OV graphs are (probably) hard instances

Josh Alman[*] Virginia Vassilevska Williams[†]

Abstract

A graph G on n nodes is an Orthogonal Vectors (OV) graph of dimension d if there are vectors $v_1, \ldots, v_n \in \{0,1\}^d$ such that nodes i and j are adjacent in G if and only if $\langle v_i, v_j \rangle = 0$ over $\mathbb{Z}$. In this paper, we study a number of basic graph algorithm problems, except where one is given as input the vectors defining an OV graph instead of a general graph. We show that for each of the following problems, an algorithm solving it faster on such OV graphs G of dimension only $d = O(\log n)$ than in the general case would refute a plausible conjecture about the time required to solve sparse MAX-k-SAT instances:

- Determining whether G contains a triangle.
- More generally, determining whether G contains a directed k-cycle for any $k \geq 3$.
- Computing the square of the adjacency matrix of G over $\mathbb{Z}$ or $\mathbb{F}_2$.
- Maintaining the shortest distance between two fixed nodes of G, or whether G has a perfect matching, when G is a dynamically updating OV graph.

We also prove some complementary results about OV graphs. We show that any problem which is NP-hard on constant-degree graphs is also NP-hard on OV graphs of dimension $O(\log n)$, and we give two problems which can be solved faster on OV graphs than in general: Maximum Clique, and Online Matrix-Vector Multiplication.

[*]Harvard University, jalman@seas.harvard.edu. Supported by a Michael O. Rabin Postdoctoral Fellowship.

[†]Supported by an NSF CAREER Award, NSF Grants CCF-1528078, CCF-1514339 and CCF-1909429, a BSF Grant BSF:2012338, a Google Research Fellowship and a Sloan Research Fellowship.

1 Introduction

Two of the most studied conjectures in fine-grained complexity are the Strong Exponential Time Hypothesis (SETH), and the Orthogonal Vectors Conjecture (OVC). SETH was introduced by Impagliazzo, Paturi and Zane [IPZ01] regarding the complexity of k-SAT:

Hypothesis 1.1 (Strong Exponential Time Hypothesis)**.** *For every $\varepsilon > 0$, there is an integer $k \geq 3$ such that k-SAT on n variables cannot be solved in $O(2^{(1-\varepsilon)n})$ (randomized) time.*

OVC concerns the Orthogonal Vectors (OV) problem: Given as input a set $A \subseteq \{0,1\}^d$ of $|A| = n$ vectors, determine whether there are $a, b \in A$ such that $\langle a, b\rangle = 0$ (all inner products in this paper, including this one, are taken over $\mathbb{Z}$ unless stated otherwise).

Hypothesis 1.2 (Orthogonal Vectors Conjecture)**.** *For every $\varepsilon > 0$, there is a $c > 0$ such that OV in dimension $d = c \log n$ cannot be solved in $O(n^{2-\varepsilon})$ (randomized) time.*

Williams [Wil05] showed that SETH implies OVC. Most of the known fine-grained implications of SETH use this result and are actually proved assuming OVC instead; see [Vas18] for a survey of the many applications of OVC to graph algorithms, string algorithms, nearest neighbors problems, and more.

In this paper, we study a mathematical object inspired by OVC which we call an OV graph: a graph G on n nodes is an OV graph of dimension d if there are n vectors $v_1, \ldots, v_n \in \{0,1\}^d$ such that nodes i and j are adjacent in G if and only if $\langle v_i, v_j\rangle = 0$. Given as input the vectors $v_1, \ldots, v_n \in \{0,1\}^d$ defining G, there are a number of natural algorithmic questions one might ask, including:

- OV: does G have any edges?
- OVΔ: does G contain any triangles?
- OV-DIR-k-CYCLE: given a partition of the nodes of G into k parts, is there a k-cycle containing one node from each part?

Detecting triangles and more generally, k-cycles are among the most basic algorithmic questions one can ask about graphs. OV graphs of low dimension $d \ll n$ make up a small fraction of all graphs: there are only $2^{O(nd)}$ such graphs on n nodes, compared to $2^{\Theta(n^2)}$ total graphs on n nodes. However, in this paper we will show that solving these problems on OV graphs of dimension only $d = O(\log n)$ may be just as hard as solving them in general graphs. Somewhat analogously to how Williams showed that faster algorithms for OV in dimension $O(\log n)$ would lead to breakthroughs in solving k-SAT, we will show that faster algorithms for OVΔ or OV-DIR-k-CYCLE on OV graphs of dimension $O(\log n)$ would lead to breakthroughs in solving MAX-k-SAT.

MAX-k-SAT In the MAX-k-SAT problem for an integer $k \geq 2$, given as input a k-CNF formula ϕ, the goal is to determine the maximum number of clauses of ϕ which can be satisfied by a single assignment.

MAX-k-SAT on n variables and m clauses can be solved in $O(2^n m)$ time by exhaustive search. Williams [Wil05, Wil07] showed that MAX-2-SAT has a much faster, $2^{\omega n/3}\text{poly}(n)$ time algorithm, where $\omega < 2.373$ is the exponent of matrix multiplication [Vas12, Gal14]. This running time for

MAX-2-SAT has remained unchallenged for over 15 years. It is an interesting open problem whether a faster algorithm exists.

Williams' techniques for MAX-2-SAT do not carry over to MAX-k-SAT for $k \geq 3$ (see [LVW18] for a discussion), and there is no known $O((2-\varepsilon)^n)$ time algorithm for MAX-k-SAT for any $k \geq 3$ and $\varepsilon > 0$.

Unlike with k-SAT [IP01], there is no known sparsification lemma for MAX-k-SAT, so that in principle MAX-k-SAT on formulas with $O(n)$ clauses might be easier than the general case of MAX-k-SAT that might have n^k clauses. This has led researchers to investigate the complexity of such sparse instances of MAX-k-SAT (e.g. [DW06, CS15, ACW16]).

The fastest known algorithms for MAX-k-SAT on n variables and cn clauses for constant c run in time $2^{n(1-1/O(\log^2 c))}\text{poly}(n)$ when $k \leq 4$, or in time $2^{n(1-1/O(c^{1/3}))}\text{poly}(n)$ when $k > 4$ [ACW16]. Unfortunately, as c grows, these running times go to 2^n, the brute force running time. Thus the following hypothesis is fully consistent with the state-of-the art of MAX-k-SAT algorithms:

Hypothesis 1.3 (Sparse MAX-3-SAT Hypothesis)**.** *For every $\varepsilon > 0$, there exists a $c > 0$ so that n variable MAX-3-SAT on cn clauses cannot be solved in time $O(2^{n(1-\varepsilon)})$.*

The hypothesis above strengthens an earlier hypothesis that MAX-3-SAT requires $2^{n(1-o(1))}$ time (see e.g. [Vas18, LVW18]); it would be equivalent to that hypothesis if a sparsification lemma for MAX-3-SAT can be proven. We will base some of our hardness results on our strengthened hypothesis. We will also use an analogous hypothesis for Sparse MAX-2-SAT:

Hypothesis 1.4 (Sparse MAX-2-SAT Hypothesis)**.** *For every $\varepsilon > 0$, there exists a $c > 0$ so that n variable MAX-2-SAT on cn clauses cannot be solved in time $O(2^{n(\omega/3-\varepsilon)})$.*

1.1 Our Results

Triangle Finding and Matrix Multiplication The best known algorithm for finding a triangle in a graph on n nodes runs in time $n^{\omega+o(1)}$, where $\omega \leq 2.373$ is the matrix multiplication exponent [Vas12, Gal14]. Our first result is that if there is a faster algorithm that finds triangles in *OV graphs*, then Hypothesis 1.4 would be violated and sparse MAX-2-SAT would have faster algorithms.

Theorem 1.1. *Suppose OVΔ in OV graphs with n nodes and dimension $O(\log n)$ can be solved in time $n^{\omega-\varepsilon+o(1)}$ for some constant $\varepsilon > 0$. Then, for any constants $a, \delta > 0$, MAX-2-SAT on n variables and $a \cdot n$ clauses can be solved in time $2^{(\omega/3-\varepsilon/3+\delta)n}$.*

The best known algorithm for triangle finding in a general graph G works by reducing to the *Boolean matrix multiplication* of two copies of the adjacency matrix of G. In Boolean matrix multiplication, given as input two matrices $A, B \in \{0,1\}^{n\times n}$, the goal is to compute their product over the (AND,OR) semiring, i.e. the matrix $C \in \{0,1\}^{n\times n}$ given by $C[i,j] = \bigvee_{k=1}^{n} A[i,k] \wedge B[k,j]$. This can be solved in time $n^{\omega+o(1)}$ using a simple reduction to matrix multiplication over either $\mathbb{F}_2$ or $\mathbb{Z}$.

Similarly, OVΔ in OV graphs with n nodes and dimension d has a simple linear-time reduction to Boolean matrix multiplication of $n \times n$ matrices which are the adjacency matrices of *OV* graphs of dimension d. In other words, these are matrices $A \in \{0,1\}^{n\times n}$ for which there are vectors $v_1, \ldots, v_n \in \{0,1\}^d$ such that $A[i,j] = 1$ if and only if $\langle v_i, v_j \rangle = 0$. If such matrices for $d = O(\log n)$ can be multiplied in $n^{\omega-\varepsilon+o(1)}$ time for some constant $\varepsilon > 0$, it would lead to a corresponding speedup for MAX-2-SAT on n variables and $O(n)$ clauses as well. In other words, this small set of only $O(2^{nd}) = O(2^{n \log n})$ matrices with such efficient descriptions may be just as difficult to multiply as arbitrary $\{0,1\}$ matrices (of which there are $2^{\Theta(n^2)}$).

Consequences of the n^ω hardness of OVΔ for dynamic algorithms. Because of its simplicity, triangle detection has been reduced to many other problems. For instance, Abboud and Vassilevska Williams [AV14] present several clean reductions from triangle detection to a variety of dynamic problems. We next show that, as a consequence of the hardness of OVΔ, many of the triangle-based lower bounds in [AV14] also hold for OV graphs with $O(\log n)$ dimension.

Typically, in dynamic graph algorithms one supports edge insertions and deletions. In OV graphs, however, the edge relation is captured by the labels on the vertices of the graph. Thus, in dynamic OV graphs, we instead support vertex label updates as above. One could also consider an alternate model where the updates may only change one bit of a vertex label, but update times in these two models differ by at most a fairly negligible $O(d)$ factor.

Although changing a vertex label can change all the incident edges to the vertex, our lower bounds apply even when each update changes at most a *single* edge. Hence the lower bounds would also apply in the standard dynamic graph algorithms model if one maintains the OV graph as a graph, rather than as a set of vectors.

Assuming that OVΔ requires $n^{\omega-o(1)}$ time (as follows from Hypothesis 1.4), we obtain the following conditional lower bounds:

- Dynamic s-t OV Shortest Paths, maintaining the distance between two fixed vertices s and t in an n node OV graph with dimension $O(\log n)$ under vertex relabel updates (change the vector representing a node) requires either $n^{\omega-o(1)}$ preprocessing time, or $n^{\omega-1-o(1)}$ amortized update time; the lower bound holds even if the updates insert or delete at most a single edge. The same lower bounds hold for $5/3-\varepsilon$-approximating the s-t distance when arbitrary label updates are allowed, even when the preprocessing time can be arbitrary.

- Dynamic bipartite perfect matching in OV graphs on n nodes and dimension $O(\log n)$ under vertex relabel updates requires either $n^{\omega-o(1)}$ preprocessing time, or $n^{\omega-1-o(1)}$ amortized update time. The lower bound holds even if the updates insert or delete at most a single edge. If the updates can be arbitrary, the lower bound holds for arbitrary preprocessing time.

In the full version of the paper we present more such reductions. Because of the structure of our reductions, the lower bounds we prove in which the relabelings change only a single edge also hold for incremental algorithms (where edges are only inserted), but the lower bound is only for worst case update time.

Directed k-Cycle. The fastest known algorithm for finding a k-cycle for any constant $k \geq 3$ in a general directed n-node graph runs in $n^{\omega+o(1)}$ time via color-coding and matrix multiplication [AYZ95]. Our next result is that under Hypothesis 1.4, this running time is essentially tight even in OV graphs:

Theorem 1.2. *Let $k \geq 3$ be any constant. Suppose that k-Cycle in directed OV graphs with n nodes and dimension $O(\log n)$ can be solved in time $n^{\omega-\varepsilon+o(1)}$ for some constant $\varepsilon > 0$. Then, for any constants $a, \delta > 0$, MAX-2-SAT on n variables and $a \cdot n$ clauses can be solved in time $2^{(\omega/3-\varepsilon/3+\delta)n}$.*

As our MAX-3-SAT Hypothesis 1.3 is potentially more believable than our MAX-2-SAT Hypothesis 1.4, we further investigate what it implies for k-Cycle detection. We show that unless Hypothesis 1.3 fails, there is no constant k for which k-cycle in an n node OV graph with dimension $O(\log n)$ can be found in $O(n^{3/2-\varepsilon})$ time for any $\varepsilon > 0$. The statement of our directed k-cycle results under Hypothesis 1.3 are strongest for $k = 4$:

Theorem 1.3. *Suppose that OV-DIR-4CYCLE in OV graphs with n nodes and dimension $O(\log n)$ can be solved in time $n^{2-\varepsilon+o(1)}$ for some constant $\varepsilon > 0$. Then, for any constants $a, \delta > 0$, MAX-3-SAT on n variables and $a \cdot n$ clauses can be solved in time $2^{(1-\varepsilon/2+\delta)n}$, and Hypothesis 1.3 fails.*

If $\omega = 2$, the above would give an essentially tight lower bound under a better hypothesis. While the lower bounds we obtain under Hypothesis 1.3 are not nearly as strong as the tight results we get under Hypothesis 1.4, they are conditioned on a slightly more believable hypothesis. Moreover, the techniques seem to be slightly more flexible, so that there might be hope that a similar result might hold for k-cycle in *undirected* graphs. When k is odd, our hardness results already hold for k-cycle in undirected OV graphs, but when k is even, k-cycle in general undirected graphs can be solved in $O(n^2)$ time regardless of k [YZ97]. Hence our $n^{\omega-o(1)}$ lower bound from Hypothesis 1.4 may not extend to undirected graphs if $\omega > 2$. A strengthening of our MAX-3-SAT-based reductions to undirected graphs might still be possible, and would be significant as it would be tight, at least for 4-cycles.

Constant Degree Graphs have low OV dimension Next, we study the complexity of various NP-hard graph problems when they are restricted to graphs on n nodes with OV dimension $O(\log n)$. We begin with problems which are known to be NP-hard on constant-degree graphs, including Hamiltonian Path and Minimum Vertex Cover. It follows from prior work [Alo86] that constant-degree graphs have OV dimension $O(\log n)$. However, the proof of this is nonconstructive, and uses the probabilistic method. We nonetheless show how to derandomize this proof, giving a deterministic polynomial-time algorithm for finding a representation of a constant-degree graph as an OV graph of dimension $O(\log n)$. We hence show:

Theorem 1.4. *Any problem which is NP-hard on graphs of constant maximum degree is also NP-hard on OV graphs of dimension $O(\log n)$.*

We also show that at least one NP-hard problem, the Max Clique problem, seems to become easier on OV graphs of dimension $d \ll n$, by reducing to a set packing problem:

Theorem 1.5. *Given as input vectors $V = \{v_1, \ldots, v_n\} \subseteq \{0,1\}^d$ defining a dimension d OV graph G_V, a maximum size clique in G_V can be found in time $2^d \cdot n^{O(1)}$.*

Online Matrix-Vector Multiplication Finally, we study the Online Matrix-Vector Multiplication (OMV) problem: preprocess a matrix $M \in \mathbb{F}_2^{n\times n}$ so that, given as input a query vector $v \in \mathbb{F}_2^n$, one can quickly return the product $M \cdot v$. The best known algorithms in general answer queries in time $n^{2-o(1)}$ [LW17]. We show that faster algorithms are possible when the matrix M is the adjacency matrix of a graph with OV dimension $c \log n$:

Theorem 1.6. *For $c > 0$, we can preprocess vectors $u_1, \ldots, u_n, v_1, \ldots, v_n \in \{0,1\}^{c \log n}$, which define a matrix $M \in \mathbb{F}_2^{n\times n}$ as $M[i,j] = 1$ if and only if $\langle u_i, v_j \rangle = 0$, in preprocessing time $\tilde{O}(n^2)$*

with high probability, such that given as input a vector $v \in \mathbb{F}_2^n$, we can compute the product $M \cdot v$ in time $n^{2-1/O(\log c)}$.

Data structures for OMV have many applications. For instance, by multiplying the adjacency matrix of a graph G by an indicator vector for a subset $S \subseteq \{1, 2, \ldots, n\}$ of the nodes of G, one gets the neighborhood of S. Similar to [LW17, Corollary 1.1], our Theorem 1.6 thus yields a data structure with $\tilde{O}(n^2)$ preprocessing time for OV graphs G of dimension $c \cdot \log n$ which, given as a query a subset S of the nodes of G, can answer whether S is independent, dominating, or a vertex cover, in time $n^{2-1/O(\log c)}$.

Our data structure for Theorem 1.6 works by expressing the matrix M as the sum of a matrix of rank $n^{0.1}$ and a sparse matrix with $n^{2-1/O(\log c)}$ nonzero entries. It does this by combining the algorithm for OV by Abboud et al. [AWY15], which makes use of the 'polynomial method in algorithm design', together with known techniques for converting polynomial method constructions into 'matrix rigidity' upper bounds [AW17].

1.2 Other Related Work

OV graphs have been studied in a number of other settings in mathematics and computer science.

The Edge Clique Cover Number (ECCN) of a graph G is the minimum number of cliques needed to cover the edges of G. We can see that a graph G is an OV graph of dimension d if and only if the ECCN of the *complement* of G is d. A line of work in the combinatorics community has shown that a number of simple classes of graphs have low ECCN; see e.g. [EGP66, Alo86, Gyá90, MPR95, MQ06].

The complexity of determining the ECCN of an input graph has also been studied. The problem is known to be NP-hard in general, but Gramm et al. [GGHN09] gave a parameterized algorithm running in time $2^{2^{O(k)}} \cdot \text{poly}(n)$ on an n node graph with ECCN k. Such a doubly-exponential dependence on k may be optimal: Cygan et al. [CPP16] showed that a $2^{2^{o(k)}} \cdot \text{poly}(n)$ time algorithm would refute the Exponential Time Hypothesis (a weak version of SETH).

Representations of graphs by the orthogonality relations of vectors have also been studied in other areas. For instance, Lovász [Lov79] describes 'orthonormal representations' where each node must be assigned a unit vector in Euclidean space, and the vectors corresponding to nonadjacent vertices must be orthogonal. In this language, if G is a graph with low OV dimension, then the complement of G has low $\{0, 1\}$-faithful orthogonality dimension; see e.g. [LSS89, Lov19].

2 Preliminaries

2.1 Notation

For a positive integer d, write 0^d for the all-zeroes vector of dimension d. Write $||$ to denote concatenation of vectors.

Define $AND : \{0, 1\}^d \times \{0, 1\}^d \to \{0, 1\}^d$, the bit-wise AND function by, for $x, y \in \{0, 1\}^d$ and $\ell \in \{1, \ldots, d\}$, $AND(x, y)[\ell] = x[\ell] \cdot y[\ell]$.

2.2 OV Graphs

Definition 2.1 (OV Graph). *For positive integers n, d and vectors $V = \{v_1, \ldots, v_n\} \subseteq \{0, 1\}^d$, the OV graph G_V is the graph on n nodes where nodes i and j are adjacent if and only if $\langle v_i, v_j \rangle = 0$,* *where the inner product is over $\mathbb{Z}$. The OV dimension of a graph G is the smallest nonnegative integer d such that G can be written as the OV graph of vectors in $\{0,1\}^d$.*

Definition 2.2 (Generalized Inner Product). *For any positive integers n, d and vectors $v_1, \dots, v_n \in \mathbb{Z}^d$, we define the generalized inner product*

$$\mathsf{IP}(\{v_1, \dots, v_n\}) = \langle v_1, \dots, v_n \rangle = \sum_{\ell=1}^{d} v_1[\ell] \cdot v_2[\ell] \cdots v_n[\ell].$$

The sum (as well as all inner products in this paper) is taken over $\mathbb{Z}$.

Definition 2.3 (k-Uniform OV Hypergraph). *For positive integers n, d and vectors $V = \{v_1, \dots, v_n\} \subseteq \{0,1\}^d$, the k-uniform OV hypergraph $G_{V,k}$ is the k-uniform hypergraph on n nodes where, for distinct nodes $i_1, i_2, \dots, i_k$, the hyperedge $(i_1, \dots, i_k)$ is in $G_{V,k}$ if and only if $\langle v_{i_1}, v_{i_2}, \dots, v_{i_k} \rangle = 0$.*

Definition 2.4 (OV dimension of a matrix). *The OV dimension of a matrix $M \in \{0,1\}^{n \times n}$ is the smallest nonnegative integer d such that there are $2n$ vectors $u_1, \dots, u_n, v_1, \dots, v_n \in \{0,1\}^d$ with the property that, for any $i, j \in \{1, 2, \dots, n\}$, we have $M[i,j] = 1$ if and only if $\langle u_i, v_j \rangle = 0$. Equivalently, the OV dimension of M is the rank of $\bar{M}$ (the all 1s matrix minus M) over the OR, AND (Boolean) semiring.*

Remark 2.1. The adjacency matrix of an OV graph of dimension d is a matrix of OV dimension d.

2.3 Algorithmic Problems

Definition 2.5. *We define some problems. For positive integers n, d:*

- $\mathsf{OV}_{n,d}$: *Given $A, B \subseteq \{0,1\}^d$ with $|A| = |B| = n$, determine whether there is an $(a,b) \in A \times B$ with $\langle a, b \rangle = 0$.*
- $\mathsf{EXACT\text{-}IP}_{n,d}$: *Given $A, B \subseteq \{0,1\}^d$ with $|A| = |B| = n$, and an integer $0 \le m \le d$, determine whether there is an $(a,b) \in A \times B$ with $\langle a, b \rangle = m$.*
- $\mathsf{OV\Delta}_{n,d}$: *Given $V \subseteq \{0,1\}^d$ with $|V| = n$, determine whether there are distinct $a, b, c \in V$ with $\langle a, b \rangle = \langle b, c \rangle = \langle c, a \rangle = 0$.*
- $\mathsf{EXACT\text{-}IP\Delta}_{n,d}$: *Given $A, B, C \subseteq \{0,1\}^d$ with $|A| = |B| = |C| = n$, and three integers $0 \le m_{AB}, m_{BC}, m_{CA} \le d$, determine whether there is a $(a,b,c) \in A \times B \times C$ with $\langle a, b \rangle = m_{AB}$, $\langle b, c \rangle = m_{BC}$, and $\langle c, a \rangle = m_{CA}$.*
- $\mathsf{OV\text{-}HYPERGRAPH}_{n,d,\ell,k}$ *(for $\ell > k \ge 2$): Given $V \subseteq \{0,1\}^d$ with $|V| = n$, determine whether there is a $T \subseteq V$ of size $|T| = \ell$ such that for all $S \subseteq T$ of size $|S| = k$ we have $\mathsf{IP}(S) = 0$.*
- $\mathsf{DIRECTED\text{-}CYCLE}_{n,d,k}$: *Given $V_1, \dots, V_k \subseteq \{0,1\}^d$ with $|V_1| = \cdots = |V_k| = n$, determine whether there is a $v_i \in V_i$ for each $i \in \{1, 2, \dots, k\}$ such that $\langle v_i, v_{i+1} \rangle = 0$ for all such i (with $v_{k+1} = v_1$).*
- $\mathsf{MAX\text{-}k\text{-}SAT}_{n,m}$: *Given a k-CNF formula ϕ on n variables and m clauses, determine the maximum number of clauses of ϕ which can be satisfied by an assignment.*

3 Reduction from MAX-2SAT to Finding Triangles in OV Graphs

Lemma 3.1. *There is a polynomial-time reduction from $\mathsf{MAX\text{-}2\text{-}SAT}_{n,m}$ to $O(m^3)$ instances of $\mathsf{EXACT\text{-}IP}\Delta_{3\cdot 2^{\lceil n/3\rceil},m}$.*

Proof. Partition the input variables into three groups X, Y, Z of $n/3$ variables each. Let S be the set of clauses of ϕ. Partition S into six sets $S_X, S_Y, S_Z, S'_X, S'_Y, S'_Z$ such that:

- S'_X contains the clauses which consist only of literals of variables from X, and S'_Y and S'_Z are defined similarly.
- $S_X \subseteq S \setminus (S'_X \cup S'_Y \cup S'_Z)$ contains the clauses which contain one literal from Y and one from Z, and S_Y and S_Z are defined similarly.

For each of the $O(m^3)$ choices of $0 \leq m_X, m_Y, m_Z \leq m$, we will determine whether it is possible to find an assignment to ϕ which satisfies all but m_X clauses of $S'_Y \cup S_Z$, all but m_Y clauses of $S'_Z \cup S_X$, and all but m_Z clauses of $S'_X \cup S_Y$. From this we can compute the desired answer.

For any assignment $\alpha : X \to \{0,1\}^{n/3}$, define the vector $v_\alpha \in \{0,1\}^m$, whose entries are indexed by clauses c in S, by

$$v_\alpha[c] = \begin{cases} 0 & \text{if } c \in S'_X \cup S_Y \cup S_Z \text{ and } \alpha \text{ satisfies a literal in } c, \\ 0 & \text{if } c \in S'_Z \cup S_X, \\ 1 & \text{otherwise.} \end{cases}$$

Similarly, for any assignment $\beta : Y \to \{0,1\}$, define $v_\beta \in \{0,1\}^m$ by

$$v_\beta[c] = \begin{cases} 0 & \text{if } c \in S_X \cup S'_Y \cup S_Z \text{ and } \alpha \text{ satisfies a literal in } c, \\ 0 & \text{if } c \in S'_X \cup S_Y, \\ 1 & \text{otherwise.} \end{cases}$$

and for any assignment $\gamma : Z \to \{0,1\}$, define $v_\gamma \in \{0,1\}^m$ by

$$v_\gamma[c] = \begin{cases} 0 & \text{if } c \in S_X \cup S_Y \cup S'_Z \text{ and } \alpha \text{ satisfies a literal in } c, \\ 0 & \text{if } c \in S'_Y \cup S_Z, \\ 1 & \text{otherwise.} \end{cases}$$

Note that for any assignment s to all the variables, letting $\alpha = s|_X$, $\beta = s|_Y$, and $\gamma = s|_Z$, we have that $\langle v_\alpha, v_\beta\rangle$ counts the number of clauses in $S'_Y \cup S_Z$ which are unsatisfied by s, $\langle v_\beta, v_\gamma\rangle$ counts the number of clauses in $S'_Z \cup S_X$ which are unsatisfied by s, and $\langle v_\gamma, v_\alpha\rangle$ counts the number of clauses in $S'_X \cup S_Y$ which are unsatisfied by s. In other words, our goal is to determine whether there is an $\alpha : X \to \{0,1\}$, $\beta : Y \to \{0,1\}$, and $\gamma : Z \to \{0,1\}$ with $\langle v_\alpha, v_\beta\rangle = m_X$, $\langle v_\beta, v_\gamma\rangle = m_Y$, and $\langle v_\gamma, v_\alpha\rangle = m_Z$. This is exactly an instance of $\mathsf{EXACT\text{-}IP}\Delta_{3\cdot 2^{\lceil n/3\rceil},m}$, as desired. $\square$

Theorem 3.1 (Reduction from $\mathsf{EXACT\text{-}IP}$ to OV implicit in [CW19, Proof of Lemma 4.2])**.** *For any positive integer n, and any $c, \varepsilon > 0$, set $s := n^{\varepsilon \log(c/\varepsilon)}$ and $d := O(2^{c/\varepsilon} \log n)$. There is a pair of maps*

$$r_1, r_2 : \{0,1\}^{c \log n} \times \{1, 2, \ldots, s\} \times \{0, 1, \ldots, c \log n\} \to \{0,1\}^d$$

which can be computed in deterministic time $O(s \cdot 2^{c/\varepsilon} \log n)$ such that for any $x, y \in \{0,1\}^{c \log n}$, and any $m \in \{0, 1, \ldots, c \log n\}$, we have $\langle x, y\rangle = m$ if and only if there is an $i \in \{1, 2, \ldots, s\}$ such that $\langle r_1(x,i,m), r_2(y,i,m)\rangle = 0$ (over $\mathbb{Z}$).

Lemma 3.2. *For every $c, \varepsilon > 0$, there is a reduction from an $\mathsf{EXACT\text{-}IP}\Delta_{n,c\log n}$ instance to $n^{3\varepsilon \log(c/\varepsilon)}$ many instances of $\mathsf{OV}\Delta_{n,O(2^{c/\varepsilon}\log n)}$. The $\mathsf{EXACT\text{-}IP}\Delta$ instance is a yes instance if and only if at least one of the $\mathsf{OV}\Delta$ instances is a yes instance. The reduction takes time $2^{O(\varepsilon \log(c/\varepsilon)\log n + c/\varepsilon)}$.*

Proof. In our $\mathsf{EXACT\text{-}IP}\Delta_{n,c\log n}$ instance, we are given $A, B, C \subseteq \{0,1\}^{c\log n}$ with $|A| = |B| = |C| = n$ and three nonnegative integers m_{AB}, m_{BC}, m_{CA}, and our goal is to determine whether there is a $(a,b,c) \in A \times B \times C$ such that $\langle a, b\rangle = m_{AB}$, $\langle b, c\rangle = m_{BC}$, and $\langle c, a\rangle = m_{CA}$.

Set $s := n^{\varepsilon \log(c/\varepsilon)}$ and $d := O(2^{c/\varepsilon}\log n)$, and let r_1, r_2 be the maps from Theorem 3.1. Then, our goal is equivalently to determine whether there is a $(a,b,c) \in A \times B \times C$ and $i_{AB}, i_{BC}, i_{CA} \in \{1, 2, \ldots, s\}$ such that $\langle r_1(a, i_{AB}, m_{AB}), r_2(b, i_{AB}, m_{AB})\rangle = \langle r_1(b, i_{BC}, m_{BC}), r_2(c, i_{BC}, m_{BC})\rangle = \langle r_1(c, i_{CA}, m_{CA}), r_2(a, i_{CA}, m_{CA})\rangle = 0$.

For each fixed choice of $i_{AB}, i_{BC}, i_{CA} \in \{1, 2, \ldots, s\}$, this is very nearly an instance of $\mathsf{OV}\Delta_{n,d}$, which would complete the proof. To convert it into an instance of $\mathsf{OV}\Delta_{n,3d}$, simply map:

$$a \to r_1(a, i_{AB}, m_{AB})||0^d||r_2(a, i_{CA}, m_{CA}),$$

$$b \to r_2(b, i_{AB}, m_{AB})||r_1(b, i_{BC}, m_{BC})||0^d,$$

$$c \to 0^d||r_2(c, i_{BC}, m_{BC})||r_1(c, i_{CA}, m_{CA}),$$

for each $a \in A$, $b \in B$, $c \in C$, where $||$ denotes concatenation of vectors, and 0^d is the all-0s vector of length d. ☐

Theorem 3.2. *Suppose that $\mathsf{OV}\Delta_{N,c\log N}$ can be solved in (deterministic) time $N^{\tau+o(1)}$ for some constant $\tau \geq 0$ and all constants $c > 0$. Then, for any constant $a > 0$, $\mathsf{MAX\text{-}2\text{-}SAT}_{n,an}$ can be solved in (deterministic) time $O(2^{(\tau/3+\delta)n})$ for every $\delta > 0$.*

Proof. Let $\varepsilon > 0$ be a small constant to be set later. Lemma 3.1 gives a reduction from $\mathsf{MAX\text{-}2\text{-}SAT}_{n,an}$ to $O(n^3)$ instances of $\mathsf{EXACT\text{-}IP}\Delta_{O(2^{n/3}),an}$. Lemma 3.2 then reduces each of those to $2^{O(n\varepsilon\log(a/\varepsilon))}$ instances of $\mathsf{OV}\Delta_{O(2^{n/3}),O(n\cdot 2^{3a/\varepsilon})}$. The total time for computing all these reductions is $2^{O(n\varepsilon\log(a/\varepsilon)+a/\varepsilon)}$.

For any $\delta > 0$, we can pick a sufficiently small $\varepsilon > 0$ so that the combination is a reduction from $\mathsf{MAX\text{-}2\text{-}SAT}_{n,an}$ to $2^{\delta n/2}$ instances of $\mathsf{OV}\Delta_{O(2^{n/3}),O(n)}$, which can be computed in $2^{\delta/2}$ time. Each of those instances can be solved in time $2^{\tau n/3+o(n)}$ using the given algorithm with $N = O(2^{n/3})$, as desired. ☐

Finally, we can extend Theorem 3.2 from showing hardness just for $\mathsf{OV}\Delta_{n,O(\log n)}$ to showing hardness for $\mathsf{DIRECTED\text{-}CYCLE}_{n,O(\log n),k}$ for any integer $k \geq 3$ (the two problems coincide when $k = 3$) via a simple reduction:

Lemma 3.3. *For any integer $k \geq 3$, there is a linear-time reduction from $\mathsf{OV}\Delta_{n,O(\log n)}$ to $\mathsf{DIRECTED\text{-}CYCLE}_{n,O(\log n),k}$.*

Proof. $\mathsf{OV}\Delta_{n,O(\log n)}$ reduces to $\mathsf{DIRECTED\text{-}CYCLE}_{n,O(\log n),3}$ by simply repeating the given input set three times (unless it contains the all-zeroes vector, in which case we include it in only one of the three sets). We next show how to reduce from $\mathsf{DIRECTED\text{-}CYCLE}_{n,O(\log n),k-1}$ to $\mathsf{DIRECTED\text{-}CYCLE}_{n,O(\log n),k}$ for any $k \geq 4$, which will complete the proof.

We are given as input $k-1$ sets $V_1, \dots, V_{k-1} \subseteq \{0,1\}^d$ for $d = O(\log n)$ and $|V_i| = n$ for all $i \in \{1, 2, \dots, k-1\}$. For each i write $V_i = \{v_{i,j}\}_{j \in \{1,\dots,n\}}$.

Next, for a sufficiently large $d' = O(\log n)$, pick $2n$ vectors $\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_n \in \{0,1\}^{d'}$ such that, for $i, j \in \{1, 2, \dots, n\}$ we have

$$\langle \alpha_i, \beta_j \rangle = \begin{cases} 0 & \text{if } i = j \\ \geq 1 & \text{otherwise.} \end{cases}$$

This can be done, for instance, by picking each α_i to be a different vector with exactly half its entries 1, and picking β_i to be $\bar{\alpha}_i$ (i.e. the all 1s vector minus α_i). We thus can pick the smallest $d' = O(\log n)$ such that $\binom{d'}{d'/2} \geq n$.

We will now pick sets $V'_1, \dots, V'_k \subseteq \{0,1\}^{2d+d'}$ to give our instance of $\mathsf{DIRECTED\text{-}CYCLE}_{n,O(\log n),k}$. We will pick $V'_i = \{v'_{i,j}\}_{j \in \{1,2,\dots,n\}}$, where the $v'_{i,j}$ vectors are defined as follows: If $i \in \{1, 2, \dots, k-2\}$ then set $v'_{i,j} = v_{i,j} || v_{i,j} || 0^{d'}$. Then, set $v'_{k-1,j} = 0^d || v_{k-1,j} || \alpha_j$ and $v'_{k,j} = v_{k-1,j} || 0^d || \beta_j$.

We can see with these choices that the vectors $(v_{1,j_1}, v_{2,j_2}, \dots, v_{k-1,j_{k-1}})$ formed a $k-1$ cycle in the original instance of $\mathsf{DIRECTED\text{-}CYCLE}_{n,O(\log n),k-1}$ if and only if $(v'_{1,j_1}, v'_{2,j_2}, \dots, v'_{k-1,j_{k-1}}, v'_{k,j_{k-1}})$ form a k cycle in the new instance of $\mathsf{DIRECTED\text{-}CYCLE}_{n,O(\log n),k}$. Moreover, $\langle v'_{k-1,j}, v'_{k,j'} \rangle = 0$ if and only if $j = j'$, so these are the only possible k cycles in the new instance, as desired. $\square$

Corollary 3.1. *Suppose for any $k \geq 3$ that $\mathsf{DIRECTED\text{-}CYCLE}_{n,O(\log n),k}$ can be solved in (deterministic) time $N^{\tau+o(1)}$ for some constant $\tau \geq 0$. Then, for any constant $a > 0$, $\mathsf{MAX\text{-}2\text{-}SAT}_{n,an}$ can be solved in (deterministic) time $O(2^{(\tau/3+\delta)n})$ for every $\delta > 0$.*

4 Hypercliques in OV Hypergraphs and Directed Cycles

A straightforward generalization of the above argument shows:

Theorem 4.1. *Suppose, for integers $\ell > k \geq 2$, that $\mathsf{OV\text{-}HYPERGRAPH}_{N,O(\log N),\ell,k}$ can be solved in (deterministic) time $N^{\tau+o(1)}$ for some constant $\tau \geq 0$. Then, for any constant $a > 0$, $\mathsf{MAX\text{-}k\text{-}SAT}_{n,an}$ can be solved in (deterministic) time $O(2^{(\tau/\ell+\delta)n})$ for every $\delta > 0$.*

Following the reduction from finding hypercliques in hypergraphs to finding directed cycles of [LVW18], we can also reduce to finding directed cycles in OV graphs, with some care to ensure that the OV dimension does not increase too much:

Theorem 4.2. *Let $k \geq 4$ be a constant integer. There is an $\tilde{O}(d \cdot n^{k-\lceil k/3 \rceil})$ deterministic time reduction from $\mathsf{OV\text{-}HYPERGRAPH}_{n,d,k,3}$ to $\mathsf{DIRECTED\text{-}CYCLE}_{n^{k-\lceil k/3 \rceil},d+O(\log n),k}$.*

The proof of the theorem is a bit technical, so we first present the special case for $k = 4$ and then highlight some changes.

Theorem 4.3. *There is an $O(d \cdot n^2 \log n)$ deterministic time reduction from $\mathsf{OV\text{-}HYPERGRAPH}_{n,d,4,3}$ to $\mathsf{DIRECTED\text{-}CYCLE}_{\binom{n}{2},d+O(\log n),4}$.*

Proof. In $\mathsf{OV\text{-}HYPERGRAPH}_{n,d,4,3}$, we are given as input a size-n set $V = \{v_1, \dots, v_n\} \subseteq \{0,1\}^d$, and we want to determine whether there is a $T \subseteq V$ of size $|T| = 4$ such that, for all $S \subseteq T$ of size $|S| = 3$, we have $\mathsf{IP}(S) = 0$.

We first give some definitions. Define $AND : \{0,1\}^d \times \{0,1\}^d \to \{0,1\}^d$ by, for $x, y \in \{0,1\}^d$ and $\ell \in \{1, \ldots, d\}$, $AND(x,y)[\ell] = x[\ell] \cdot y[\ell]$. Next, similar to Lemma 3.3, for a sufficiently large $d' = O(\log n)$, pick $2n$ vectors $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n \in \{0,1\}^{d'}$ such that, for $i, j \in \{1, 2, \ldots, n\}$ we have

$$\langle \alpha_i, \beta_j \rangle = \begin{cases} 0 & \text{if } i = j \\ \geq 1 & \text{otherwise.} \end{cases}$$

As before, we can pick the smallest $d' = O(\log n)$ such that $\binom{d'}{d'/2} \geq n$. Finally, define $\chi : \{1,2,3,4\}^2 \times \{1,2,\ldots,n\}^2 \to \{0,1\}^{d'}$ by

$$\chi(p,q,i,j) = \begin{cases} \alpha_j & \text{if } p = q \\ \beta_i & \text{if } p = q+1 \pmod 4 \\ 0^{d'} & \text{otherwise.} \end{cases}$$

Our reduction constructs sets $V_1, V_2, V_3, V_4 \subseteq \{0,1\}^{d+4d'}$, each of size $\binom{n}{2}$, as follows. For each $p \in \{1,2,3\}$, and $1 \leq i < j \leq n$, put into set V_p the vector

$$v_{i,j,p} := AND(v_i, v_j) || \chi(p,1,i,j) || \chi(p,2,i,j) || \chi(p,3,i,j) || \chi(p,4,i,j),$$

where $||$ denotes vector concatenation. Finally, for each $1 \leq i < j \leq n$, put into V_4 the vector

$$v_{j,i,4} := AND(v_i, v_j) || \chi(4,1,j,i) || \chi(4,2,j,i) || \chi(4,3,j,i) || \chi(4,4,j,i).$$

We picked the function χ so that, for $a, b, c, d \in \{1, 2, \ldots, n\}$,

- for $p \in \{1,2,3\}$, we have $\langle v_{a,b,p}, v_{c,d,p+1} \rangle = 0$ if and only if $a < b = c < d$ and

$$\langle AND(v_{i_a}, v_{i_b}), AND(v_{i_b}, v_{i_d}) \rangle = \langle v_{i_a}, v_{i_b}, v_{i_d} \rangle = 0.$$

- For $p = 4$, we have $\langle v_{a,b,4}, v_{c,d,1} \rangle = 0$ if and only if $a > b = c < d$ and $\langle v_{i_a}, v_{i_b}, v_{i_d} \rangle = 0$.

In other words, there are $a, b, c, d \in \{1, 2, \ldots, n\}$ such that $\langle v_{a,b,1}, v_{b,c,2} \rangle = \langle v_{b,c,2}, v_{c,d,3} \rangle = \langle v_{c,d,3}, v_{d,a,4} \rangle = \langle v_{d,a,4}, v_{a,b,1} \rangle = 0$ if and only if $1 \leq a < b < c < d \leq n$ and, for every $S \subseteq \{v_{i_a}, v_{i_b}, v_{i_c}, v_{i_d}\}$ of size $|S| = 3$ we have $\mathsf{IP}(S) = 3$. In other words, the sets V_1, V_2, V_3, V_4 form the desired instance of $\mathsf{DIRECTED\text{-}CYCLE}_{\binom{n}{2}, d+4d', 4}$. □

Now we highlight the main changes to the above proof needed to obtain Theorem 4.2.

Sketch of the proof of Theorem 4.2. We start with $\mathsf{OV\text{-}HYPERGRAPH}_{n,d,k,3}$ where we are given as input a size-n set $V = \{v_1, \ldots, v_n\} \subseteq \{0,1\}^d$, and we want to determine whether there is a $T \subseteq V$ of size $|T| = k$ such that, for all $S \subseteq T$ of size $|S| = 3$, we have $\mathsf{IP}(S) = 0$.

Let $\gamma = k - \lceil k/3 \rceil$.

The reduction from k-hyperclique in 3-uniform hypergraphs to k-hypercycle in 3-uniform hypergraphs from [LVW18] carries over to our case and we see (details in the full version) that $\mathsf{OV\text{-}HYPERGRAPH}_{n,d,k,3}$ reduces to the following OV Hypercycle problem: given k sets of vectors $W_1, \ldots, W_k$ where $W_i \subseteq \{0,1\}^d$ with $|Q_i| = n$, are there $a_1 \in W_1, \ldots, a_k \in Q_k$ so that for every $j \in \{1, \ldots, k\}$, we have that $\langle a_j, a_{j+1}, \ldots, a_{j+\gamma} \rangle = 0$, where indices are taken mod k.

We now take this hypercycle problem with generalized inner product over $(\gamma+1)$-tuples and reduce it to k-cycle in a graph with roughly n^γ vertices extending our construction for 4-cycle.

We extend the definition of AND to take the bitwise AND of a γ-tuple of d length bit vectors. We then select for sufficiently high d', $2n^{\gamma-1}$ vectors $\alpha_1, \dots, \alpha_{n^{\gamma-1}}, \beta_1, \dots, \beta_{n^{\gamma-1}} \in \{0,1\}^{d'}$ such that, for $i, j \in \{1, 2, \dots, n\}^{\gamma-1}$ we have

$$\langle \alpha_i, \beta_j \rangle = \begin{cases} 0 & \text{if } i = j \\ \geq 1 & \text{otherwise.} \end{cases}$$

This can be done as before by picking each α_i to be a different vector with exactly half its entries 1, and picking β_i to be $\bar{\alpha}_i$. Then we get $d' = O(\log n)$ by picking the smallest d' with $\binom{d'}{d'/2} \geq n^{\gamma-1}$. We define $\chi : \{1, 2, \dots, k\}^2 \times \{1, 2, \dots, n\}^\gamma \to \{0,1\}^{d'}$ by the below, where $i_1, \dots, i_\gamma \in \{1, \dots, n\}$:

$$\chi(p, q, i_1, \dots, i_\gamma) = \begin{cases} \alpha_{i_2, \dots, i_\gamma} & \text{if } p = q \\ \beta_{i_1, \dots, i_{\gamma-1}} & \text{if } p = q + 1 \pmod{k} \\ 0^{d'} & \text{otherwise.} \end{cases}$$

Our reduction constructs sets $V_1, V_2, \dots, V_k \subseteq \{0,1\}^{d+kd'}$, each of size roughly n^γ, as follows. For each $p \in \{1, 2, \dots, k\}$, and each γ-tuple $i_1, \dots, i_\gamma \in \{1, \dots, n\}$ put into set V_p the vector

$$v_{i_1, \dots, i_\gamma, p} := AND(v^p_{i_1}, \dots, v^p_{i_\gamma}) || \chi(p, 1, v^p_{i_1}, \dots, v^p_{i_\gamma}) || \chi(p, 2, v^p_{i_1}, \dots, v^p_{i_\gamma}) || \dots || \chi(p, k, v^p_{i_1}, \dots, v^p_{i_\gamma}),$$

where $||$ denotes vector concatenation and v^p_j is the jth node of W_p from the hypercycle instance.

We picked the function χ so that,

$$\langle v_{i_1, \dots, i_\gamma, p}, v_{j_1, \dots, j_\gamma, p+1} \rangle = 0$$

if and only if $(v^p_{i_2}, \dots, v^p_{i_\gamma}) = (v^{p+1}_{i_1}, \dots, v^{p+1}_{i_{\gamma-1}})$ and

$$\langle AND(v^p_{i_1}, \dots, v^p_{i_\gamma}), (v^{p+1}_{j_1}, \dots, v^{p+1}_{j_\gamma}) \rangle = \langle v^p_1, v^p_2, \dots, v^p_\gamma, v^{p+1}_\gamma \rangle = 0.$$

In other words, there is an OV k-cycle if and only if there was an OV k-hypercycle. $\square$

We obtain the following corollary:

Corollary 4.1. *Suppose that there is some $\varepsilon > 0$ so that DIRECTED-CYCLE$_{n,O(\log n),k}$ can be solved in*

- *$O(n^{3/2-\varepsilon})$ time for some $k \geq 6$ divisible by 3, or*
- *$O(n^{3/2+1/(2\ell)-\varepsilon})$ time for some $k = 3\ell + 1 \geq 4$, or*
- *$O(n^{3/2+1/(4\ell+2)-\varepsilon})$ time for some $k = 3\ell + 2 \geq 5$.*

Then, for any constant $a > 0$, MAX-3-SAT$_{n,an}$ can be solved in (deterministic) time $O(2^{(1-\varepsilon/4+\delta)n})$ for every $\delta > 0$.

In particular, if DIRECTED-CYCLE$_{n,O(\log n),4}$ can be solved in time $n^{2-\varepsilon}$ time for some $\varepsilon > 0$, then MAX-3-SAT$_{n,an}$ has a faster algorithm.

5 Consequences for dynamic problems

In this section we will give two reductions that show that under Hypothesis 1.4, dynamic graph problems are hard even in OV graphs. We will need two tools to prove our theorems.

The first is from the previous section. It says that for any n we can construct $2n$ Boolean vectors of length $O(\log n)$, $\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_n$ so that $\langle \alpha_i, \beta_j \rangle = 0$ if and only if $i = j$.

The second tool is a way to make an OV graph 'layered'.

Claim 5.1. *For any $\ell \geq 2$, there exist ℓ nonzero Boolean vectors $a_1, \dots, a_\ell$ of length $2+(\ell-2)(\ell-1)/2$ so that for each i, j, $\langle a_i, a_j \rangle = 0$ if and only if $j = i+1$ or $i = j+1$.*

Proof. We proceed by induction. For $\ell = 2$, the vectors are $[1, 0]$ and $[0, 1]$. Suppose the vectors for ℓ are $a_1, \dots, a_\ell$ and have length L, then for every $j < \ell$, replace a_j with a_j concatenated with the length $\ell - 1$ bit vector that is all 0 except in position j in which it is 1. Replace a_ℓ with a_ℓ concatenated with the length $\ell - 1$ all 0s vector. Finally set $a_{\ell+1}$ to be the length L all 0s vector concatenated with the length $\ell - 1$ all 1s vector.

After the vector replacement, $\langle a_j, a_{\ell+1} \rangle > 0$ for all $j < \ell$, and $\langle a_i, a_j \rangle$ is the same as before the replacement for $i, j \leq \ell$, and $\langle a_\ell, a_{\ell+1} \rangle = 0$. The length of the vectors goes up by $\ell - 1$. If we assume inductively that $L = 2 + (\ell - 2)(\ell - 1)/2$, adding $\ell - 1$, we complete the proof as $(\ell - 2)(\ell - 1)/2 + (\ell - 1) = (\ell - 1)\ell/2$. □

Remark 5.1. In fact, we will see in Theorem 6.1 below that the bound $2 + (\ell - 2)(\ell - 1)/2$ in Claim 5.1 can be replaced by only $O(\log \ell)$. However, such an improvement is unimportant for the results in this section, in which we will always pick ℓ to be a constant.

First let us consider the dynamic s-t OV Shortest Paths which asks to maintain the distance between two fixed vertices s and t in an n node OV graph with dimension $O(\log n)$ under vertex relabel updates (change the vector representing a node).

Theorem 5.1. *Under Hypothesis 1.4, dynamic s-t OV Shortest Paths with vertex label updates that change at most one edge at a time requires either $n^{\omega-o(1)}$ preprocessing time, or $n^{\omega-1-o(1)}$ amortized update time. If arbitrary vertex label updates are supported, then $5/3 - \varepsilon$-approximating the s-t distance requires $n^{\omega-1-o(1)}$ amortized update time even when starting with an empty graph (all vectors are all 1s).*

Proof. Suppose we are given an instance of $\mathsf{OV}\Delta$ with n vectors V of dimension $d = O(\log n)$.

We will start with the proof for vertex label updates that change at most a single edge.

Let us create 4 sets of vectors V_1, V_2, V_3, V_4, using the vectors $a_1, a_2, \dots, a_6$ from Claim 5.1: for each $i \in \{1, \dots, 4\}$ and for every $v \in V$, put in V_j the vector v^j that concatenates v with a_{j+1}. In addition, create two new vectors, s and t, where s is the d-length all 1s vector concatenated with a_1 and t is the d-length all 1s vector concatenated with a_6.

By construction, s and t are non-orthogonal with all other vectors, and a vector $v^j \in V_j$ and a vector $u^k \in V_k$ for $j \leq k$ are orthogonal if and only if $k = j+1$ and u and v are orthogonal in the original $\mathsf{OV}\Delta$ instance.

We will use the length $d' = O(\log n)$ vectors $\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_n$ from our first tool. For every vector $v^j \in V^j$ for $j \in \{2, 3\}$, we concatenate the length d' all 0s vector to its end. For a vector $v_q^j \in V^j$ for $j \in \{1, 4\}$ (where v_q is the qth vector of V) we concatenate β_q to the end of it. This does not change the orthogonality of the vectors. We also concatenate the length d' all zeros vector to the ends of both s and t.

This completes the preprocessing stage.

Now, the updates are as follows. We have n stages, one for each vector $v_q \in V$. In the stage for v_q, we replace the last d' bits of s and t with α_q and the first d bits with the all 0 vector. This has the effect of inserting the edges (s, v_q^1) and (v_q^4, t) and leaving all other edges unchanged.

Then, the distance between s and t in the new OV graph is 5 if and only if there are vectors $a, b \in V$ such that $\langle v_q, a\rangle = \langle a, b\rangle = \langle b, v_q\rangle = 0$. At the end of the stage for v_q, we undo the relabeling of s and t and we move on to the next vector in V. This effectively deletes the two edges that were inserted.

The number of stages is n, and so if the preprocessing time is $O(n^{\omega-\varepsilon})$ for $\varepsilon > 0$, then the total time of all n update stages needs to be $n^{\omega-o(1)}$ under our Hypothesis, and hence the amortized time per update is $n^{\omega-1-o(1)}$.

Now we present a simpler proof where the updates are arbitrary relabelings. In the previous proof since the edges in the initial graph would be too expensive to insert one by one using single edge updates, we needed preprocessing. Here there will be no need for preprocessing since we can simply label all vertices as needed one by one.

Create just two sets of vectors V_1, V_2, using the vectors a_1, a_2, a_3, a_4 from Claim 5.1: for each $i \in \{1, 2\}$ and for every $v \in V$, put in V_j the vector v^j that concatenates v with a_{j+1}. In addition, create two new vectors, s and t, where s is the d-length all 1s vector concatenated with a_1 and t is the d-length all 1s vector concatenated with a_4. Then we will have n stages one for each vector $v_q \in V$ in which we will change s and t as follows. In the stage for v_q, we replace the first d bits of s and t with v_q. Now, s will only be orthogonal to the vectors in V_1 corresponding to original vectors that are orthogonal to v_q and t will be orthogonal to the vectors in V_2 corresponding to original vectors that are orthogonal to v_q. Thus the distance between s and t is 3 if there is a triangle in the OV graph, and it is at least 5 otherwise. □

The second dynamic problem we will prove hardness for is dynamic bipartite perfect matching in OV graphs: given an OV graph maintain whether it has a perfect matching, under vertex relabel updates.

Theorem 5.2. *Dynamic bipartite perfect matching in OV graphs on n nodes and dimension $O(\log n)$ under vertex relabel updates that change only a single edge at a time requires either $n^{\omega-o(1)}$ preprocessing time, or $n^{\omega-1-o(1)}$ amortized update time. If arbitrary vertex relabelings are supported, the same lower bound holds but with arbitrary preprocessing.*

Proof. The proof is similar to the proof of the previous theorem.

We start with an instance of $\mathsf{OV}\Delta$, and make it layered with 10 layers, $s, V^{1,1}, V^{1,2}, V^{2,1}, V^{2,2}, V^{3,1}, V^{3,2}, V^{4,1}, V^{4,2}, t$. Here s and t are vertices and the rest contain vectors corresponding to the vectors in V. The layering is accomplished using Claim 5.1 as in the previous theorem. This adds a constant number of coordinates to the vectors.

For a particular j, the edges between $V^{j,1}$ and $V^{j,2}$ are just a matching between the vectors $v_q^{j,1}$ and $v_q^{j,2}$ corresponding to a vector $v_q \in V$. This is accomplished using the vectors α_i and β_i from our first tool.

The edges between $V^{j,2}$ and $V^{j+1,1}$ are between vectors $v_q^{j,2}$ and $v_r^{j+1,1}$ that correspond to $v_q, v_r \in V$ that are orthogonal.

The above two tasks are accomplished as follows. Let $L = O(1)$ be the length of the vectors a_j for $j \in \{1, \ldots, 12\}$ from Claim 5.1 . Let $d = O(\log n)$ be the length of the vectors from the original $\mathsf{OV}\Delta$ instance and let $d' = O(\log n)$ be the length of the vectors $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n$ from our first tool. Let 0_t denote the length-t all 0s vector. Each of the vectors $v_q^{j,b}$ for $j \in \{1,2,3,4\}, b \in \{1,2\}$ is of length $L + 2d' + 2d$ as follows. If j is odd, a vector $v_q^{j,1}$ is the concatenation of a_{2j+2}, $0_{d'}$, α_q, v_q and 0_d, and a vector $v_q^{j,2}$ is the concatenation of a_{2j+3}, $0_{d'}$, β_q, 0_d and v_q. On the other hand, if j is even, a vector $v_q^{j,1}$ is the concatenation of a_{2j+2}, α_q,$0_{d'}$, 0_d and v_q, and a vector $v_q^{j,2}$ is the concatenation of a_{2j+3}, β_q, $0_{d'}$, v_q and 0_d.

We make s and t orthogonal to each other but not to anything else. This can be done by letting s be a_1 followed by $0_{2d'+2d}$ and t be a_2 followed by $0_{2d'+2d}$. Since all other vectors start with a_k for $k \geq 4$, s and t are not orthogonal to any of the other vectors but are orthogonal to each other.

This completes the preprocessing stage. At this point there is a perfect matching consisting of (s,t) and the perfect matchings between $V^{j,1}$ and $V^{j,2}$ for $j \in \{1,2,3,4\}$.

There is a stage for each $v_q \in V$. In the stage for v_q, we make s be in layer 1 and only orthogonal to $v_q^{1,1}$, and t be in layer 10 and only orthogonal to $v_q^{4,2}$. This effectively deletes the edge (s,t) and inserts edges $(s, v_q^{1,1})$ and $(v_q^{4,2}, t)$. (These updates are undone at the end of the stage.)

Since $v_q^{1,1}$ is the concatenation of $a_4, \alpha_q, 0_{d'}, 0_d, v_q$ and $v_q^{4,2}$ is the concatenation of $a_{11}, \beta_q, 0_{d'}, v_q, 0_d$, we only need to set s to be the concatenation of $a_3, \beta_q, 0_{d'}, 0_d, 0_d$ and t to be the concatenation of $a_{12}, \alpha_q, 0_{d'}, 0_d, 0_d$.

Now the only way that for there to be a perfect matching is if there are $v_a, v_b \in V$ so that v_q, v_a, v_b form a triangle in the original OV graph. The number of updates is $O(n)$, and hence we get the same conditional lower bound as in the previous theorem. □

6 Problems still NP-hard on OV graphs

We begin by recalling Alon's original proof that graphs with low maximum degree can be written as OV graphs of low dimension:

Lemma 6.1 ([Alo86, Lemma 3.2]). *For any graph G with n nodes and maximum degree d, there are n vectors in $\{0,1\}^{O(d^2 \log n)}$ whose OV graph is G.*

Proof. (We give here the original proof of [Alo86].) Let $k = \lceil 2e^2(d+1)^2 \ln n \rceil$. We begin by picking n vectors $v_1, \ldots, v_n \in \{0,1\}^k$ at random: for each $i \in \{1,2,\ldots,n\}$ and each $\ell \in \{1,2,\ldots,k\}$ we set $v_i[\ell] = 1$ independently with probability $1/(d+1)$, and $v_i[\ell] = 0$ otherwise. Next, we perform a 'correction': for each $i,j \in \{1,2,\ldots,n\}$ which are adjacent nodes in G, and each $\ell \in \{1,2,\ldots,k\}$ such that $v_i[\ell] = v_j[\ell] = 1$, we set both $v_i[\ell]$ and $v_j[\ell]$ to 0. After this correction, every edge in G is also an edge in the OV graph for $V = \{v_1, \ldots, v_n\}$.

Now, consider any $i,j \in \{1,2,\ldots,n\}$ which are not adjacent in G. They are also not adjacent in the OV graph of V so long as there is an $\ell \in \{1,2,\ldots,k\}$ such that (1) $v_i[\ell] = v_j[\ell] = 1$ before the correction phase, and (2) for every $i' \in \{1,2,\ldots,n\} \setminus \{i,j\}$ which is adjacent to i or j (there are at most $2d$ such i's), $v_{i'}[\ell] = 0$ before the correction phase. For a fixed $\ell \in \{1,2,\ldots,k\}$, this happens with probability

$$\frac{1}{(d+1)^2}\left(1 - \frac{1}{d+1}\right)^{2d} \geq \frac{1}{e^2(d+1)^2}.$$

Thus, for a fixed pair $i,j \in \{1,2,\ldots,n\}$ which are not adjacent in G, the probability that they are adjacent in the OV graph of V is at most

$$\left(1 - \frac{1}{e^2(d+1)^2}\right)^k \leq e^{-k/e^2(d+1)^2} \leq \frac{1}{n^2}.$$

It follows that the expected number of $i, j \in \{1, 2, \ldots, n\}$ which are not adjacent in G but are adjacent in the OV graph of V is at most $\binom{n}{2} \cdot n^{-2} < 1/2$. Hence, by the probabilistic method, there is a choice of randomness for which there are no such i, j, and hence the OV graph of V is exactly our graph G, as desired. $\square$

Theorem 6.1. *Given a graph G with n nodes and maximum degree d, there is a deterministic algorithm running in time $n^{O(d^2)}$ to find n vectors in $\{0,1\}^{O(d^2 \log n)}$ whose OV graph is G.*

Proof. We will derandomize the construction from Lemma 6.1. We first note that, for a fixed ℓ, when picking $v_i[\ell]$ for all i before the correction phase, it is sufficient to pick them $(2d+2)$-wise independently, rather than fully independently as we did in the original proof. Indeed, in the proof that any $i, j \in \{1, 2, \ldots, n\}$ which are not adjacent in G are also not adjacent in the OV graph of V, we only needed to consider $v_{i'}[\ell]$ for $(2d+2)$ choices of i'. For a fixed ℓ, we can thus draw the required $v_i[\ell]$ before the correction phase for all i, $(2d+2)$-wise independently, to be 1 with probability $1/(d+1)$, and 0 otherwise, using $O(d \log n)$ random bits via standard constructions.

Next, rather than independently sample such a vector for each ℓ, we use the standard derandomization trick of using a random walk on a constant-degree expander graph (see e.g. [V$^+$12, Section 4.2]). We saw above that for a given $i, j \in \{1, 2, \ldots, n\}$ which are not adjacent in G, the required vector entries for a particular ℓ can be drawn with $O(d \log n)$ random bits, and will cause them to be not adjacent in the OV graph of V with probability $1/O(d^2)$. Hence, using an expander random walk, we can draw the vector entries for $t = O(d^2 \log n)$ different ℓs using $O(t + d \log n) = O(d^2 \log n)$ random bits. As before, there is a choice of randomness for which the resulting OV graph is G. We can iterate over all $2^{O(d^2 \log n)} = n^{O(d^2)}$ choices of randomness to find such an OV graph as desired. $\square$

This immediately gives a polynomial-time reduction from algorithmic problems on sparse graphs to algorithmic problems on OV graphs of dimension $O(\log n)$. For instance:

Corollary 6.1. *Any problem which is NP-hard on graphs of constant maximum degree is also NP-hard on OV graphs of dimension $O(\log n)$.*

Finally, we note that some NP-hard graph problems do become easier on OV graphs:

Theorem 6.2. *Given as input vectors $V = \{v_1, \ldots, v_n\} \subseteq \{0,1\}^d$ defining a dimension d OV graph G_V, a maximum size clique in G_V can be found in time $2^d \cdot n^{O(1)}$.*

Proof. A clique in G_V of size s corresponds to a subset $S \subseteq V$ of size $|S| = s$ such that, for each $i \in \{1, 2, \ldots, d\}$, there is at most one $v \in V$ such that $v[i] = 1$. Equivalently, if we view each $v \in V$ as a subset of $U := \{1, 2, \ldots, d\}$ (i.e. $i \in v$ if and only if $v[i] = 1$), then $S \subseteq V$ is a clique if and only if it is a collection of disjoint subsets of U. Finding the maximum size clique is hence the set packing problem in a universe of size d, which can be solved in $2^d \cdot n^{O(1)}$ time [BHK09]. $\square$

7 OV Matrices and Online Matrix-Vector Multiplication

Theorem 7.1. *For any real numbers a, c with $c > 2a > 0$, and any matrix $M \in \mathbb{F}_2^{n \times n}$ of OV dimension $c \cdot \log n$, we can write $M = L \cdot R + S$ where $L, R^T \in \mathbb{F}_2^{n \times r}$ for $r = n^{a \log(c/a) + o(1)}$, and $S \in \mathbb{F}_2^{n \times n}$ has at most $O(n^{2-a})$ entries equal to 1. Moreover, given the vectors $u_1, \ldots, u_n, v_1, \ldots, v_n \in \{0,1\}^{c \log n}$ such that $M[i,j] = 1$ if and only if $\langle u_i, v_j \rangle = 0$ (over $\mathbb{Z}$), we can compute L, R and S in randomized time $\tilde{O}(n^2)$ with high probability.*

Proof. We use the polynomial method similar to [AWY15, Theorem 1.1]. Let $d = c\log n$. For every subset $S \subseteq \{1, 2, \ldots, d\}$, define the polynomial $p_S : \mathbb{F}_2^d \to \mathbb{F}_2$ by $p_S(x) = \sum_{\ell \in S} x[\ell]$. Notice that, when $x = 0^d$, then $p_S(x) = 0$ for all S, and when $x \neq 0^d$, then $p_S(x) = 1$ for half of all S.

Let $m = a\log n$. We begin by picking m independent, uniformly random subsets $S_1, \ldots, S_m \subseteq \{1, 2, \ldots, d\}$. We then compute the set E of all pairs $(i, j) \in \{1, \ldots, n\}^2$ such that $AND(u_i, v_j) \neq 0^d$ but $p_{S_t}(AND(u_i, v_j)) = 0$ for all $t \in \{1, \ldots, m\}$. This can be computed in $O(n^2 md) = \tilde{O}(n^2)$ time. From the above discussion, a given (i, j) will be in E with probability $2^{-m} = n^{-a}$, and so the expected size of E is n^{2-a}. Hence, by Markov's inequality, $|E| \leq 100 \cdot n^{2-a}$ with probability at least 0.99; if this is not the case, repeat independently until it is.

Now, consider the polynomial $p : \mathbb{F}_2^d \to \mathbb{F}_2$ given by, for $z \in \mathbb{F}_2^d$, $p(z) = 1 + \prod_{t=1}^{m}(1 + p_{S_t}(z))$. Since p is over $\mathbb{F}_2$, with $x^2 = x$ for all $x \in \mathbb{F}_2$, we can assume that p is *multilinear* by reducing any exponent larger than 1 to 1. Hence, we can expand p into a sum of multilinear monomials, and the number r of monomials will be at most[1]

$$r \leq \sum_{g=0}^{m} \binom{d}{g} \leq O\left(\frac{d}{m}\right)^m = O\left(\frac{c\log n}{a\log n}\right)^{a\log n} \leq n^{a\log(c/a)+o(1)}.$$

In particular, since for $x, y \in \mathbb{F}_2^d$ the vector $AND(x, y)$ consists of a single monomial in terms of x and y in each entry, i.e. $AND(x, y)[\ell] = x[\ell] \cdot y[\ell]$, we can write

$$p(AND(x, y)) = \sum_{w=1}^{r} \prod_{\ell \in T_w} x[\ell] \cdot y[\ell]$$

for some subsets $T_1, \ldots, T_r \subseteq \{1, 2, \ldots, d\}$. Define the map $V : \mathbb{F}_2^d \to \mathbb{F}_2^r$ by $V(x)[w] = \prod_{\ell \in T_w} x[\ell]$. Hence, $p(AND(x, y)) = \langle V(x), V(y) \rangle$ (where the inner product is over $\mathbb{F}_2$).

Finally, we can define our matrices L, R, S as follows: $L \in \mathbb{F}_2^{n \times r}$ is the matrix whose ith row is $V(u_i)$, $R \in \mathbb{F}_2^{r \times n}$ is the matrix whose jth column is $V(v_j)$, and $S \in \mathbb{F}_2^{n \times n}$ is the matrix whose entry $S[i, j]$ is 1 if and only if $(i, j) \in E$. Hence, we can see that $(L \cdot R)[i, j] = p(AND(u_i, v_j))$, and we have defined E so that $S[i, j] = 1$ if and only if $p(AND(u_i, v_j)) \neq M[i, j]$. □

Corollary 7.1. *For $c > 0$, given vectors $u_1, \ldots, u_n, v_1, \ldots, v_n \in \{0, 1\}^{c\log n}$ which define a matrix $M \in \mathbb{F}_2^{n \times n}$ of OV dimension $c\log n$, we can compute matrices $L \in \mathbb{F}_2^{n \times n^{0.1}}$, $R \in \mathbb{F}_2^{n^{0.1} \times n}$ and $S \in \mathbb{F}_2^{n \times n}$ in time $\tilde{O}(n^2)$ with high probability such that S has at most $n^{2-1/O(\log c)}$ nonzero entries and $M = L \cdot R + S$.*

Proof. Apply Theorem 7.1 with $a = 1/O(\log c)$ so that $a\log(c/a) < 0.1$. □

Corollary 7.2. *For $c > 0$, we can preprocess vectors $u_1, \ldots, u_n, v_1, \ldots, v_n \in \{0, 1\}^{c\log n}$ which define a matrix $M \in \mathbb{F}_2^{n \times n}$ of OV dimension $c\log n$, in preprocessing time $\tilde{O}(n^2)$ with high probability, such that given as input a vector $v \in \mathbb{F}_2^n$, we can compute the product $M \cdot v$ in time $n^{2-1/O(\log c)}$.*

Proof. In preprocessing we compute the matrices L, R, S from Corollary 7.1. Then, on input v, we compute $L \cdot (R \cdot v)$ in $O(n^{1.2})$ time and $S \cdot v$ in $n^{2-1/O(\log c)}$ time, and sum the results. □

[1]Here we use the fact, from Stirling's inequality, that for any $1 \leq k \leq n$ we have $\binom{n}{k} \leq \frac{n^k}{k!} \leq \left(\frac{e \cdot n}{k}\right)^k \leq O(n/k)^k$.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments on an earlier version.

References

[ACW16] Josh Alman, Timothy M Chan, and Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 467–476. IEEE, 2016.

[Alo86] Noga Alon. Covering graphs by the minimum number of equivalence relations. *Combinatorica*, 6(3):201–206, 1986.

[AV14] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443, 2014.

[AW17] Josh Alman and Ryan Williams. Probabilistic rank and matrix rigidity. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 641–652. ACM, 2017.

[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. Society for Industrial and Applied Mathematics, 2015.

[AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[BHK09] Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM Journal on Computing*, 39(2):546–563, 2009.

[CPP16] Marek Cygan, Marcin Pilipczuk, and Michał Pilipczuk. Known algorithms for edge clique cover are probably optimal. *SIAM Journal on Computing*, 45(1):67–83, 2016.

[CS15] Ruiwen Chen and Rahul Santhanam. Improved algorithms for sparse MAX-SAT and max-k-csp. In *Theory and Applications of Satisfiability Testing - SAT 2015 - 18th International Conference, Austin, TX, USA, September 24-27, 2015, Proceedings*, pages 33–45, 2015.

[CW19] Lijie Chen and Ryan Williams. An equivalence class for orthogonal vectors. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 21–40. SIAM, 2019.

[DW06] Evgeny Dantsin and Alexander Wolpert. MAX-SAT for formulas with constant clause density can be solved faster than in $\mathrm{o}(\mathrm{s}^2)$ time. In *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*, pages 266–276, 2006.

[EGP66] Paul Erdös, Adolph W Goodman, and Louis Pósa. The representation of a graph by set intersections. *Canadian Journal of Mathematics*, 18:106–112, 1966.

[Gal14] François Le Gall. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation, ISSAC 2014, Kobe, Japan, July 23-25, 2014*, pages 296–303, 2014.

[GGHN09] Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Data reduction and exact algorithms for clique cover. *Journal of Experimental Algorithmics (JEA)*, 13:2, 2009.

[Gyá90] András Gyárfás. A simple lower bound on edge coverings by cliques. *Discrete Mathematics*, 85(1):103–104, 1990.

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[Lov79] László Lovász. On the shannon capacity of a graph. *IEEE Transactions on Information theory*, 25(1):1–7, 1979.

[Lov19] László Lovász. *Graphs and geometry.* American Mathematical Society, 2019.

[LSS89] László Lovász, Michael Saks, and Alexander Schrijver. Orthogonal representations and connectivity of graphs. *Linear Algebra and its applications*, 114:439–454, 1989.

[LVW18] Andrea Lincoln, Virginia Vassilevska Williams, and Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1236–1252. SIAM, 2018.

[LW17] Kasper Green Larsen and Ryan Williams. Faster online matrix-vector multiplication. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2182–2189. Society for Industrial and Applied Mathematics, 2017.

[MPR95] Sylvia D Monson, Norman J Pullman, and Rolf Rees. A survey of clique and biclique coverings and factorizations of (0, 1)-matrices. *Bull. Inst. Combin. Appl*, 14:17–86, 1995.

[MQ06] TS Michael and Thomas Quint. Sphericity, cubicity, and edge clique covers of graphs. *Discrete Applied Mathematics*, 154(8):1309–1313, 2006.

[V^+12] Salil P Vadhan et al. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.

[Vas12] Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 887–898, 2012.

[Vas18] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the ICM*, 2018.

[Wil05] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005.

[Wil07] Ryan Williams. *Algorithms and resource requirements for fundamental problems.* PhD thesis, Ph. D. Thesis, Carnegie Mellon University, CMU-CS-07-147, 2007.

[YZ97] Raphael Yuster and Uri Zwick. Finding even cycles even faster. *SIAM J. Discrete Math.*, 10(2):209–222, 1997.