have similar dynamic programming formulations. Can we formalize a class of problems with such dynamic programming algorithms problems and find complete problems for this class? More generally, we would like taxonomies of the problems within P that would classify more of the problems that have conjectured hardness, or have provable hardness based on conjectures about other problems. Such a taxonomy might have to be based on the structure of the conjectured best algorithms for the problems rather than on resource limitations.

Acknowledgments

First of all, we thank Virginia Vassilevska Williams for her inspiring ideas. We would like to thank Marco Carmosino, Antonina Kolokolova, Ivan Mihajlin and Victor Vianu for proofreading and suggestions on this paper. We also thank Valentine Kabanets, Ramamohan Paturi, Ramyaa and Stefan Schneider for many useful discussions.

References

[ABW15] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Quadratic-time hardness of LCS and other sequence similarity measures. *CoRR*, abs/1501.07053, 2015.

[AHWW15] Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends or: A polylog shaved is a lower bound made. *arXiv preprint arXiv:1511.06022*, 2015.

[Ajt83] Miklós Ajtai. Σ_1^1-formulae on finite structures. *Annals of pure and applied logic*, 24(1):1–48, 1983.

[AWW15] Amir Abboud, Virginia Vassilevska Williams, and Joshua Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter. *arXiv preprint arXiv:1506.01799*, 2015.

[AWY15] Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 218–230. SIAM, 2015.

[BCH14] Michele Borassi, Pierluigi Crescenzi, and Michel Habib. Into the square-on the complexity of quadratic-time solvable problems. *arXiv preprint arXiv:1407.4972*, 2014.

[BI15] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 51–58, 2015.

[BK15] Karl Bringmann and Marvin Kunnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 79–97. IEEE, 2015.

[Bri14] Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless seth fails. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 661–670. IEEE, 2014.

[CGI$^+$16] Marco L Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 261–270. ACM, 2016.

[CIP02] Moses Charikar, Piotr Indyk, and Rina Panigrahy. New algorithms for subset query, partial match, orthogonal range searching, and related problems. In *Automata, Languages and Programming*, pages 451–462. Springer, 2002.

[DF92] Rodney G Downey and Michael R Fellows. Fixed-parameter intractability. In *Structure in Complexity Theory Conference, 1992., Proceedings of the Seventh Annual*, pages 36–49. IEEE, 1992.

[Fag76] Ronald Fagin. Probabilities on finite models. *The Journal of Symbolic Logic*, 41(01):50–58, 1976.

[FG06] Jörg Flum and Martin Grohe. Parameterized complexity theory, volume xiv of texts in theoretical computer science. an eatcs series, 2006.

[FSS84] Merrick Furst, James B Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[GO95] Anka Gajentaan and Mark H Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational geometry*, 5(3):165–185, 1995.

[HKNS15] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 21–30. ACM, 2015.

[IP99] Russell Impagliazzo and Ramamohan Paturi. Complexity of k-SAT. In *Computational Complexity, 1999. Proceedings. Fourteenth Annual IEEE Conference on*, pages 237–240. IEEE, 1999.

[IPZ98] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 653–662. IEEE, 1998.

[JMV15] Hamid Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. In *Automata, Languages, and Programming*, pages 749–760. Springer, 2015.

[JS99] David S Johnson and Mario Szegedy. What are the least tractable instances of max independent set? In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 927–928. Society for Industrial and Applied Mathematics, 1999.

[MPS15] Daniel Moeller, Ramamohan Paturi, and Stefan Schneider. Subquadratic algorithms for succinct stable matching. *arXiv preprint arXiv:1510.06452*, 2015.

[NRS95] Ashish V Naik, Kenneth W Regan, and D Sivakumar. On quasilinear-time complexity theory. *Theoretical Computer Science*, 148(2):325–349, 1995.

[SHI90] Richard Edwin Stearns and Harry B Hunt III. Power indices and easier hard problems. *Mathematical Systems Theory*, 23(1):209–225, 1990.

[Var82] Moshe Y Vardi. The complexity of relational query languages. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 137–146. ACM, 1982.

[Wil] Virginia Vassilevska Williams. STOC tutorial: Hardness and equivalences in P. `http://theory.stanford.edu/~virgi/stoctutorial.html`. Accessed: 2010-09-30.

[Wil05] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365, 2005.

[Wil13] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM Journal on Computing*, 42(3):1218–1244, 2013.

[Wil14] Ryan Williams. 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)*, page 80. ACM, 2014.

[WW10] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 645–654. IEEE, 2010.

Appendix A Baseline algorithm

This section gives an $O(n^{k-1}m)$ time algorithm solving $MC(k+1)$ with any quantifier structure for $k \geq 1$, thus proving Lemma 3.1, which states that the running time $T_k(n,m)$ of $MC(k+1)$ on graphs of n vertices and m edges follows the recurrence

$$\begin{aligned} &T_k(n,m) \leq n \cdot T_{k-1}(n, O(m)) + O(m), \text{ for } k \geq 2. \\ &T_1(n,m) = O(m). \end{aligned}$$

In this section we will use the conventions and assumptions given in Section 4.2.2.

Base Case. We prove that when $k = 1$, $T_k(n,m) = m$. For each $v_1 \in V_1$, the algorithm computes $\#(v_1) = |\{v_2 \in V_2 \mid (v_1, v_2) \models \psi\}|$. Thus we can list the sets of v_1 s.t $\#(v_1) > 0$ (if the inner quantifier is $\exists$), or those that satisfy $\#(v_1) = |V_2|$ (if it is $\forall$).

Algorithm 1 shows the details for counting $\#(v_1)$ for all v_1. Let P be the set of all predicates in ψ. Similarly to the proof of Lemma 4.5, we consider the $2^{|P|}$ different truth assignments of all predicates in P. Let an *interpretation* α be a binary string of length $|P|$, that encodes all truth values assigned to all predicates in P. Different interpretations are disjoint cases, so we treat them separately. For each interpretation α satisfying φ, we count the number of v_2's for each v_1 so that $(v_1, v_2) \models \alpha$. We consider two cases based on whether all binary predicates are false in α: If some binary predicate $R(x_1, x_2)$ is true as specified by α, then we can directly go through all edges in E_R incident on v_1, and enumerate the v_2's satisfying α. Otherwise if all binary predicates are false (so that α specifies there should be no edges connecting x_1 and x_2, i.e., the co-sparse case), then we can first over-count number of v_2, and then go through all edges incident on v_1 so as to exclude the over-counted v_2's.

Note that because $2^{|P|}$ is a constant number, the running time is linear to the sum of degrees of each v_1, or $O(m)$.

Algorithm 1: Counting $\#(v_1)$ for $MC_{(Q_1x_1)(Q_2x_2)\psi(x_1,x_2)}$

for Each interpretation $\alpha \in \{0,1\}^{|P|}$ **do**
    if α satisfies ψ **then**
        if Some binary predicates are true in α **then** // Sparse case
            for Each $v_1 \in V_1$ that agrees with all unary predicates on x_1 specified by α **do**
                Let $\#_\alpha(v_1)$ be the number of $v_2 \in V_2$ that agrees with all unary predicates on x_2, and that (v_1, v_2) agrees with all binary predicates on (x_1, x_2) and (x_2, x_1) specified by α
                $\#(v_1) = \#(v_1) + \#_\alpha(v_1)$
        else // co-sparse case
            Let $\#_1(v_1)$ be the number of $v_2 \in V_2$ that agrees with all unary predicates on x_2 specified by α // over-counting
            for Each $v_1 \in V_1$ that agrees with all unary predicates on x_1 specified by α **do**
                Let $\#_2(v_1)$ be the number of $v_2 \in V_2$ incident with v_1 that agrees with all unary predicates on x_2 specified by α // excluding v_2's that are adjacent to v_1
                Let $\#_\alpha(v_1) \leftarrow \#_1(v_1) - \#_2(v_1)$
                $\#(v_1) = \#(v_1) + \#_\alpha(v_2)$

Inductive Step. For $k \geq 2$, we give a quantifier-eliminating downward reduction, thus proving the recurrence relation. Assume $\varphi = (Q_1x_1)\dots(Q_{k+1}x_{k+1})\psi(x_1, \dots, x_{k+1})$ For each $v_1 \in V_1$, create new formula $\varphi_{v_1} = (Q_2x_2)\dots(Q_{k+1}x_{k+1})\psi(x_2, \dots, x_{k+1})$, and in ψ we replace each occurrence of unary predicate $R_i(x_1)$ with a constant $R_i(v_1)$, and replace each occurrence of binary predicate $R_i(x_1, x_j)$ (or $R_i(x_j, x_1)$) with unary predicate $R'_i(x_j)$ whose value equals $R_i(v_1, x_j)$ (or $R_i(x_j, v_1)$). Our algorithm enumerates all $v_1 \in V_1$, and then computes if the graph induced by $V_2, \dots, V_{k+1}$ satisfies φ_{v_1}. If x_1 is quantified by $\exists$, we accept iff any of them accepts. Otherwise we accept iff all of them accepts. The construction of φ_{v_1} takes time $O(m)$. The created graph has $O(n)$ vertices and $O(m)$ edges. Thus the recursion follows.

This process is a quantifier-eliminating downward reduction from a $MC(k+1)$ problem to a $MC(k)$ problem. It makes $O(m)$ queries, each of size $O(m)$. Then if problems in $MC(k)$ are solvable in time $O(m^{k-1-\epsilon})$, then problems in $MC(k+1)$ are solvable in time $m \cdot O(m^{k-1-\epsilon}) = O(m^{k-\epsilon})$. This quantifier-eliminating downward reduction implies that for problem $L_1 \in MC(k+1)$, there exists $L_2 \in MC(k)$ so that $(L_1, m^k) \leq_{FGT} (L_2, m^{k-1})$.

From the recursion and the base case, we have the running time $O(n^{k-1}m)$ by induction.

Appendix B Algorithms for easy cases

In this section we show that any $(k+1)$-quantifer problem with a quantifier sequence ending with $\exists\exists$ or $\forall\forall$ is solvable in time $O(m^{k-0.5})$. First of all, we use the quantiferi-eliminating downward reduction to reduce the problem to a $MC(3)$ problem. Then from the next two subsections we see that these problems are solvable in $O(m^{1.5})$.

Lemma B.1. *Problems in $MC(\exists\exists\exists)$ and $MC(\forall\forall\forall)$ are solvable in $O(m^{1.5})$.*

For problems in $MC(\forall\forall\forall)$, we decide its negation, which is a $MC(\exists\exists\exists)$ problem.

We define nine *Atomic Problems*, which are special $MC(3)$ problems. Let the Atomic Problem labeled by ℓ to be $MC_{(\exists x\in X)(\exists y\in Y)(\exists z\in Z)\,\psi_\ell}$, and referred to as $\Delta[\ell]$. It is defined on a tripartite graph on vertex sets (X,Y,Z), whose edge sets are E_{XY}, E_{YZ}, E_{XZ} defined on $(X,Y),(Y,Z),(X,Z)$ respectively. The graph is undirected, i.e., E_{XY}, E_{YZ} and E_{XZ} are symmetric relations. For simplicity we define an edge predicate E so that $E(v_1,v_2)$ is true iff there is an edge in any of E_{XY}, E_{YZ}, E_{XZ} connecting (v_1,v_2) or (v_2,v_1). Besides, we use $deg_Y(x)$ to denote the number of x's neighbors in Y.

The ψ_ℓ for all Atomic Problems are defined in the following table.

$\psi_2 = E(x,y)\wedge E(x,z)$	$\psi_{2+} = E(x,y)\wedge E(x,z)\wedge E(y,z)$	$\psi_{2-} = E(x,y)\wedge E(x,z)\wedge\neg E(y,z)$
$\psi_1 = E(x,y)\wedge\neg E(x,z)$	$\psi_{1+} = E(x,y)\wedge\neg E(x,z)\wedge E(y,z)$	$\psi_{1-} = E(x,y)\wedge\neg E(x,z)\wedge\neg E(y,z)$
$\psi_0 = \neg E(x,y)\wedge\neg E(x,z)$	$\psi_{0+} = \neg E(x,y)\wedge\neg E(x,z)\wedge E(y,z)$	$\psi_{0-} = \neg E(x,y)\wedge\neg E(x,z)\wedge\neg E(y,z)$

For problem MC_φ where $\varphi = (\exists x\in X)(\exists y\in Y)(\exists z\in Z)\,\psi(x,y,z)$, we write ψ as a DNF, and split the terms. Then we decide if there is a term so that there exist x,y,z satisfying this term. On each term t, which is a conjunction of predicates and negated predicates, we work on the induced subgraph whose vertices satisfy all the true unary predicates and unsatisfy all the false unary predicates defined on them in t. Then we can remove all unary predicates from the conjunction, which is now a conjunction of binary predicates or their negations. (If the conjunction is a single predicate or a single negated predicate, then we can deal with it easily, so we don't consider this case here.) If we define $E(x,y) = \bigwedge_{R\text{ is a positive binary predicate in }t} R(x,y)\wedge\bigwedge_{R\text{ is a negative binary predicate in }t}\neg R(x,y)$, and define $E(y,z)$ and $E(x,z)$ similarly, then t becomes equivalent with some Atomic Problem, or a disjunction of Atomic Problems (because variables y and z are interchangeable, the Atomic Problems and their disjunctions cover all possible cases).

In our algorithm for each problem $\Delta[\ell]$, instead of deciding the existence of satisfying x,y,z, we consider these problems as counting problems, where for each x we compute

$$\#_\ell(x) = |\{(y,z)\mid x,y,z\text{ satisfy }\psi_\ell\}|.$$

Problems $\Delta[2],\Delta[1],\Delta[0]$ can be computed straightforwardly.

- In $\Delta[2]$, $\#_2(x) = deg_Y(x)\times deg_Z(x)$.
- In $\Delta[1]$, $\#_1(x) = deg_Y(x)\times(|Z|-deg_Z(x))$.
- In $\Delta[0]$, $\#_0(x) = (|Y|-deg_Y(x))\times(|Z|-deg_Z(x))$.

Next we show for labels $\ell\in\{2+,1+,0+,2-,1-,0-\}$, problems $\Delta[\ell]$ can be computed in $O(m^{1.5})$.

Algorithm 2 solves $\Delta[2+]$, which is the triangle detection problem. The first part of the algorithm only considers small degree y. On each iteration of the outer loop, the inner loop is run for at most $\sqrt{m}$ times. The second part only considers large degree y. Because there are at most $\sqrt{m}$ of them, the outer loop is run for at most $\sqrt{m}$ times. Therefore the running time of the algorithm is $O(m^{1.5})$.

Algorithm 2: $\Delta[2+]$

for all $(x, y) \in E_{XY}$ **do** // Small degree y
  if $deg_Z(y) \leq \sqrt{m}$ **then**
    for all z s.t. $(y, z) \in E_{YZ}$ **do**
      if $(x, z) \in E_{XZ}$ **then**
        $\#_{2+}(x) \leftarrow \#_{2+}(x) + 1$
for all $y \in Y$ s.t. $deg_Z(y) > \sqrt{m}$ **do** // Large degree y
  for all $(x, z) \in E_{XZ}$ **do**
    if $(x, y) \in E_{XY}$ and $(y, z) \in E_{YZ}$ **then**
      $\#_{2+}(x) \leftarrow \#_{2+}(x) + 1$
if $\#_{2+}(x) > 0$ for some $x \in X$ **then** accept **else** reject

Algorithm 3 solves $\Delta[1+]$, which detects $(x - y - z)$ paths where there is no edge between x and z. The first part is similar as $\Delta[2+]$. The second part first over-counts $(x - y - z)$ paths for all large degree y without restricting the edge between x and z, and then counts the number of over-counted cases in order to exclude them from the final result. In the first block, the inner loop is run for at most $\sqrt{m}$ times for each edge in E_{XY}. The second block takes time $O(m)$. The outer loop of the third block is run for at most $\sqrt{m}$ times, because there are at most $\sqrt{m}$ sets with degree at least $\sqrt{m}$. So in all, the running time is $O(m^{1.5})$.

Algorithm 3: $\Delta[1+]$

for all $(x, y) \in E_{XY}$ **do** // Small degree y
  if $deg_Z(y) \leq \sqrt{m}$ **then**
    for all z s.t. $(y, z) \in E_{YZ}$ **do**
      if $(x, z) \notin E_{XZ}$ **then**
        $\#_{1+}(x) \leftarrow \#_{1+}(x) + 1$
for all $(x, y) \in E_{XY}$ **do** // Large degree y
  if $deg_Z(y) \geq \sqrt{m}$ **then** // Over-counting
    $\#_{1+}(x) = \#_{1+}(x) + deg_Z(y)$
for all $y \in Y$ s.t. $deg_Z(y) > \sqrt{m}$ **do**
  for all $(x, z) \in E_{XZ}$ **do** // for all z connected to x
    if $(x, y) \in E_{XY}$ and $(y, z) \in E_{YZ}$ **then** // if we just over-counted the pair (y, z)
      $\#_{1+}(x) \leftarrow \#_{1+}(x) - 1$ // then we exclude the pair by subtracting one.
if $\#_{1+}(x) > 0$ for some $x \in X$ **then** accept **else** reject

For $\Delta[0+]$, we first compute $\#_{2+}(x)$ which is the result of $\Delta[2+]$, and then compute $\#_{1+}(x)$ and $\#'_{1+}(x)$, which are results of $\Delta[1+]$ on vertex sets (X, Y, Z) and (X, Z, Y) respectively. Finally let $\#_{0+}(x) \leftarrow |E_{YZ}| - (\#_{2+}(x) + \#_{1+}(x) + \#'_{1+}(x))$.

$\#_{2-}(x), \#_{1-}(x), \#_{0-}(x)$ can be computed by respectively taking the differences of $\#_2(x), \#_1(x), \#_0(x)$ and $\#_{2+}(x), \#_{1+}(x), \#_{0+}(x)$.

Lemma B.2. *Problems in $MC(\forall\exists\exists)$ and $MC(\exists\forall\forall)$ are solvable in $O(m^{1.5})$.*

For problems in $MC(\exists\forall\forall)$, we decide its negation, which is a $MC(\forall\exists\exists)$ problem.

For problem MC_φ where $\varphi = (\forall x \in X)(\exists y \in Y)(\exists z \in Z)\, \psi(x, y, z)$, we use the same algorithm to compute $\#_\ell(x)$ for all $x \in X$. If the value of $\#_\ell(x)$ is greater than zero for all $x \in X$, then we accept, otherwise reject. Again, we write ψ as a DNF, and split the terms. By the same argument as the previous lemma, we transform the problem to a disjunction of Atomic Problems. If for all $x \in X$, at least in one of the Atomic Problem, $\#_\ell(x)$ is greater than zero, then we accept, otherwise reject.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de