Electronic Colloquium on Computational Complexity, Report No. 180 (2023)

New Graph Decompositions and Combinatorial Boolean Matrix Multiplication Algorithms

Amir Abboud[*] Nick Fischer[†] Zander Kelley[‡] Shachar Lovett[§] Raghu Meka[¶]

Abstract

We revisit the fundamental Boolean Matrix Multiplication (BMM) problem. With the invention of algebraic fast matrix multiplication over 50 years ago, it also became known that BMM can be solved in truly subcubic $O(n^{\omega})$ time, where $\omega < 3$; much work has gone into bringing ω closer to 2. Since then, a parallel line of work has sought comparably fast *combinatorial* algorithms but with limited success. The naïve $O(n^3)$-time algorithm was initially improved by a $\log^2 n$ factor [Arlazarov *et al.*; RAS'70], then by $\log^{2.25} n$ [Bansal and Williams; FOCS'09], then by $\log^3 n$ [Chan; SODA'15], and finally by $\log^4 n$ [Yu; ICALP'15].

We design a combinatorial algorithm for BMM running in time $n^3/2^{\Omega(\sqrt[7]{\log n})}$—a speed-up over cubic time that is stronger than any poly-log factor. This comes tantalizingly close to refuting the conjecture from the 90s that truly subcubic combinatorial algorithms for BMM are impossible. This popular conjecture is the basis for dozens of fine-grained hardness results.

Our main technical contribution is a new *regularity decomposition* theorem for Boolean matrices (or equivalently, bipartite graphs) under a notion of regularity that was recently introduced and analyzed analytically in the context of communication complexity [Kelley, Lovett, Meka; arXiv'23], and is related to a similar notion from the recent work on 3-term arithmetic progression free sets [Kelley, Meka; FOCS'23].

[*]Weizmann Institute of Science. This work is part of the project CONJEXITY that has received funding from the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme (grant agreement No. 101078482). Supported by an Alon scholarship and a research grant from the Center for New Scientists at the Weizmann Institute of Science. Email: `amir.abboud@weizmann.ac.il`

[†]Weizmann Institute of Science. Supported by the project CONJEXITY as above. Email: `nick.fischer@weizmann.ac.il`

[‡]Department of Computer Science, University of Illinois at Urbana-Champaign. Supported by NSF CAREER award 2047310. Email: `awk2@illinois.edu`

[§]Department of Computer Science and Engineering, University of California, San Diego. Supported by NSF DMS award 1953928, NSF CCF award 2006443, and a Simons investigator award. Email: `slovett@ucsd.edu`

[¶]Department of Computer Science, University of California, Los Angeles. Supported by NSF AF 2007682 and NSF Collaborative Research Award 2217033. Email: `raghum@cs.ucla.edu`

ISSN 1433-8092

1 Introduction

Boolean Matrix Multiplication (BMM) is one of the most basic and fundamental combinatorial problems. It can be solved in $O(n^\omega)$ time, where $2 \le \omega < 2.3716$ [33, 74] is the exponent of (integer) matrix multiplication. The algebraic technique underlying Strassen's [66] and all subsequent "fast matrix multiplication" algorithms have several limitations (discussed in Section 1.1) related to *generalizability*, *elegance*, and *practical efficiency*. Therefore, in addition to the line of work trying to enhance this algebraic machinery aiming to reach $\omega = 2$, a parallel line of work aims to match (or improve) these subcubic bounds with different combinatorial techniques.

The first result in this direction is the "Four-Russians" algorithm by Arlazarov, Dinic, Kronrod, and Faradžev [12] that achieves $o(n^3)$ complexity by precomputing the answers to small subinstances. This approach can give $O(n^3/\log^2 n)$ time, but nothing faster [11]. After many decades, Bansal and Williams [13] used *regularity lemmas* to gain an additional $\log^{0.25} n$ factor speed-up. The usefulness of graph regularity techniques was undermined a few years later when Chan [23] gave a better, $O(n^3/\log^3 n)$ bound only using simple divide-and-conquer. Yu [75] optimized the divide-and-conquer method to achieve an $O(n^3/\log^4 n)$ bound that stood since 2015.[1]

A pessimistic conjecture that has been popular since the 90s [65, 56] states that truly subcubic running times are impossible for combinatorial algorithms; that is, one may be able to shave some more logarithmic factors, but we cannot reach $O(n^{3-\epsilon})$ time with $\epsilon > 0$.

Conjecture 1.1 (Combinatorial BMM)**.** *There is no combinatorial algorithm for BMM running in time $O(n^{3-\epsilon})$, for any $\epsilon > 0$.*

This conjecture has served as the basis for many conditional lower bounds. Refuting it would reopen the quest for polynomially faster combinatorial algorithms for a host of fundamental discrete problems [64, 72, 7, 8, 27, 24, 22, 25, 20, 58, 21, 30, 1, 19, 31, 5, 18, 16, 50, 49][2].

Main Result. In this paper, we prove a new regularity decomposition theorem that leads to a *quasi-polynomial*[3] saving for BMM, combinatorially, coming much closer than before to refuting Conjecture 1.1.

Theorem 1.2 (Combinatorial BMM)**.** *There is a deterministic combinatorial algorithm computing the Boolean product of two $n \times n$ matrices in time $n^3/2^{\Omega(\sqrt[7]{\log n})}$.*

We immediately get a similar combinatorial *super-poly-logarithmic* saving for many other problems (that can be reduced to BMM). The list includes central problems in their domains such as *context-free grammar parsing* from formal languages [68], computing the *transitive closure* of a directed graph [35, 60], *join-project queries* from databases [10, 69], parameterized problems such as *k-clique* [61] and *k-dominating-set* [34], and various matrix product problems [59, 29, 70].

Our (and most previous) results are obtained by designing algorithms for the simpler *triangle detection* problem and then using a well-known subcubic equivalence with BMM [72]. Therefore, we focus on triangle detection below.

[1] All running times in this paragraph are up to log-log-factors.

[2] Some of these references assume the stronger Combinatorial k-Clique Conjecture, that would also be refuted if Conjecture 1.1 is false.

[3] Recall that bounds of the form $2^{(\log n)^c}$ are called quasi-polynomial. In our case, $c < 1$.

A New Regularity Decomposition Theorem. In abstract terms, a graph is *regular* if it behaves somewhat pseudo-randomly, and a *regularity decomposition* theorem states that any graph can be decomposed into a "small" number of regular subgraphs. Such results are interesting mathematically because they say that any graph can be simplified dramatically, and also algorithmically because they let us reduce a problem from arbitrary to (the often easier) random-like graphs. The possibility and efficiency of such results depend on the precise notion of regularity; generally, there is a trade-off between strength (i.e. how close "regular" is to random) and efficiency (i.e. the number of subgraphs in the decomposition).

For example, the celebrated Szemerédi's Regularity Lemma [67] yields a decomposition into subgraphs with very strong pseudo-random properties, but to achieve meaningful results for graphs with density δ, the number of parts inherently scales as a tower function of height poly$(1/\delta)$ [42].[4] A comparable but weaker notion of regularity due to Frieze and Kannan [38] admits decompositions into fewer, but still exponentially many parts (specifically, $2^{O(\delta^{-2})}$). At the other end of the spectrum, *expander decompositions* use a much weaker notion of pseudo-randomness, but significantly gain in efficiency.

Considering a specific problem, e.g. triangle detection, the challenge is finding a sweet spot in which the regularity notion is strong enough to make the problem algorithmically easy, yet weak enough to make the decomposition efficient. Unfortunately, expanders are too weak [6]. Based on Szemerédi-regularity and Frieze-Kannan-regularity, Bansal and Williams [13] indeed scored non-trivial algorithmic improvements for Boolean Matrix Multiplication: A $(\log^*(n))^{\Omega(1)}$-shave based on Szemerédi's Regularity Lemma,[5] and a $(\log n)^{1/4}$-shave based on Frieze-Kannan regularity. In both cases, however, due to the excessive number of pieces in the regularity decomposition, it seems hopeless to go beyond log-shaves.

In this paper we employ a regularity notion called *grid regularity* that was recently introduced by Kelley, Lovett and Meka [52] in the context of communication complexity and is based on similar results in the work of Kelley and Meka [53] on 3-term arithmetic progressions. This regularity notion is weak but still useful for triangle detection and BMM. Phrased in terms of matrices, the key takeaway from their work is that a single grid-regular matrix is not necessarily very random, but the *product of two grid-regular matrices is very random* (see Theorem 2.1). Equivalently, in a 3-layered graph in which both edge sets are grid regular, the number of 2-paths from left to right behaves randomly. Our main contribution is that we (1) establish a decomposition theorem for this notion of regularity into *quasi-polynomially* many parts (specifically, $2^{O((\log \delta^{-1})^7)}$), and (2) provide an efficient deterministic algorithm to compute this decomposition.

1.1 On Combinatorial Algorithms

Despite the large number of papers on combinatorial algorithms for BMM, there is currently no satisfying and precise definition for this notion. The main reason for this, we believe, is that there are multiple strong motivations for seeking combinatorial algorithms (discussed below) that are not necessarily consistent with one another. A simple algorithm need not be practical, or vice versa, and an algorithm that generalizes to one setting may not generalize to another. Therefore, one can either focus on precise definitions that are limited to one motivation or embrace a more inclusive but loose definition. Many examples of the former approach exist (see below), and they

[4] I.e., $f(\delta) \leq 2^{2^{2^{\cdots}}}$ where the tower has height poly(δ^{-1}).

[5] In fact, their algorithm is based on the Triangle Removal Lemma which can be proven via Szemerédi's Regularity Lemma, but could in principle (and, in fact, does) admit better quantitative bounds [36].

give rise to interesting research questions. The latter, however, is more popular in the community: we currently have no truly subcubic algorithm for BMM other than Strassen's algorithm (and its successors) so we should first seek to break Conjecture 1.1 by *any other technique* and hope that *at least one* of the motivations gets satisfied.

To help shed light on this, let us review the limitations of the existing algebraic technique that motivate us to seek other algorithms. The first two may be more well-known, but the third is more pressing for fine-grained complexity and algorithm design. Along the way, we discuss to what extent our algorithm satisfies each consideration.

- **Simplicity:** Strassen's algorithm (and even more so for its successors) exploits cancellations using formulas that may be considered unintuitive; consequently, the values manipulated at intermediate stages of the computation are quite uninterpretable. Instead, one may hope for techniques that are simpler and more interpretable; this is probably the historical reason for the name "combinatorial". Some works have proposed precise definitions along these lines, e.g. for solving triangle detection an algorithm can only generate sets by basic operations on the neighborhoods of nodes, and strong lower bounds exist [11, 32]. Unfortunately, the current definitions are not flexible enough to capture the regularity decomposition techniques of Bansal-Williams [13] and of our paper, nor even the simple divide-and-conquer of Chan [23] and Yu [75], even though these techniques are widely-considered "combinatorial" (even by the authors proposing the definitions). In particular, our algorithm decomposes the graph by repeatedly finding and removing irregular pieces (i.e., that contain too many bicliques) and then uses a brute-force algorithm on the (sparse) parts.

- **Practical Efficiency**: Here, one should make a distinction between Strassen's $n^{2.81}$ algorithm, and its successors that have reduced ω much further. The latter algorithms are considered "galactic" and the interest in them is mostly theoretical. On the other hand, Strassen's algorithm has been used in practice, but the gains are limited; some reasons include its bad locality (generating too many cache misses) and the need to manipulate large numbers. For many decades, researchers have sought techniques that are more efficient in practice *and also* have worst-case guarantees.[6] The lack of success in finding such algorithms partly motivated Conjecture 1.1 in the 90s [65, 56, 9]. A definition of this notion has to be empirical, and determining if our algorithm satisfies it requires experiments. Regularity decompositions are infamously impractical, but our underlying paradigm of decomposing the input into pseudo-random parts has the potential to be practical.

- **Generalizability:** Much of the interest in BMM and triangle detection is because they are simplified special cases of more difficult, important problems. A technique that only solves the special case is not satisfying. In the following, we give four examples of such problems where (1) for well-established reasons, Strassen's technique does not give a truly subcubic algorithm, and (2) a truly subcubic algorithm that does generalize would be groundbreaking as it would refute a popular conjecture in fine-grained complexity that has nothing to do with "combinatorial algorithms", i.e. we do not know how to refute it with any algorithms. In each case, a precise definition of "combinatorial" in the sense of generalizing to that particular setting can readily be made: it is an algorithm that solves the corresponding problem.

[6]Note that the latter requirement is crucial because very fast matrix multiplication algorithms do exist in practice, but are due to highly optimized implementations of the cubic time solution and to special-purpose hardware whose sole purpose is to perform these implementations. Needless to say, a real algorithmic speedup is still desirable.

– To refute the famous All-Pairs Shortest-Paths (APSP) Conjecture, it is enough (in fact, equivalent) to solve *weighted* generalizations of BMM and triangle detection in truly subcubic time: $(min, +)$-matrix-multiplication and Negative-Triangle [72]. The issue with Strassen's technique is that it exploits cancellations by subtracting numbers, and min does not have an inverse.

– To refute the Online Matrix-Vector (OMV) Conjecture [48], we need to solve BMM in an *online* setting in which the columns of the second matrix arrive one by one and, at each step, we must output the answer before seeing the next column, in a total time that is truly subcubic. The issue with Strassen's algorithm is that its formulas depend on later columns.

– To refute the Hyper-Clique Conjecture [58], we need a generalization to *hypergraphs* that lets us detect a 4-clique in a 3-uniform hypergraph. A technique entirely different from Strassen's is needed because formulas that reduce the number of multiplications provably do not exist—the border rank of the corresponding tensor matches the trivial upper bound [58].

– To refute the famous 3-SUM Conjecture [62, 54] (and also the APSP Conjecture [73]), it is enough to obtain a generalization to the (witness) *reporting* setting. In particular, we would like an algorithm that preprocesses a graph in truly subcubic time and can then enumerate all triangles with constant delay. Unfortunately, the witness information is lost under the cancellations that are exploited in Strassen's algorithm.

Does our new algorithm and, more generally, the regularity decompositions technique generalize to these four settings? For the first three, it is unclear and left for future research (there was much less incentive to do it before this work since it was outperformed by divide-and-conquer).[7] For the reporting setting, Abboud, Fischer, and Shechter [4] recently observed that the ideas in the Bansal-Williams algorithm can give a triangle enumeration algorithm with $n^3/\log^{2.25} n$ preprocessing and constant delay, which is the state-of-the-art even when using algebraic techniques. Building on our new decomposition theorem, we give an improved bound, demonstrating that our techniques are useful beyond "combinatorial" algorithms.

Theorem 1.3 (Triangle Enumeration Algorithm)**.** *There is a deterministic algorithm that preprocesses a given graph in time $n^3/(\log n)^6 \cdot (\log\log n)^{O(1)}$ and then enumerates all triangles with constant delay.*

It should appear strange that we shave only a $\log^6 n$ factor and not a super-poly-log. The reason for this is that $O(n^3/\log^6 n)$ is the best we know how to achieve (using any technique) for triangle enumeration *even on random graphs* (see Section 7.1). Thus, it is a natural limit for any technique (like ours) based on a reduction to random-like instances. Moreover, due to the reduction from 3-SUM to triangle enumeration [62, 54], shaving any additional $\log^{\epsilon} n$ factor over our bound would improve on the longstanding upper bound for (integer) 3-SUM [14] (see Section 7.4).

[7]For the first [71] and second [55] settings, breakthrough works have already managed to shave a quasi-polynomial factor by fancy reductions to matrix multiplication (and then using the algebraic methods); nonetheless, achieving such improvements with our combinatorial methods would be interesting. For the third setting, it is a big open question.

2 Preliminaries

We write $[n] = \{1, \dots, n\}$ and $\mathrm{poly}(n) = n^{O(1)}$. Occasionally, we write $a = b \pm \epsilon$ to express that $|a - b| \leq \epsilon$.

2.1 Machine Model

We assume the standard Word RAM model with word size $\Theta(\log n)$ (where n is the input size). Since, for most of our algorithmic results, additional log-factors in the running times would not matter, the choice of the machine model is not crucial. Only in Section 7, when we care about log-factors, this choice matters.

2.2 Graphs and Matrices

We typically denote sets (of nodes) by X, Y, Z and matrices by A, B, C. Moreover, we typically view binary matrices $A \in \{0,1\}^{X \times Y}$ as bipartite graphs on the node sets X, Y, where an edge (x, y) is present if and only if $A(x, y) = 1$.

Let $A \in \mathbb{R}_{\geq 0}^{X \times Y}$ and $B \in \mathbb{R}_{\geq 0}^{Y \times Z}$ be matrices. We denote by AB their standard matrix product, and by $A \circ B$ a scaled matrix product defined by

$$(A \circ B)(x, z) = \mathop{\mathbb{E}}_{y \in Y} A(x, y) B(y, z) = \frac{1}{|Y|} \cdot \sum_{y \in Y} A(x, y) B(y, z).$$

Following the bipartite graph analogy, for sets $X' \subseteq X$ and $Y' \subseteq Y$, we let $A[X', Y'] \in \mathbb{R}_{\geq 0}^{X \times Y}$ denote the submatrix restricted to the rows in X' and the columns in Y'—that is, the subgraph induced by $X' \cup Y'$. Let A^T denote the transpose of A—that is, the subgraph obtained by exchanging the sides X and Y. We call

$$\mathbb{E}[A] = \mathop{\mathbb{E}}_{\substack{x \in X \\ y \in Y}} A(x, y) = \frac{1}{|X|\,|Y|} \cdot \sum_{\substack{x \in X \\ y \in Y}} A(x, y)$$

the *density* of A. For nodes $x \in X$ and $y \in Y$, we define their *(relative) degrees* as

$$\deg_A(x) = \mathop{\mathbb{E}}_{y \in Y} A(x, y) = \frac{1}{|Y|} \cdot \sum_{y \in Y} A(x, y),$$
$$\deg_A(y) = \mathop{\mathbb{E}}_{x \in X} A(x, y) = \frac{1}{|X|} \cdot \sum_{x \in X} A(x, y).$$

We say that A is *ϵ-left-min-degree* or simply *ϵ-min-degree* if $\min_{x \in X} \deg_A(x) \geq (1 - \epsilon)\,\mathbb{E}[A]$. (The symmetric notion of *ϵ-right-min-degree* is never used in the paper.) Moreover, for $\alpha, \epsilon, \delta \geq 0$, we say that A is *$(\alpha, \epsilon, \delta)$-uniform* if

$$\mathop{\mathbb{P}}_{(x,y) \in X \times Y} [\, (1 - \epsilon)\alpha \leq A(x, y) \leq (1 + \epsilon)\alpha \,] \geq 1 - \delta.$$

2.3 Grid Regularity

Recall that we abstractly consider a graph *regular* if it behaves somewhat pseudo-randomly.[8] In this paper we employ the following formal notion of regularity, defined via the *"grid norm"* of a matrix. Specifically, for a matrix with non-negative entries $A \in \mathbb{R}_{\geq 0}^{X \times Y}$ and integers $k, \ell \geq 1$, we define its (k,ℓ)-grid norm as

$$\|A\|_{U(k,\ell)} = \left(\mathop{\mathbb{E}}_{\substack{x_1,\dots,x_k \in X \\ y_1,\dots,y_\ell \in Y}} \prod_{\substack{i \in [k] \\ j \in [\ell]}} A(x_i, y_j) \right)^{\frac{1}{k\ell}} ;$$

note that equivalently

$$\|A\|_{U(k,\ell)}^{k\ell} = \mathop{\mathbb{E}}_{x_1,\dots,x_k \in X} \left(\mathop{\mathbb{E}}_{y \in Y} \prod_{i \in [k]} A(x_i, y) \right)^{\ell} = \mathop{\mathbb{E}}_{y_1,\dots,y_\ell \in Y} \left(\mathop{\mathbb{E}}_{x \in X} \prod_{i \in [\ell]} A(x, y_j) \right)^{k}.$$

Strictly speaking, $\|\cdot\|_{U(k,\ell)}$ is not necessarily a norm, but we will nevertheless intuitively treat it as such.[9] In combinatorial terms, the grid norm of a bipartite graph $A \in \{0,1\}^{X \times Y}$ measures (up to normalization) the number of (k,ℓ)-bicliques that occur as subgraphs of A (including subgraphs in which some nodes of the biclique coincide).

Note that the grid norm $\|A\|_{U(k,\ell)}$ ranges from $\mathbb{E}[A]$ to 1, and thereby constitutes some measure of pseudo-randomness: On the one hand, purely random bipartite graphs (with edge density $\mathbb{E}[A]$) have grid norm $\|A\|_{U(k,\ell)} \approx \mathbb{E}[A]$, whereas structured graphs (e.g., graphs with large induced subgraphs of increased density) often have larger grid norms. In this spirit, we say that A is *(ϵ, k, ℓ)-regular* if

$$\|A\|_{U(k,\ell)} \leq (1+\epsilon)\,\mathbb{E}[A].$$

For specific constant values of k and ℓ, grid norms have appeared in many previous mathematical works (e.g., [43, 44]), and also implicitly in recent algorithmic structure-to-randomness reductions [3, 2, 51].

2.4 Kelley-Lovett-Meka's Structural Theorem

What makes grid norms useful for us? In a recent result, Kelley, Lovett and Meka [52] use analytical methods to obtain the following structural result, linking the regularity of two graphs A and B to their product matrix.

Theorem 2.1 (Regular Matrices Have Uniform Products [52, Lemma 4.8])**.** *Let $A \in \mathbb{R}_{\geq 0}^{X \times Y}$ and $B \in \mathbb{R}_{\geq 0}^{Y \times Z}$, let $\epsilon \in (0, \frac{1}{80})$, let $d \geq 2/\epsilon$ and assume that*

(a) A and B^T are $(\epsilon, 2, d)$-regular, and

(b) A and B^T are ϵ-min-degree.

Then $A \circ B$ is $(\mathbb{E}[A]\,\mathbb{E}[B], 80\epsilon, 2^{-\epsilon d/2})$-uniform.

[8]This concept is *not* related to degree-regularity (where each node in the graph has the same number of neighbors).

[9]It is however known that $\|\cdot\|_{U(k,\ell)}$ is a semi-norm whenever k and ℓ are even [47, Theorems 2.8 and 2.9].

3 Technical Overview

Theorem 2.1 is the starting point Our goal in the following is to exploit this structural theorem algorithmically and derive an improved combinatorial algorithm for Boolean matrix multiplication. In this section we describe our key ideas.

Boolean Matrix Multiplication and Triangle Detection. Recall that the Triangle Detection problem is to test whether a given undirected, tripartite graph (X, Y, Z, A, B, C) (with vertex parts X, Y, Z and edge parts $A \in \{0,1\}^{X\times Y}, B \in \{0,1\}^{Y\times Z}, C \in \{0,1\}^{X\times Z}$) contains a *triangle* (that is, a vertex triple $(x, y, z) \in (X, Y, Z)$ with $A(x, y) = B(y, z) = C(x, z) = 1$). While at first glance Triangle Detection appears to be a simpler problem than BMM (note that the output consists of a single bit versus n^2 bits), it is known since the early days of fine-grained complexity that both problems are, in fact, equivalent in terms of subcubic algorithms [72]. Specifically, if Triangle Detection can be solved in time $O(n^3/f(n))$, then Boolean Matrix Multiplication is in time $O(n^3/f(n^{1/3}))$. This reduction is essentially loss-less for the quasi-polynomial speed-up that we aim for in this paper. Therefore, we focus on designing an efficient algorithm for Triangle Detection in the following exposition.

Triangle Detection on Regular Graphs. We start by describing a dream scenario to understand how Kelley-Lovett-Meka's structural theorem [52] comes into play.

Our aim is to solve Triangle Detection in time $n^3/2^{\Omega(d)}$ for some parameter d. Moreover, let $\epsilon > 0$ be a small constant (say, $\epsilon = \frac{1}{160}$). For the dream scenario suppose that the edge parts A and B are regular in the sense of Theorem 2.1:

(a) A and B^T are $(\epsilon, 2, d)$-regular, and

(b) A and B^T are ϵ-min-degree.

Under these assumptions, Theorem 2.1 yields that the scaled matrix product $A \circ B$ is $(\mathbb{E}[A]\,\mathbb{E}[B], 80\epsilon, 2^{-\epsilon d/2})$-uniform. Explicitly, for our choice of $\epsilon = \frac{1}{160}$, this means that at least a $(1-2^{-\epsilon d/2})$-fraction of the entries of $A \circ B$ fall in the range $[\frac{1}{2}\,\mathbb{E}[A]\,\mathbb{E}[B], \frac{3}{2}\,\mathbb{E}[A]\,\mathbb{E}[B]]$. In particular, the matrix $A \circ B$ (and thereby also AB) has zeros in at most a $2^{-\epsilon d/2}$-fraction of its entries (assuming that A and B are nonzero).

This puts us in a win-win situation: Either the matrix C is sparse ($\mathbb{E}[C] \leq 2^{-\epsilon d/2}$), in which case we can detect a triangle in time $n^3/2^{\Omega(d)}$ (by enumerating all $n^2/2^{\Omega(d)}$ edges in C and all remaining nodes $y \in Y$). Or the matrix C is dense ($\mathbb{E}[C] > 2^{-\epsilon d/2}$), and it follows from the uniformity that AB and C have a common nonzero entry. Note that this certifies that there is a triangle without the need to compute anything further.

Our Regularity Decomposition. Of course, we cannot simply assume the dream scenario where A and B satisfy the regularity and min-degree conditions. Instead, we hope to decompose A and B into regular pieces in the same flavor as Szemerédi's or Frieze-Kannan's regularity lemmas. For grid regularity, unfortunately, such a decomposition theorem was not known.

One of our key contributions is such a decomposition theorem, see Theorem 3.1. We emphasize that this theorem is novel even existentially (i.e., even without the extra requirement that the decomposition must be computed efficiently).

Theorem 3.1 (AB-Decomposition)**.** *Let $A \in \{0,1\}^{X\times Y}, B \in \{0,1\}^{Y\times Z}$, let $\epsilon \in (0,1)$ and $d \geq 1$. There is an algorithm* ABDecomposition$(X, Y, Z, A, B, \epsilon, d)$ *that computes a collection of tuples $\{(X_k, Y_k, Z_k, A_k, B_k)\}_{k=1}^{K}$, where $X_k \subseteq X$, $Y_k \subseteq Y$, $Z_k \subseteq Z$, $A_k \in \{0,1\}^{X_k\times Y_k}$, $B_k \in \{0,1\}^{Y_k\times Z_k}$ such that*

1. $AB = \sum_{k=1}^{K} A_k B_k$.
2. *For all $k \in [K]$:*
   (i) $\mathbb{E}[A_k] \leq 2^{-d}$ *or* $\mathbb{E}[B_k] \leq 2^{-d}$*, or*
   (ii) A_k *and* B_k^T *are both $(\epsilon, 2, d)$-regular and ϵ-min-degree.*
3. $\sum_{k=1}^{K} |X_k|\,|Y_k|\,|Z_k| \leq 2(d+2)^2 \cdot |X|\,|Y|\,|Z|$.
4. $K \leq \exp(d^7 \operatorname{poly}(\epsilon^{-1}))$.

The algorithm is deterministic and runs in time $n^2 \cdot \exp(d^7 \operatorname{poly}(\epsilon^{-1}))$ (where $n = |X| + |Y| + |Z|$).

To illustrate how these four properties become useful, we complete the description of the Triangle Detection algorithm. We first precompute the decomposition as in the theorem. Additionally, define $C_k = C[X_k, Z_k]$. Property (1) of the theorem states that $AB = \sum_k A_k B_k$ (here, by slight abuse of notation, in the sum we interpret each term $A_k B_k$ as the $X \times Z$-matrix by extending $A_k B_k$ with zeros). Therefore, the set of triangles in the original graph is exactly the disjoint union of the triangles in the tripartite subgraphs $(X_k, Y_k, Z_k, A_k, B_k, C_k)$. It thus remains to detect a triangle in any of these subgraphs.

For each such subgraph, we are again in a win-win situation: If at least one of the edge parts is sparse (i.e., $\mathbb{E}[A_k] \leq 2^{-d}$ or $\mathbb{E}[B_k] \leq 2^{-d}$ or $\mathbb{E}[C_k] \leq 2^{-\epsilon d/2}$), then we can solve the subinstance efficiently in time $|X_k|\,|Y_k|\,|Z_k| \,/\, 2^{\Omega(d)}$. Otherwise, Property (2) of the theorem implies that we are in the dream scenario that A_k and B_k^T are $(\epsilon, 2, d)$-regular and ϵ-min degree. Following the same argument as before, building on the structural Theorem 2.1, it follows that $A_k B_k$ and C_k share a common nonzero entry, which entails the existence of a triangle. In summary, the algorithm solves each sparse subinstance in time $|X_k|\,|Y_k|\,|Z_k| \,/\, 2^{\Omega(d)}$ and stops as soon as it encounters a dense subinstance.

The remaining Properties (3) and (4) are necessary to bound the running time of this algorithm. On the one hand, by Property (3) solving all sparse instances takes total time

$$\sum_{k=1}^{K} \frac{|X_k|\,|Y_k|\,|Z_k|}{2^{\Omega(d)}} \leq |X|\,|Y|\,|Z| \cdot \frac{\operatorname{poly}(d)}{2^{\Omega(d)}} \leq \frac{n^3}{2^{\Omega(d)}}.$$

On the other hand, precomputing the decomposition, and testing for each subinstance, whether it is dense or sparse, takes time $n^2 \cdot \exp(d^7 \operatorname{poly}(\epsilon^{-1})) = n^2 \cdot 2^{O(d^7)}$. By choosing $d = \Theta(\sqrt[7]{\log n})$ sufficiently small such that the precomputation time becomes $O(n^{2.1})$, say, the total running time becomes $n^3/2^{\Omega(\sqrt[7]{\log n})}$ as claimed.

The remainder of this overview is devoted to a proof overview of Theorem 3.1.

3.1 Enforcing Regularity and Min-Degree

Towards proving the decomposition theorem, our first milestone is to develop tools to enforce the (a) regularity and (b) min-degree conditions.

Both tools follow a common theme: To achieve some property we either certify that (a large part of) the given graph already satisfies the property, or that we can alternatively find a large induced subgraph which is substantially *denser* than average (*density increment*). In the former case we have been successful, and in the latter case we will simply *recurse* on the selected denser piece. Since the density increases with each recursive call, we control the recursion depth and the loss we thereby incur. More details follow in Sections 3.2 and 3.3.

Enforcing Min-Degree. Let us start with the conceptually easier min-degree property. Here, specifically, we would like to ensure the ϵ-min-degree condition, i.e. that all nodes $x \in X$ satisfy $\deg_A(x) \geq (1-\epsilon)\,\mathbb{E}[A]$. In fact, it is enough for us if we can find a subgraph of, say, half the total size that satisfies that it is ϵ-min-degree. An easy algorithm is to repeatedly remove low-degree nodes x until the remaining graph becomes ϵ-min-degree. If this algorithm terminates before removing half the nodes, then we have succeeded in finding a large ϵ-min-degree subgraph. If instead the algorithm removes half the nodes in X and the graph A' is still not ϵ-min-degree, then we claim that the remaining graph has density $\mathbb{E}[A'] \geq (1+\frac{\epsilon}{2})\,\mathbb{E}[A]$—i.e., we have found a density increment. For more details, see Lemma 5.1.

Enforcing Regularity. The more challenging task is to ensure that a graph is regular (or that we can alternatively find a density increment). Following the terminology from [52] (which in turn originates from [53]), we refer to this step as "sifting". Specifically, we rely on the following theorem that we will later apply with $k = 2$ and $\ell = d$:

Theorem 3.2 (Sifting). *Let $A \in \{0,1\}^{X \times Y}$, let $\epsilon > 0$ and $k, \ell \geq 1$. There is an algorithm* Sift$(X, Y, A, \epsilon, k, \ell)$ *that returns either*

1. *"regular", in which case A is (ϵ, k, ℓ)-regular, or*
2. *sets $X' \subseteq X, Y' \subseteq Y$ with $|X'|\,|Y'| \geq \frac{\epsilon}{16} \cdot \mathbb{E}[A]^{k\ell} \cdot |X|\,|Y|$ and $\mathbb{E}[A[X', Y']] \geq (1 + \frac{\epsilon}{2})\,\mathbb{E}[A]$.*

The algorithm is deterministic and runs in time $n^2 \cdot (\epsilon\,\mathbb{E}[A]/k)^{-O(k\ell(k+\ell))}$ (where $n = |X| + |Y|$).

The existential claim of Theorem 3.2 was already established by Kelley, Lovett and Meka [52] (up to insignificant changes in the parameters) by a simple "one-shot" proof. While it is possible to turn their ideas into a randomized sifting algorithm, we follow a different proof that can ultimately be turned into a *deterministic* algorithm. The rough idea is to prove that whenever A is not (ϵ, k, ℓ)-regular, then either many nodes $x \in X$ have exceptionally high degree $\deg_A(x) \geq (1 + \frac{\epsilon}{2})\,\mathbb{E}[A]$ (in which case we can return the set X' of such high-degree nodes and $Y' = Y$), or we can find a large induced subgraph of A that is $(\epsilon, k-1, \ell)$-irregular (see Lemma 4.2). In the latter case, we recurse on that subgraph, so after at most k recursive calls we find a density increment.

In order to detect this exceptionally irregular subgraph, it is necessary to obtain an accurate estimate of its grid norm. To this end, we prove that any grid norm $\|A\|_{U(k,\ell)}$ can be approximated up to some additive error $\alpha > 0$ in time $n^2 \cdot \alpha^{-O(k\ell(k+\ell))}$ by a deterministic algorithm (Lemma 4.6). Here we crucially build on the technology of *oblivious samplers*. We defer further details to Section 4.

3.2 Decomposing A

As a warm-up and building block towards Theorem 3.1, let us first focus on decomposing a single bipartite graph $A \in \{0,1\}^{X \times Y}$. Specifically, we establish the following decomposition with four analogous properties to Theorem 3.1.

Theorem 3.3 (A-Decomposition)**.** *Let $A \in \{0,1\}^{X \times Y}$, let $\epsilon \in (0,1)$ and $d \geq 1$. There is an algorithm* ADECOMPOSITION(X, Y, A, ϵ, d) *computing tuples $\{(X_\ell, Y_\ell, A_\ell)\}_{\ell=1}^{L}$ with $X_\ell \subseteq X$, $Y_\ell \subseteq Y$, and $A_\ell \in \{0,1\}^{X_\ell \times Y_\ell}$ such that:*

1. *$A = \sum_{\ell=1}^{L} A_\ell$.*
2. *For all $\ell \in [L]$:*
   - *(i) $\mathbb{E}[A_\ell] \leq 2^{-d}$, or*
   - *(ii) A_ℓ is $(\epsilon, 2, d)$-regular and ϵ-min-degree.*
3. *$\sum_{\ell=1}^{L} |X_\ell|\,|Y_\ell| \leq (d+2) \cdot |X|\,|Y|$.*
4. *$L \leq \exp(d^3 \operatorname{poly}(\epsilon^{-1}))$ and $\min_{\ell=1}^{L} |X_\ell|\,|Y_\ell| \geq \exp(-d^3 \operatorname{poly}(\epsilon^{-1})) \cdot |X|\,|Y|$.*

The algorithm is deterministic and runs in time $n^2 \cdot \exp(d^3 \operatorname{poly}(\epsilon^{-1}))$ (where $n = |X| + |Y|$).

Theorem 3.3 in itself is already an interesting regularity decomposition, which we believe will likely find further applications in the future. The proof of the theorem is along the following lines. First of all, if $\mathbb{E}[A] \leq 2^{-d}$, then we simply return the trivial decomposition $\{(X, Y, A)\}$. So assume from now on that $\mathbb{E}[A] \geq 2^{-d}$.

Consider the following subtask (see Lemma 5.2): The goal is to find $X^* \subseteq X$ and $Y^* \subseteq Y$ such that the induced subgraph $A[X^*, Y^*]$ is $(\epsilon, 2, d)$-regular and ϵ-min-degree; we call $X^* \times Y^*$ a *good rectangle*. We can find a good rectangle using the density increment technique. First, make half of X satisfy the min-degree condition. Then, apply Theorem 3.2 to certify that this remaining graph is $(\epsilon, 2, d)$-regular. If both steps succeed we have successfully identified a good rectangle (namely, the entire remaining graph). Otherwise, if either step fails and instead returns a large subgraph with density at least $(1 + \frac{\epsilon}{2})\,\mathbb{E}[A]$, we simply *recurse* on the denser subgraph to find a good rectangle. With each recursive call the density strictly increases, and thus this process eventually terminates. In fact, the recursion depth is bounded by $O(d/\epsilon)$ given that the initial density is $\mathbb{E}[A] \geq 2^{-d}$. Since with each recursive call we reduce the number of vertices to a $(\epsilon\,\mathbb{E}[A])^{-O(d)} = \exp(-d^2 \operatorname{poly}(\epsilon^{-1}))$-fraction (by Theorem 3.2), the returned good rectangle covers at least a $\exp(-d^3 \operatorname{poly}(\epsilon^{-1}))$-fraction of the original graph.

Coming back to Theorem 3.3, we can compute the decomposition using density *decrements*. Namely, we repeatedly find good rectangles $X^* \times Y^*$ as in the previous paragraph, take the subgraph $(X^*, Y^*, A[X^*, Y^*])$ as one part in the decomposition, and then remove all edges in the rectangle $X^* \times Y^*$ from A. Eventually A becomes 2^{-d}-sparse and at this point we return the remaining trivial decomposition $\{(X, Y, A)\}$. In each step we remove $\exp(-d^3 \operatorname{poly}(\epsilon^{-1})) \cdot |X| \cdot |Y|$ edges from A, and therefore this process leads to at most $L \leq \exp(d^3 \operatorname{poly}(\epsilon^{-1}))$ many parts.

So far we have neglected Property 3, but it turns out that a closer inspection of this process indeed yields that $\sum_{\ell=1}^{L} |X_\ell|\,|Y_\ell| \leq O(d) \cdot |X|\,|Y|$. Proving this statement builds on the critical insight that, for any good rectangle that we remove, we always have $\mathbb{E}[A[X^*, Y^*]] \geq \mathbb{E}[A]$. Specifically, let $\overline{A}_\ell$ denote the remaining matrix A before the ℓ-th step of the algorithm, and let L_1 be the smallest index such that $\mathbb{E}[\overline{A}_{L_1}] \leq \frac{1}{2}\,\mathbb{E}[A]$. Then we can express

$$\mathbb{E}[A] = \sum_{\ell=1}^{L_1-1} \mathbb{E}[A_\ell] \cdot \frac{|X_\ell|\,|Y_\ell|}{|X|\,|Y|} + \mathbb{E}[\overline{A}_{L_1}] \geq \frac{\mathbb{E}[A]}{2} \cdot \sum_{\ell=1}^{L_1-1} \frac{|X_\ell|\,|Y_\ell|}{|X|\,|Y|},$$

using that $\mathbb{E}[A_\ell] \geq \mathbb{E}[\overline{A}_\ell] > \frac{1}{2}\,\mathbb{E}[A]$. It follows that $\sum_{\ell=1}^{L_1-1} |X_\ell|\,|Y_\ell| \leq 2 \cdot |X|\,|Y|$. We can apply the same argument to analyze the algorithm in *phases*. That is, letting L_i be the smallest step with $\mathbb{E}[\overline{A}_{L_i}] \leq \frac{1}{2^i}\,\mathbb{E}[A]$, we can similarly bound $\sum_{\ell=L_i}^{L_{i+1}-1} |X_\ell|\,|Y_\ell| \leq 2 \cdot |X|\,|Y|$ for each phase. After the d-th phase the process has reduced the density of the remaining graph to at most 2^{-d}, and the process terminates. Therefore, all in all, we have $\sum_{\ell=1}^{L} |X_\ell|\,|Y_\ell| \leq 2d \cdot |X|\,|Y|$. (In the formal proof we obtain a slightly sharper bound.)

3.3 Decomposing AB

We finally turn to the full decomposition from Theorem 3.1. The idea is to use the one-part decomposition developed in the previous subsection as a black-box to decompose A, and to decompose B via density increments/decrements. Unfortunately, the details of this step are much more intricate.

Let us first sketch an approach that will not work out as planned. In light of the previous subsection, the hope is that we can find a good rectangle $Y^* \times Z^*$ in B along with a regularity decomposition $\{(X_\ell, Y_\ell, A_\ell)\}_{\ell=1}^{L}$ of $A[X, Y^*]$, such that additionally each subgraph $B[Y_\ell, Z^*]$ is $(\epsilon, 2, d)$-regular and ϵ-min-degree (see Lemma 5.3). We loosely refer to $Y^*, Z^*, \{(X_\ell, Y_\ell, A_\ell)\}_{\ell=1}^{L}$ as a *good cube*. If there was an algorithm to find good cubes, then we would easily obtain the desired decomposition: We repeatedly find a good cube $Y^*, Z^*, \{(X_\ell, Y_\ell, A_\ell)\}_{\ell=1}^{L}$, emit $\{(X_\ell, Y_\ell, Z^*, A_\ell, B[Y_\ell, Z^*])\}_{\ell=1}^{L}$ as parts in the decomposition and remove the edges in $Y^* \times Z^*$ from B.

However, we face serious problems trying to find a good cube. The natural idea is to use the density increment technique to find $Y^* \times Z^* \subseteq Y \times Z$ such that $B[Y^*, Z^*]^T$ is $(\epsilon, 2, d)$-regular and ϵ-min-degree. Then we can apply Theorem 3.3 to decompose the matrix $A[X, Y^*]$ into pieces $\{(X_\ell, Y_\ell, A_\ell)\}_{\ell=1}^{L}$. However, the subgraphs $B[Y_\ell, Z^*]$ are not necessarily $(\epsilon, 2, d)$-regular and ϵ-min-degree. The first issue is fixable: Using the sifting algorithm we can test whether all subgraphs $B[Y_\ell, Z^*]$ are $(\epsilon, 2, d)$-regular—if any such subgraphs fails this test, then Theorem 3.2 instead returns a denser subgraph of $B[Y_\ell, Z^*]$. We thus recurse on that subgraph to find a good cube. The second issue, that $B[Y_\ell, Z^*]$ is not ϵ-min-degree, is more serious. In contrast to the regularity condition we cannot enforce the min-degree condition for the whole graph $B[Y_\ell, Z^*]$, but only for a subgraph $B[Y_\ell, Z_\ell]$, where $Z_\ell \subseteq Z^*$ is large.

Our solution to this issue is somewhat reminiscent to the divide-and-conquer approaches for Triangle Detection. As outlined before, we can compute a good cube $Y^*, Z^*, \{(X_\ell, Y_\ell, Z_\ell, A_\ell)\}$ such that each subgraph $B[Y_\ell, Z_\ell]$ is $(\epsilon, 2, d)$-regular and ϵ-min-degree. By tweaking the parameters of the min-degree lemma, we can further achieve that $|Z_\ell| \geq (1 - \gamma)|Z^*|$ for some parameter γ to be determined soon. As before, for each good cube we emit $\{(X_\ell, Y_\ell, Z_\ell, A_\ell, B[Y_\ell, Z_\ell])\}$ as one part of the decomposition, and then remove the edges in $Y^* \times Z^*$ from B and repeat. However, we additionally recurse on all subinstances on the vertex parts $(X_\ell, Y_\ell, Z^* \setminus Z_\ell)$ to cover the edges missed in the previous parts. To control the cost caused by this additional layer of recursion we need to guarantee that

$$\sum_{\ell=1}^{L} |X_\ell|\,|Y_\ell|\,|Z^* \setminus Z_\ell| \le \tfrac{1}{2} \cdot |X|\,|Y|\,|Z|,$$

say. And indeed, using that $\sum_{\ell=1}^{L} |X_\ell|\,|Y_\ell| \le \mathrm{poly}(d)\cdot|X|\,|Y|$ and by setting $\gamma = \frac{1}{\mathrm{poly}(d)}$ small enough this can be achieved. The overhead nevertheless leads to a significant blow-up in the dependence on d, from $\exp(d^3)$ to $\exp(d^7)$. See Section 5 for the details.

3.4 Further Improvements?

While we have successfully achieved quasi-polynomial savings combinatorially for BMM (and many other problems), Conjecture 1.1 still stands, and the question remains whether we can do better. E.g., can we achieve savings of the form $2^{\Theta(\sqrt{\log n})}$ rather than $2^{\Theta(\sqrt[7]{\log n})}$, or possibly even truly polynomial savings?

Over random matrices in which each entry is 1 with probability p, we can solve BMM in $\widetilde{O}(n^{2.5})$ time,[10] which means that the general framework of worst-case to random-case reductions via regularity decompositions could go much further. However, it is not clear whether the specific notion of grid regularity can go beyond $2^{\Theta(\sqrt{\log n})}$ savings, because the known lower bounds for Triangle Removal (à la Behrend's construction [15, 76]) seem to apply as well. We have focused on presenting our new technique in an easy and modular fashion rather than on optimizing the constant in the quasi-polynomial savings. It remains an interesting open question to fine-tune the parameters.

4 Sifting

In this section we describe the "sifting" algorithm that, given a graph A, either determines that A is regular or finds a subgraph of A that is denser than average. Kelley, Lovett and Meka [52] have proved this statement via a non-algorithmic proof that can rather easily be turned into a randomized algorithm. Our approach here differs from that original version, as our more ambitious goal is to obtain a *deterministic* sifting algorithm. We start with a simple inverse of Markov's inequality:

Lemma 4.1 (Inverse of Markov's Inequality)**.** *Let Z be a random variable that takes values in $[0,1]$. Then, for any $\alpha \in [0,1]$,*

$$\mathbb{P}[Z \ge \alpha] \ge \mathbb{E}[Z] - \alpha.$$

Proof. Observe that $\mathbb{1}(Z \ge \alpha) \ge Z - \alpha$. Thus, $\mathbb{P}[Z \ge \alpha] = \mathbb{E}[\mathbb{1}(Z \ge \alpha)] \ge \mathbb{E}[Z] - \alpha$. ☐

The high-level idea behind the sifting algorithm is that we can either (1) find a denser subgraph by simply taking the high-degree nodes, or (2) recurse on a smaller subgraph with parameter $k-1$. This idea is recorded in the next lemma. Here and for the remainder of this section we write $Y_x = \{y \in Y : A(x,y) = 1\}$ and $A_x = A[X, Y_x]$.

Lemma 4.2 (Recursive Sifting)**.** *Let $A \in \{0,1\}^{X\times Y}$, let $\delta, \epsilon > 0$ and $k, \ell \ge 1$ and assume that $\|A\|_{U(k,\ell)} \ge (1+\epsilon)\delta$. Then one of the following two cases applies:*

[10]For $p \gg 1/\sqrt{n}$ the answer is the all-ones matrix with good probability, and otherwise the matrices are sparse.

1. $|\{x \in X : \deg_A(x) \geq \delta\}| \geq \frac{\epsilon}{2} \cdot \delta^{k\ell} \cdot |X|$,

2. or $k > 1$ *and there is some* $x \in X$ *such that:*

- $\deg_A(x) \geq \delta^k$, *and*
- $\|A_x\|_{U(k-1,\ell)} \geq (1+\epsilon)\delta$.

Proof. First consider the case $k = 1$. We prove that case 1 applies by sampling $x \in X$ uniformly at random, and showing that with probability at least $\epsilon \cdot \delta^\ell$ this choice satisfies $\deg_A(x) \geq \delta$. Indeed, using the inverse Markov inequality:

$$\begin{aligned}\mathop{\mathbb{P}}_{x\in X}\left[\deg_A(x)^\ell \geq \delta^\ell\right] &\geq \mathop{\mathbb{E}}_{x\in X} \deg_A(x)^\ell - \delta^\ell = \mathop{\mathbb{E}}_{x\in X}\left(\mathop{\mathbb{E}}_{y\in Y} A(x,y)\right)^\ell - \delta^\ell \\ &= \|A\|^\ell_{U(1,\ell)} - \delta^\ell \geq (1+\epsilon)^\ell\delta^\ell - \delta^\ell \geq \epsilon \cdot \delta^\ell.\end{aligned}$$

Next, let $k > 1$ and suppose that case 1 does not hold. We prove that selecting a uniformly random element $x \in X$ satisfies case 2 with positive probability. In fact, we prove that a uniformly random element $x \in X$ satisfies the following two stronger properties with positive probability:

(i) $\deg_A(x) \leq \delta$,

(ii) $\deg_A(x) \cdot \|A_x\|^{k-1}_{U(k-1,\ell)} \geq (1+\epsilon)^{k-1}\delta^k$.

Clearly, (i) fails with probability at most $\frac{\epsilon}{2} \cdot \delta^{k\ell}$ (by the assumption that case 1 of the lemma statement does not hold). Considering property (ii), we first bound the following expectation:

$$\begin{aligned}&\mathop{\mathbb{E}}_{x\in X} \deg_A(x)^\ell \cdot \|A_x\|^{(k-1)\ell}_{U(k-1,\ell)} \\ &\qquad = \mathop{\mathbb{E}}_{x\in X}\ \mathop{\mathbb{E}}_{x_2,\dots,x_k\in X}\left(\deg_A(x) \cdot \mathop{\mathbb{E}}_{y\in Y_x} \prod_{i=2}^{k} A(x_i,y)\right)^\ell\end{aligned}$$

Recall that Y_x is the set of neighbors of x. Hence, for any function f we can rewrite the expectation $\deg_A(x) \cdot \mathbb{E}_{y\in Y_x} f(y)$ as $\mathbb{E}_{y\in Y} f(y) \cdot A(x,y)$, and thus

$$\begin{aligned}&= \mathop{\mathbb{E}}_{x_1,\dots,x_k\in X}\left(\mathop{\mathbb{E}}_{y\in Y} \prod_{i\in[k]} A(x_i,y)\right)^\ell \\ &= \mathop{\mathbb{E}}_{\substack{x_1,\dots,x_k\in X\\ y_1,\dots,y_\ell\in Y}} \prod_{\substack{i\in[k]\\ j\in[\ell]}} A(x_i,y_j) \\ &= \|A\|^{k\ell}_{U(k,\ell)}.\end{aligned}$$

Therefore, by the inverse Markov inequality, for a uniformly random $x \in X$ property (ii) holds with probability at least

$$\begin{aligned}
&\mathop{\mathbb{P}}_{x\in X}\Big[\deg_A(x)\cdot\|A_x\|_{U(k-1,\ell)}^{k-1} \geq (1+\epsilon)^{k-1}\delta^k\Big]\\
&\quad= \mathop{\mathbb{P}}_{x\in X}\Big[\deg_A(x)^\ell\cdot\|A_x\|_{U(k-1,\ell)}^{(k-1)\ell} \geq (1+\epsilon)^{(k-1)\ell}\delta^{k\ell}\Big]\\
&\quad\geq \mathop{\mathbb{E}}_{x\in X}\deg_A(x)^\ell\cdot\|A_x\|_{U(k-1,\ell)}^{(k-1)\ell} - (1+\epsilon)^{(k-1)\ell}\delta^{k\ell}\\
&\quad\geq \|A\|_{U(k,\ell)}^{k\ell} - (1+\epsilon)^{(k-1)\ell}\delta^{k\ell}.\\
&\quad\geq (1+\epsilon)^{k\ell}\delta^{k\ell} - (1+\epsilon)^{(k-1)\ell}\delta^{k\ell}\\
&\quad\geq \epsilon\cdot\delta^{k\ell}.
\end{aligned}$$

By a union bound, both properties (i) and (ii) hold simultaneously with positive probability at least $\frac{\epsilon}{2}\cdot\mathbb{E}[A]^{k\ell}$.

Finally, consider an element x satisfying (i) and (ii); we show that x also satisfies the two conditions from the lemma statement. Since $\delta\cdot\|A_x\|_{U(k-1,\ell)}^{k-1} \geq \deg_A(x)\cdot\|A_x\|_{U(k-1,\ell)}^{k-1} \geq (1+\epsilon)^{k-1}\delta^k$, it follows that indeed $\|A_x\|_{U(k-1,\ell)} \geq (1+\epsilon)\delta$. Moreover, using the trivial bound $\|A_x\|_{U(k-1,\ell)} \leq 1$, we conclude that $\deg_A(x) \geq \delta^k$. □

For the sifting algorithm we also need the following lemma about approximating $\|\cdot\|_{U(k,\ell)}$. We postpone the deterministic proof of Lemma 4.3 to Section 4.1, and encourage the reader to instead think of Lemma 4.3 as the straightforward randomized algorithm (that subsamples $X^k \times Y^\ell$ to approximately count the number of (k,ℓ)-bicliques).

Lemma 4.3 (Deterministic Regularity Approximation)**.** *Let $A \in \{0,1\}^{X\times Y}$, let $\alpha > 0$ and $k, \ell \geq 1$. There is a deterministic algorithm that computes, for all $x \in X$, an approximation v_x satisfying that $v_x = \|A_x\|_{U(k,\ell)} \pm (\alpha/\deg_A(x)^{\frac{1}{k}})$, and runs in time $n^2\cdot\alpha^{-O(k\ell(k+\ell))}$ (where $n = |X| + |Y|$).*

We are ready to prove Theorem 3.2. For convenience, we restate the statement here.

Theorem 3.2 (Sifting)**.** *Let $A \in \{0,1\}^{X\times Y}$, let $\epsilon > 0$ and $k, \ell \geq 1$. There is an algorithm* Sift$(X, Y, A, \epsilon, k, \ell)$ *that returns either*

1. *"regular", in which case A is (ϵ, k, ℓ)-regular, or*
2. *sets $X' \subseteq X, Y' \subseteq Y$ with $|X'|\,|Y'| \geq \frac{\epsilon}{16}\cdot\mathbb{E}[A]^{k\ell}\cdot|X|\,|Y|$ and $\mathbb{E}[A[X',Y']] \geq (1+\frac{\epsilon}{2})\,\mathbb{E}[A]$.*

The algorithm is deterministic and runs in time $n^2\cdot(\epsilon\,\mathbb{E}[A]/k)^{-O(k\ell(k+\ell))}$ (where $n = |X| + |Y|$).

Proof. Throughout we assume that $k \leq \ell$, as otherwise we can simply exchange k and ℓ and work on the transposed graph A^T. Moreover, for the sake of a cleaner presentation we design an algorithm Sift'$(X, Y, A, \delta, \epsilon, k, \ell)$ that receives an additional input parameter $\delta > 0$ with the modified task to return either

1. sets $X' \subseteq X, Y' \subseteq Y$ with $|X'|\,|Y'| \geq \frac{\epsilon}{4}\cdot\delta^{k\ell}\cdot|X|\,|Y|$ and $\mathbb{E}[A[X',Y']] \geq \delta$, or
2. "regular", in which case $\|A\|_{U(k,\ell)} \leq (1+\epsilon)\delta$.

Algorithm 4.1 Implements the algorithm from Theorem 3.2.

```
1:  procedure SIFT'($X, Y, A, \delta, \epsilon, k, \ell$)
2:      Let $X' \leftarrow \{x \in X : \deg_A(x) \geq \delta\}$
3:      if $|X'| \geq \frac{\epsilon}{2} \cdot \delta^{k\ell} \cdot |X|$ then
4:          return $X', Y$
5:      if $k = 1$ then
6:          return "regular"
7:      else
8:          Compute approximations $v_x$ of $\|A_x\|_{U(k-1,\ell)}$ by Lemma 4.3 with parameter $\alpha = \frac{\epsilon\delta^2}{2k^2}$
9:          Select $x \in X$ with $\deg_A(x) \geq \delta^k$ maximizing $v_x$
10:         return SIFT'($X, Y_x, A_x, \delta, \epsilon \cdot (1 - \frac{1}{k^2}), k - 1, \ell$)

11: procedure SIFT($X, Y, A, \epsilon, k, \ell$)
12:     return SIFT'($X, Y, A, (1 + \frac{\epsilon}{2})\,\mathbb{E}[A], \frac{\epsilon}{4}, k, \ell$)
```

From this alternative algorithm we can easily obtain the desired algorithm $\textsc{Sift}(X, Y, A, \epsilon, k, \ell)$: Simply call and return $\textsc{Sift'}(X, Y, A, (1 + \frac{\epsilon}{2})\,\mathbb{E}[A], \frac{\epsilon}{4}, k, \ell)$.

We design $\textsc{Sift'}(X, Y, A, \delta, \epsilon, k, \ell)$ as a simple recursive algorithm; see Algorithm 4.1 for the pseudocode. In a first step (Lines 2 to 4) we construct the set $X' \leftarrow \{x \in X : \deg_A(x) \geq \delta\}$ of high-degree nodes. If this set turns out to be sufficiently large, $|X'| \geq \frac{\epsilon}{2} \cdot \delta^{k\ell} \cdot |X|$, we can successfully return X' and $Y' \leftarrow Y$. Otherwise, we distinguish two cases: If $k = 1$, then we simply report "regular" (Lines 5 to 6). If instead $k > 1$, then our principle strategy is to identify a node $x \in X$ such that subgraph A_x is *as irregular as possible,* and to recurse on that subgraph (Lines 8 to 10). Specifically, using Lemma 4.3 we compute approximations v_x of $\|A_x\|_{U(k-1,\ell)}$, for all $x \in X$, with parameter $\alpha = \frac{\epsilon\delta^2}{2k^2}$. We then select the element $x \in X$ with $\deg_A(x) \geq \delta^k$ that maximizes v_x. Finally, we recurse on $\textsc{Sift'}(X, Y_x, A_x, \delta, \epsilon \cdot (1 - \frac{1}{k^2}), k - 1, \ell)$. (Here we tweak the parameter ϵ to account for the loss in the approximation.)

Correctness. First observe that if the algorithm returns sets X', Y', then indeed $\mathbb{E}[A[X', Y']] \geq \delta$. Next we verify by induction that $|X'|\,|Y'| \geq \frac{\epsilon}{2} \cdot (\prod_{i=2}^{k}(1 - \frac{1}{i^2})) \cdot \delta^{k\ell} \cdot |X|\,|Y|$ (which yields the desired bound using that $\prod_{i=2}^{k}(1 - \frac{1}{i^2}) = \frac{k+1}{2k} \geq \frac{1}{2}$). Indeed, in the base case we return the sets X' and $Y' = Y$ with size at least $|X'|\,|Y'| \geq \frac{\epsilon}{2} \cdot \delta^{k\ell} \cdot |X|\,|Y|$. Otherwise, we recursively obtain sets of size $|X'|\,|Y'| \geq \frac{\epsilon}{2} \cdot (1 - \frac{1}{k^2}) \cdot (\prod_{i=2}^{k-1}(1 - \frac{1}{i^2})) \cdot \delta^{(k-1)\ell} \cdot |X|\,|Y_x|$. Recall that $|Y_x| \geq \delta^k \cdot |Y|$, which completes the claim.

It remains to prove that whenever the given graph is irregular, i.e. $\|A\|_{U(k,\ell)} \geq (1 + \epsilon)\delta$, then our algorithm does not return "regular". On the one hand, if $k = 1$ then Lemma 4.2 implies that the set of high-degree nodes $X' = \{x \in X : \deg_A(x) \geq \delta\}$ has size at least $\frac{\epsilon}{2} \cdot \delta^{\ell} \cdot |X|$, and therefore the algorithm terminates in Line 4 by returning X', Y. On the other hand, consider the case $k > 1$. Then either the algorithm terminates in Line 4, or Lemma 4.2 implies that there is some node x^* such that $\deg_A(x^*) \geq \delta^k$ and $\|A_{x^*}\|_{U(k-1,\ell)} \geq (1 + \epsilon)\delta$. Since we select x to be the element satisfying $\deg_A(x) \geq \delta^k$ maximizing the approximation of $\|A_x\|_{U(k-1,\ell)}$ with additive error $\alpha / \deg_A(x)^{\frac{1}{k}} \leq \frac{\epsilon\delta}{2k^2}$, we select an element with $\|A_x\|_{U(k-1,\ell)} \geq (1+\epsilon)\delta - \frac{\epsilon\delta}{k^2} = (1+\epsilon \cdot (1 - \frac{1}{k^2}))\delta$. By induction the recursive call does not return "regular".

Running Time. The recursion depth of the algorithm is at most k, hence it suffices to bound the running time of a single execution. The only costly step is the computation of the approximations of $\|A_x\|_{U(k-1,\ell)}$ which takes time $n^2 \cdot \alpha^{-O(k\ell(k+\ell))} = n^2 \cdot (\epsilon\delta/k)^{-O(k\ell(k+\ell))}$ by Lemma 4.3; all other steps can be implemented in time $O(n^2)$. $\square$

4.1 Deterministic Regularity Approximation

To obtain a deterministic algorithm, it remains to prove Lemma 4.3. As the key tool in our derandomization we rely on *oblivious samplers* as developed in an extensive line of research [26, 41, 77, 39, 63, 46] (see also the survey [40]). For our application the exact dependence on the accuracy parameters ϵ, δ does not matter much, and we thus rely on one of the early constructions:[11]

Lemma 4.4 (Oblivious Sampling [41])**.** *Let X be a set and let $\delta, \epsilon > 0$. There is a deterministic algorithm computing, in time $|X| \cdot \mathrm{poly}(\epsilon^{-1}, \delta^{-1}, \log|X|)$, a family $\mathcal{S}$ of subsets $S \subseteq X$ such that*

1. $|\mathcal{S}| \leq |X| \cdot \mathrm{poly}(\epsilon^{-1}, \delta^{-1})$,
2. $|S| \leq \mathrm{poly}(\epsilon^{-1}, \delta^{-1})$ *for all* $S \in \mathcal{S}$,
3. *For every function* $f : X \to [0,1]$,

$$\mathop{\mathbb{P}}_{S \in \mathcal{S}}\left[\mathop{\mathbb{E}}_{x \in X} f(x) = \mathop{\mathbb{E}}_{x \in S} f(x) \pm \epsilon\right] \geq 1 - \delta.$$

We call $\mathcal{S}$ an (ϵ, δ)-oblivious sampler *of X.*

Lemma 4.5 (Simple Bounds for Additive Approximations)**.** *Let $a, b \in [0,1]$ and $\epsilon > 0, k \geq 1$. Then:*

- *If $a = b \pm \epsilon$, then $a^k = b^k \pm 2\epsilon k$.*
- *If $a^k = b^k \pm \epsilon^k$, then $a = b \pm \epsilon$.*

Proof. The first claim is trivial if $\epsilon \geq \frac{1}{2k}$, so suppose otherwise. Assuming that $a \leq b + \epsilon$, it follows that $a^k - b^k \leq (b+\epsilon)^k - b^k = \sum_{i=1}^k \binom{k}{i} \cdot \epsilon^i \cdot b^{k-i} \leq \sum_{i=1}^k k^i \cdot \epsilon^i \leq \epsilon k \cdot \sum_{i=0}^\infty k^i \cdot \epsilon^i \leq 2\epsilon k$. The second claim is immediate: If $a > b + \epsilon$, then $a^k > (b+\epsilon)^k > b^k + \epsilon^k$ (and similarly if $b > a + \epsilon$). $\square$

Lemma 4.6 (Regularity Approximation via Oblivious Sampling)**.** *Let $A \in \{0,1\}^{X \times Y}$, let $\delta, \epsilon > 0$ and $k, \ell \geq 1$, and let $\mathcal{S}, \mathcal{T}$ be (ϵ, δ)-oblivious samplers of X and Y, respectively. Then:*

$$\|A\|_{U(k,\ell)}^{k\ell} = \mathop{\mathbb{E}}_{\substack{S \in \mathcal{S} \\ T \in \mathcal{T}}} \|A[S,T]\|_{U(k,\ell)}^{k\ell} \pm (2\epsilon k + 2\epsilon\ell + 2\delta).$$

[11] Let us restate Lemma 4.4 in the language of samplers. We arbitrarily identify X with $\{0,1\}^n$ where $n = \lceil \log|X| \rceil$. Lemma 4.4 states that there is a randomized algorithm (an *oblivious sampler*) returning a sample set $S \subseteq \{0,1\}^n$ of size $|S| \leq \mathrm{poly}(\epsilon^{-1}, \delta^{-1})$ such that, for any function $f : \{0,1\}^n \to [0,1]$, with probability at least $1 - \delta$ it holds that $\mathbb{E}_{x \in X} f(x) = \mathbb{E}_{x \in S} f(x) \pm \epsilon$. Moreover, the algorithm runs in polynomial time $\mathrm{poly}(n, \epsilon^{-1}, \delta^{-1})$ and has randomness complexity $n + O(\log \epsilon^{-1}) + O(\log \delta^{-1})$ (i.e., it tosses at most that many unbiased coins). In our formulation, $\mathcal{S}$ denotes the set of all sample sets S obtainable from the algorithm for some sequence of coin tosses.

Proof. Fix $y_1, \dots, y_\ell \in Y$ and consider the function $f(x) = \prod_{j \in [\ell]} A(x, y_j)$. We sample a uniformly random set $S \in \mathcal{S}$. Since $\mathcal{S}$ is an (ϵ, δ)-oblivious sampler of X, with probability at least $1 - \delta$ we have that $\mathbb{E}_{x \in X} f(x) = \mathbb{E}_{x \in S} f(x) \pm \epsilon$ and thus $(\mathbb{E}_{x \in X} f(x))^k = (\mathbb{E}_{x \in S} f(x))^k \pm 2\epsilon k$ (by Lemma 4.5). Since moreover both expectations are $[0, 1]$-bounded, we conclude that

$$\left(\mathop{\mathbb{E}}_{x \in X} f(x)\right)^k = \mathop{\mathbb{E}}_{S \in \mathcal{S}} \left(\mathop{\mathbb{E}}_{x \in S} f(x)\right)^k \pm (2\epsilon k + \delta).$$

Now unfix $y_1, \dots, y_\ell \in Y$. From the previous consideration it follows that

$$\begin{aligned}
\|A\|_{U(k,\ell)}^{k\ell} &= \mathop{\mathbb{E}}_{y_1, \dots, y_\ell \in Y} \left(\mathop{\mathbb{E}}_{x \in X} \prod_{j \in [\ell]} A(x, y_j)\right)^k \pm (2\epsilon k + \delta) \\
&= \mathop{\mathbb{E}}_{S \in \mathcal{S}} \mathop{\mathbb{E}}_{y_1, \dots, y_\ell \in Y} \left(\mathop{\mathbb{E}}_{x \in S} \prod_{j \in [\ell]} A(x, y_j)\right)^k \pm (2\epsilon k + \delta) \\
&= \mathop{\mathbb{E}}_{S \in \mathcal{S}} \|A[S, Y]\|_{U(k,\ell)}^{k\ell} \pm (2\epsilon k + \delta).
\end{aligned}$$

We can now apply the same argument again to $A[S, Y]$, with the roles of X and Y interchanged, to obtain that

$$\|A[S, Y]\|_{U(k,\ell)}^{k\ell} = \mathop{\mathbb{E}}_{T \in \mathcal{T}} \|A[S, T]\|_{U(k,\ell)}^{k\ell} \pm (2\epsilon \ell + \delta),$$

and therefore

$$\|A\|_{U(k,\ell)}^{k\ell} = \mathop{\mathbb{E}}_{\substack{S \in \mathcal{S} \\ T \in \mathcal{T}}} \|A[S, T]\|_{U(k,\ell)}^{k\ell} \pm (2\epsilon k + 2\epsilon \ell + 2\delta)$$

The claim follows. □

Proof of Lemma 4.3. We precompute (ϵ, δ)-oblivious samplers $\mathcal{S}$ and $\mathcal{T}$ of X and Y, respectively, for parameters $\epsilon, \delta > 0$ to be determined later. Then we enumerate each pair $S \in \mathcal{S}, T \in \mathcal{T}$ and each tuple $(x_1, \dots, x_k) \in S^k, (y_1, \dots, y_\ell) \in T^\ell$ to compute the values

$$u_{T, y_1, \dots, y_\ell} \leftarrow \mathop{\mathbb{E}}_{S \in \mathcal{S}} \mathop{\mathbb{E}}_{x_1, \dots, x_k \in S} \prod_{\substack{i \in [k] \\ j \in [\ell]}} A(x_i, y_j).$$

Next, we enumerate each $x \in X$ and compute

$$u_x \leftarrow \left(\mathop{\mathbb{E}}_{T \in \mathcal{T}} \mathop{\mathbb{E}}_{y_1, \dots, y_\ell \in T} u_{T, y_1, \dots, y_\ell} \cdot \prod_{j \in [\ell]} A(x, y_j)\right)^{\frac{1}{k\ell}},$$

by enumerating each $T \in \mathcal{T}$ and each tuple $(y_1, \dots, y_\ell) \in T^\ell$. Finally, we return $v_x \leftarrow u_x / \deg_A(x)^{\frac{1}{k}}$ as the desired approximations.

For the correctness, let $A'_x \in \{0,1\}^{X \times Y}$ be the matrix obtained from A where all columns $y \notin Y_x$ are zeroed out (in contrast to A_x where we have deleted these columns). Then:

$$\begin{aligned}
u_x &= \left(\mathop{\mathbb{E}}_{\substack{S \in \mathcal{S} \\ T \in \mathcal{T}}} \mathop{\mathbb{E}}_{\substack{x_1,\dots,x_k \in S \\ y_1,\dots,y_\ell \in T}} \prod_{\substack{i \in [k] \\ j \in [\ell]}} A(x_i, y_j) \cdot \prod_{j \in [\ell]} A(x, y_j) \right)^{\frac{1}{k\ell}} \\
&= \left(\mathop{\mathbb{E}}_{\substack{S \in \mathcal{S} \\ T \in \mathcal{T}}} \mathop{\mathbb{E}}_{\substack{x_1,\dots,x_k \in S \\ y_1,\dots,y_\ell \in T}} \prod_{\substack{i \in [k] \\ j \in [\ell]}} A'_x(x_i, y_j) \right)^{\frac{1}{k\ell}} \\
&= \left(\mathop{\mathbb{E}}_{\substack{S \in \mathcal{S} \\ T \in \mathcal{T}}} \|A'_x[S,T]\|_{U(k,\ell)}^{k\ell} \right)^{\frac{1}{k\ell}},
\end{aligned}$$

and thus, by the previous Lemmas 4.5 and 4.6,

$$\begin{aligned}
&= \left(\|A'_x\|_{U(k,\ell)}^{k\ell} \pm (2\epsilon k + 2\epsilon \ell + 2\delta) \right)^{\frac{1}{k\ell}} \\
&= \|A'_x\|_{U(k,\ell)} \pm (2\epsilon k + 2\epsilon \ell + 2\delta)^{\frac{1}{k\ell}}.
\end{aligned}$$

By the definitions of A_x and A'_x, we have that $\|A'_x\|_{U(k,\ell)}^{k\ell} = \deg_A(x)^\ell \cdot \|A_x\|_{U(k,\ell)}^{k\ell}$ which implies that $\|A'_x\|_{U(k,\ell)} = \deg_A(x)^{\frac{1}{k}} \cdot \|A_x\|_{U(k,\ell)}$ and thus

$$v_x = \|A_x\|_{U(k,\ell)} \pm \frac{(2\epsilon k + 2\epsilon \ell + 2\delta)^{\frac{1}{k\ell}}}{\deg_A(x)^{\frac{1}{k}}}.$$

To achieve the claimed bound α on the additive error, we choose $\epsilon = \delta = \alpha^{k\ell}/(2k + 2\ell + 2) = \alpha^{O(k\ell)}$.

We finally consider the running time. The precomputation takes time $n \operatorname{poly}(\epsilon^{-1}, \delta^{-1}, \log n)$ by Lemma 4.4. Computing the intermediate values $u_{T,y_1,\dots,y_\ell}$ takes time

$$O\left(\sum_{\substack{S \in \mathcal{S} \\ T \in \mathcal{T}}} |S|^k \cdot |T|^\ell \cdot k\ell \right) = O\left(\sum_{\substack{S \in \mathcal{S} \\ T \in \mathcal{T}}} \operatorname{poly}(\epsilon^{-1}, \delta^{-1})^{k+\ell} \right) = n^2 \cdot \operatorname{poly}(\epsilon^{-1}, \delta^{-1})^{k+\ell}.$$

Similarly, computing the values u_x then takes time $O(|X| \cdot \sum_{T \in \mathcal{T}} |T|^\ell \cdot k\ell) = n^2 \operatorname{poly}(\epsilon^{-1}, \delta^{-1})^\ell$. All contributions are bounded by $n^2 \cdot \alpha^{-O(k\ell(k+\ell))}$. □

5 Regularity Decompositions

In this section we establish the regularity decompositions (Theorems 3.1 and 3.3). The structure of this section closely follows the outline from Section 3.

We start with the following lemma stating that any graph can either be made ϵ-min-degree without loosing many nodes, or we can find a denser subgraph.

Algorithm 5.1 Implements the algorithm from Lemma 5.1.

```
1: procedure MinDegree($X, Y, A, \epsilon, \gamma$)
2:     Initialize $X' \leftarrow X$ and $A' \leftarrow A$
3:     while $\exists x \in X'$ with $\deg_{A'}(x) < (1-\epsilon)\,\mathbb{E}[A']$ do
4:         Update $X' \leftarrow X' \setminus \{x\}$ and $A' \leftarrow A[X', Y]$
5:         if $|X'| \le (1-\gamma) \cdot |X|$ then
6:             return $X'$ (Case 2)
7:     return $X'$ (Case 1)
```

Lemma 5.1 (Minimum Degree)**.** *Let $A \in \{0,1\}^{X \times Y}$, and let $\epsilon, \gamma > 0$. There is an algorithm* MinDegree$(X, Y, A, \epsilon, \gamma)$ *computing a set $X' \subseteq X$ of size $|X'| \ge \lfloor (1-\gamma) \cdot |X| \rfloor$ such that, writing $A' = A[X', Y]$, one of the following cases holds:*

1. *A' is ϵ-min-degree and $\mathbb{E}[A'] \ge \mathbb{E}[A]$, or*
2. *$\mathbb{E}[A'] \ge (1+\gamma\epsilon)\,\mathbb{E}[A]$.*

The algorithm is deterministic and runs in time $O(|X|\,|Y|)$.

Proof. Consider the following algorithm; for the pseudocode see Algorithm 5.1. Initially, we assign $X' \leftarrow X$ and $A' \leftarrow A$. As long as there exists some $x \in X'$ with $\deg_{A'}(x) < (1-\epsilon)\,\mathbb{E}[A']$, we remove x by updating $X' \leftarrow X' \setminus \{x\}$ and $A' \leftarrow A[X', Y]$. When this rule terminates we output the resulting set X' (Case 1). If, however, we reach size $|X'| \le (1-\gamma) \cdot |X|$, then we stop the algorithm prematurely and return the set X' from that stage of the algorithm (Case 2).

Correctness of Case 1. Suppose that the algorithm terminates in Case 1. It is clear that A' is ϵ-min-degree. Moreover, one can easily verify that $\mathbb{E}[A'] \ge \mathbb{E}[A]$ since we have only removed nodes with degree smaller than average. Finally, since the algorithm has not stopped in Case 2 before, we indeed have $|X'| \ge (1-\gamma) \cdot |X|$.

Correctness of Case 2. Suppose now that the algorithm terminates in Case 2. We first argue that $\mathbb{E}[A'] \ge (1+\gamma\epsilon)\,\mathbb{E}[A]$. To this end, we let X' and A' be as when the algorithm terminates. As this happens in the first iteration when $|X'| \le (1-\gamma) \cdot |X|$ we have that $|X'| = \lfloor (1-\gamma) \cdot |X| \rfloor$, and thus also $|X \setminus X'| \ge \gamma \cdot |X|$. Now let $A'' = A[X \setminus X', Y]$. Since the density of A' only increases over the course of the algorithm, and since we only remove nodes with $\deg_{A'}(x) < (1-\epsilon)\,\mathbb{E}[A']$, we have that $\mathbb{E}[A''] < (1-\epsilon)\,\mathbb{E}[A']$. Therefore:

$$\begin{aligned}
\mathbb{E}[A] &= \frac{|X'|}{|X|} \cdot \mathbb{E}[A'] + \frac{|X \setminus X'|}{|X|} \cdot \mathbb{E}[A''] \\
&\le \frac{|X'|}{|X|} \cdot \mathbb{E}[A'] + \frac{|X \setminus X'|}{|X|} \cdot (1-\epsilon)\,\mathbb{E}[A'] \\
&= \left(1 - \epsilon \cdot \frac{|X \setminus X'|}{|X|}\right) \cdot \mathbb{E}[A'] \\
&\le (1-\gamma\epsilon) \cdot \mathbb{E}[A'].
\end{aligned}$$

Rearranging yields that $\mathbb{E}[A'] \ge (1+\gamma\epsilon)\,\mathbb{E}[A]$ as claimed.

Algorithm 5.2 Implements the algorithm from Lemma 5.2.

```
1: procedure GoodRect($X, Y, A, \epsilon, d$)
2:     Compute $X' \leftarrow$ MinDegree($X, Y, A, \epsilon, \frac{1}{2}$) and $A' \leftarrow A[X', Y]$
3:     if $\mathbb{E}[A'] \geq (1 + \frac{\epsilon}{2})\,\mathbb{E}[A]$ then
4:         return GoodRect($X', Y, A', \epsilon, d$)
5:     if Sift($X', Y, A', \epsilon, d$) returns a denser rectangle $X'' \times Y'' \subseteq X' \times Y$ then
6:         return GoodRect($X'', Y'', A[X'', Y''], \epsilon, d$)
7:     return $X', Y$
```

Running Time. It is easy to check that this algorithm can be implemented in time $O(|X|\,|Y|)$: We precompute the degrees of all nodes in X, and sort X according to these degrees. Throughout we maintain the set X and the size $|A'|$. In each step we can find in constant time a node $x \in X'$ with $\deg_{A'}(x) < (1 - \epsilon)\,\mathbb{E}[A']$ if it exists (namely the node in X' with smallest degree). ☐

We remark that, while the previous lemma only guarantees the left-sided min-degree condition, it is equally possible to guarantee the condition on both sides. However, we never need this stronger statement in our upcoming proofs and therefore stick to this simpler version.

5.1 *A*-Decomposition

In this subsection we prove Theorem 3.3 (i.e., the decomposition of a *single bipartite* graph into regular subgraphs). As outlined in Section 3, the proof consists of two steps: A method to find *good rectangles* via density increments (see Lemma 5.2), and a decomposition via density decrements that repeatedly remove good rectangles (see Theorem 3.3).

Lemma 5.2 (Finding a Good Rectangle)**.** *Let $A \in \{0,1\}^{X \times Y}$, let $\epsilon \in (0,1)$ and $d \geq 1$, and assume that $\mathbb{E}[A] \geq 2^{-d}$. There is an algorithm* GoodRect(X, Y, A, ϵ, d) *computing $X^* \subseteq X, Y^* \subseteq Y$ such that:*

1. *Let $A^* = A[X^*, Y^*]$. Then A^* is $(\epsilon, 2, d)$-regular and ϵ-min-degree.*
2. *$\mathbb{E}[A^*] \geq \mathbb{E}[A]$.*
3. *$|X^*|\,|Y^*| \geq \exp(-d^3 \operatorname{poly}(\epsilon^{-1})) \cdot |X|\,|Y|$.*

The algorithm is deterministic and runs in time $n^2 \cdot \exp(d^3 \operatorname{poly}(\epsilon^{-1}))$ (where $n = |X| + |Y|$).

Proof. We start with the description of the algorithm; see Algorithm 5.2 for the pseudocode. We first apply Lemma 5.1 to compute $X' \subseteq X$ and $A' \leftarrow A[X', Y]$ (with parameters ϵ and $\gamma \leftarrow \frac{1}{2}$, Line 2). The lemma guarantees one of two cases: Either A' is ϵ-min-degree, or $\mathbb{E}[A'] \geq (1 + \frac{\epsilon}{2})\,\mathbb{E}[A]$. In the latter case we recurse on the subinstance (X', Y, A') to find a good rectangle (Lines 3 and 4). In the former case we continue and apply Theorem 3.2 to (X', Y, A') (with parameters ϵ and d). If the theorem returns a rectangle $X'' \times Y'' \subseteq X' \times Y$ with $\mathbb{E}[A[X'', Y'']] \geq (1 + \frac{\epsilon}{2})\,\mathbb{E}[A]$, then we recurse on the subinstance $(X'', Y'', A[X'', Y''])$ (Lines 5 and 6). Otherwise, Theorem 3.2 guarantees that A' is $(\epsilon, 2, d)$-regular. In this case we finally return $X^* \leftarrow X', Y^* \leftarrow Y$ (Line 7).

The correctness of Property 1 is clear: We either recurse, or return the submatrix A' that is ϵ-min-degree and $(\epsilon, 2, d)$-regular. Moreover, it is easy to prove that Property 2 holds: Lemma 5.1 states that $\mathbb{E}[A'] \geq \mathbb{E}[A]$ in either case; and if the algorithm recurses, then the density even strictly increases. It remains to prove Property 3 and to analyze the running time of this algorithm.

Algorithm 5.3 Implements the algorithm from Theorem 3.3.

```
1: procedure ADecomposition($X, Y, A, \epsilon, d$)
2:     if $\mathbb{E}[A] \le 2^{-d}$ then
3:         return $\{(X, Y, A)\}$
4:     Compute $X^*, Y^* \leftarrow$ GoodRect($X, Y, A, \epsilon, d$)
5:     return $\{(X^*, Y^*, A[X^*, Y^*])\} \cup$ ADecomposition($X, Y, A - A[X^*, Y^*], \epsilon, d$)
```

Correctness of Property 3. First assume that the algorithm does not recurse and returns X', Y. In this case, Lemma 5.1 guarantees that $|X'| \ge \frac{1}{2} \cdot |X|$, and thus $|X^*|\,|Y^*| \ge \frac{1}{2} \cdot |X|\,|Y|$. Next, consider the recursive cases. By Lemma 5.1 and Theorem 3.2, in both cases we recurse on a rectangle of size at least $\min\{\frac{1}{2}, \frac{\epsilon}{16} \cdot \mathbb{E}[A]^{2d}\} \cdot |X|\,|Y| = \frac{\epsilon}{16} \cdot \mathbb{E}[A]^{2d} \cdot |X|\,|Y|$ and with density at least $(1 + \frac{\epsilon}{2})\,\mathbb{E}[A]$. It follows by induction that the algorithm returns a rectangle of size

$$|X^*|\,|Y^*| \ge \tfrac{1}{2} \cdot \left(\tfrac{\epsilon}{16} \cdot \mathbb{E}[A]^{2d}\right)^{\log_{1+\frac{\epsilon}{2}}(\mathbb{E}[A]^{-1})} \cdot |X|\,|Y| \ge \exp(-d^3 \operatorname{poly}(\epsilon^{-1})) \cdot |X|\,|Y|,$$

where in the latter bound we used that $\log(1+\epsilon) \ge \epsilon$ for all $\epsilon \in (0, 1)$, and that initially $\mathbb{E}[A] \ge 2^{-d}$.

Running Time. The algorithm reaches recursion depth at most $\log_{1+\frac{\epsilon}{2}}(\mathbb{E}[A]^{-1}) = O(d/\epsilon)$, which causes a negligible overhead in the running time. The call to MinDegree (Lemma 5.1) takes time $O(n^2)$, and the call to Sift (Theorem 3.2) takes time $n^2 \cdot (\epsilon\,\mathbb{E}[A])^{-O(d^2)} = n^2 \cdot \exp(d^3 \operatorname{poly}(\epsilon^{-1}))$. All in all, the running time is $n^2 \cdot \exp(d^3 \operatorname{poly}(\epsilon^{-1}))$ as claimed. ☐

Theorem 3.3 (A-Decomposition). *Let $A \in \{0,1\}^{X \times Y}$, let $\epsilon \in (0, 1)$ and $d \ge 1$. There is an algorithm* ADecomposition(X, Y, A, ϵ, d) *computing tuples $\{(X_\ell, Y_\ell, A_\ell)\}_{\ell=1}^{L}$ with $X_\ell \subseteq X$, $Y_\ell \subseteq Y$, and $A_\ell \in \{0,1\}^{X_\ell \times Y_\ell}$ such that:*

1. *$A = \sum_{\ell=1}^{L} A_\ell$.*
2. *For all $\ell \in [L]$:*
   (i) $\mathbb{E}[A_\ell] \le 2^{-d}$, or
   (ii) A_ℓ is $(\epsilon, 2, d)$-regular and ϵ-min-degree.
3. *$\sum_{\ell=1}^{L} |X_\ell|\,|Y_\ell| \le (d + 2) \cdot |X|\,|Y|$.*
4. *$L \le \exp(d^3 \operatorname{poly}(\epsilon^{-1}))$ and $\min_{\ell=1}^{L} |X_\ell|\,|Y_\ell| \ge \exp(-d^3 \operatorname{poly}(\epsilon^{-1})) \cdot |X|\,|Y|$.*

The algorithm is deterministic and runs in time $n^2 \cdot \exp(d^3 \operatorname{poly}(\epsilon^{-1}))$ (where $n = |X| + |Y|$).

Proof. We start with the description of the algorithm; see Algorithm 5.3 for the pseudocode. As a first step, we compute the density $\mathbb{E}[A]$ and test whether $\mathbb{E}[A] \le 2^{-d}$. In this case we can immediately stop and return the trivial decomposition $\{(X, Y, A)\}$ (Lines 2 and 3). Otherwise, we can apply Lemma 5.2 to compute a good rectangle $X^* \times Y^* \subseteq X \times Y$. We then take $(X^*, Y^*, A[X^*, Y^*])$ as one piece of the decomposition, and recurse on the remaining graph where we remove all edges in $X^* \times Y^*$ (Lines 4 and 5; in the pseudocode by slight abuse of notation we denote the remaining graph by $A - A[X^*, Y^*]$).

It is easy to see that Property 1 holds: We either cover A entirely in the base case, or we cover some part $X^* \times Y^*$ and remove that part in the recursive call. Moreover, Property 2 easily follows from the guarantee of Lemma 5.2. It remains to prove Properties 3 and 4, and to analyze the running time.

Correctness of Property 3. For a collection of tuples $\mathcal{S}$ as returned by the algorithm, let us define

$$C(\mathcal{S}) = \sum_{(X',Y',A') \in \mathcal{S}} |X'|\,|Y'|.$$

Our goal is to prove that $C(\mathcal{S}) \leq (d+2) \cdot |X|\,|Y|$, for any input (X, Y, A), where $\mathcal{S}$ is the set returned by the algorithm. This is clear whenever $\mathbb{E}[A] \leq 2^{-d}$ (as then the algorithm returns the trivial partition $\{(X, Y, A)\}$). By induction we prove that whenever $\mathbb{E}[A] \geq 2^{-d}$ then

$$C(\mathcal{S}) \leq (d + 2 - \log(\mathbb{E}[A]^{-1})) \cdot |X|\,|Y|.$$

Let $\mathcal{S}^*$ denote the output of the recursive call $\textsc{ADecomposition}(X, Y, A - A[X^*, Y^*], \epsilon, d)$, and let δ^* denote the density of $A - A[X^*, Y^*]$. Clearly, $C(\mathcal{S}) = |X^*|\,|Y^*| + C(\mathcal{S}^*)$. If $\delta^* \leq 2^{-d}$, then $C(\mathcal{S}^*) \leq |X|\,|Y|$ and thus $C(\mathcal{S}) \leq 2|X|\,|Y|$ as claimed. Otherwise, recall that Lemma 5.2 guarantees that the density of the rectangle $X^* \times Y^*$ does not decrease, $\mathbb{E}[A[X^*, Y^*]] \geq \mathbb{E}[A]$. It follows that $\delta^* \geq \mathbb{E}[A] - \mathbb{E}[A] \cdot \frac{|X^*|\,|Y^*|}{|X|\,|Y|}$, and thus by induction:

$$\begin{aligned}
C(\mathcal{S}) &= |X^*|\,|Y^*| + C(\mathcal{S}^*) \\
&\leq |X^*|\,|Y^*| + (d + 2 - \log((\delta^*)^{-1}) \cdot |X|\,|Y| \\
&\leq |X^*|\,|Y^*| + (d + 2 - \log(\mathbb{E}[A]^{-1}) - \log((1 - \tfrac{|X^*|\,|Y^*|}{|X|\,|Y|})^{-1})) \cdot |X|\,|Y| \\
&\leq |X^*|\,|Y^*| + (d + 2 - \log(\mathbb{E}[A]^{-1}) - \tfrac{|X^*|\,|Y^*|}{|X|\,|Y|}) \cdot |X|\,|Y| \\
&= (d + 2 - \log(\mathbb{E}[A]^{-1})) \cdot |X|\,|Y|.
\end{aligned}$$

Correctness of Property 4. Note that L is the recursion depth of the algorithm. To prove that L is bounded as claimed, we argue that with every recursive call the density of the matrix decreases. Specifically, each output X^*, Y^* has size at least $|X^*|\,|Y^*| \geq \exp(-d^3 \operatorname{poly}(\epsilon^{-1})) \cdot |X| \cdot |Y|$ by Lemma 5.2, and the density of the submatrix $A^* = A[X^*, Y^*]$ satisfies $\mathbb{E}[A^*] \geq \mathbb{E}[A] \geq 2^{-d}$ (as otherwise, if $\mathbb{E}[A] \leq 2^{-d}$, the algorithm had terminated already). Thus, each recursive call reduces the density of the graph by $\mathbb{E}[A^*] \cdot \frac{|X^*|\,|Y^*|}{|X|\,|Y|} \geq 2^{-d} \cdot \exp(-d^3 \operatorname{poly}(\epsilon^{-1})) = \exp(-d^3 \operatorname{poly}(\epsilon^{-1}))$, and the algorithm necessarily terminates after $L \leq \exp(d^3 \operatorname{poly}(\epsilon^{-1}))$ recursive calls.

Running Time. We have already bounded the recursion depth in the previous paragraph, so focus on a single execution. The dominant cost is the call to $\textsc{GoodRect}$ (Lemma 5.2) which takes time $n^2 \cdot \exp(d^3 \operatorname{poly}(\epsilon^{-1}))$. The total time is $L \cdot n^2 \cdot \exp(d^3 \operatorname{poly}(\epsilon^{-1})) = n^2 \cdot \exp(d^3 \operatorname{poly}(\epsilon^{-1}))$. □

5.2 *AB*-Decomposition

We finally turn to the proof of Theorem 3.1. The outline, as discussed in Section 3, follows the previous subsection on a high-level (but differs in many more difficult technical aspects). We first devise a method to find *good cube* via density increments (see Lemma 5.3), and then derive the decomposition via density decrements that repeatedly remove good cubes (see Theorem 3.1).

Lemma 5.3 (Finding a Good Cube)**.** *Let $A \in \{0,1\}^{X \times Y}, B \in \{0,1\}^{Y \times Z}$, let $\epsilon \in (0,1), \gamma \in (0, \frac{1}{2})$ and $d \geq 1$ and assume that $\mathbb{E}[B] \geq 2^{-d}$. There is an algorithm* GoodCube$(X, Y, Z, A, B, \epsilon, \gamma, d)$ *computing sets $Y^* \subseteq Y$, $Z^* \subseteq Z$ and $\{(X_\ell, Y_\ell, Z_\ell, A_\ell)\}_{\ell=1}^L$ with $X_\ell \subseteq X$, $Y_\ell \subseteq Y^*$, $Z_\ell \subseteq Z^*$ and $A_\ell \in \{0,1\}^{X_\ell \times Y_\ell}$ such that*

1. $A[X, Y^*] = \sum_{\ell=1}^L A_\ell$.
2. *For all $\ell \in [L]$, writing $B_\ell = B[Y_\ell, Z_\ell]$:*
   - $\mathbb{E}[A_\ell] \leq 2^{-d}$, *or*
   - *A_ℓ and B_ℓ^T are both $(\epsilon, 2, d)$-regular and ϵ-min-degree.*
3. $\mathbb{E}[B[Y^*, Z^*]] \geq \mathbb{E}[B]$.
4. $\sum_{\ell=1}^L |X_\ell|\,|Y_\ell|\,|Z_\ell| \leq (d+2) \cdot |X|\,|Y^*|\,|Z^*|$
5. $\sum_{\ell=1}^L |X_\ell|\,|Y_\ell|\,|Z^* \setminus Z_\ell| \leq \gamma(d+2) \cdot |X|\,|Y^*|\,|Z^*|$.
6. $|Y^*|\,|Z^*| \geq \exp(-d^4 \gamma^{-1} \operatorname{poly}(\epsilon^{-1})) \cdot |Y|\,|Z|$.
7. $L \leq \exp(d^3 \operatorname{poly}(\epsilon^{-1}))$.

The algorithm is deterministic and runs in time $n^2 \cdot \gamma^{-1} \exp(d^3 \operatorname{poly}(\epsilon^{-1}))$ (for $n = |X| + |Y| + |Z|$).

Proof. Let us start with a description of the algorithm; for the pseudocode see Algorithm 5.4. We first call MinDegree$(Y, Z, B, \frac{\epsilon\gamma}{2}, \frac{1}{2})$ (Lemma 5.1) to compute some $Y' \subseteq Y$. Write $A' \leftarrow A[X, Y']$ and $B' \leftarrow B[Y', Z]$. Lemma 5.1 states that either B' is $\frac{\epsilon\gamma}{2}$-min-degree, or $\mathbb{E}[B'] \geq (1 + \frac{\epsilon\gamma}{4})\,\mathbb{E}[B]$. In the latter case we simply recurse on the subinstance induced by X, Y', Z (Lines 2 to 4).

Next, we run Theorem 3.3 to compute an edge decomposition $\{(X_\ell, Y_\ell, A_\ell)\}_{\ell=1}^L$ of (X, Y', A') (Line 5). The hope is that, for all pieces $\ell \in [L]$, the induced graphs $B[Y_\ell, Z]$ also satisfy the min-degree and regularity conditions, in which case we could return the decomposition without changes. To ensure both, we enumerate each $\ell \in [L]$ (Line 6). By calling MinDegree$(Z, Y_\ell, B^T[Z, Y_\ell], \epsilon, \gamma)$ (Lemma 5.1) we compute a set $Z_\ell \subseteq Z$ such that, writing $B_\ell \leftarrow B[Y_\ell, Z_\ell]$, either B_ℓ^T is ϵ-min-degree or B_ℓ has increased density. The former case is exactly the desired min-degree condition, and in the latter case we again recurse on the subinstance induced by X, Y_ℓ, Z_ℓ (Lines 7 to 9). It remains to ensure regularity. To this end we call Sift$(Z_\ell, Y_\ell, B_\ell^T, \epsilon, 2, d)$ (Theorem 3.2), which either certifies that B_ℓ^T is $(\epsilon, 2, d)$-regular, or finds $Z'' \subseteq Z_\ell$ and $Y'' \subseteq Y_\ell$ such that the density of the induced subgraph $B_\ell[Y'', Z'']$ increases. In the latter case, again, we recurse (Lines 10 and 11).

If after all these tests the algorithm has not recursed, we finally return $Y^* \leftarrow Y'$, $Z^* \leftarrow Z$ and the collection $\{(X_\ell, Y_\ell, Z_\ell, A_\ell)\}_{\ell=1}^L$ (Line 12).

Some properties of the algorithm are easy to prove. For instance, Properties 1 and 7 follow immediately from Theorem 3.3. Property 2 is easy to prove as well: Theorem 3.3 implies that for each $\ell \in [L]$, we have that $\mathbb{E}[A_\ell] \leq 2^{-d}$ or that A_ℓ is ϵ-min-degree and $(\epsilon, 2, d)$-regular. In addition, the algorithm only terminates and returns an output after certifying that, for all $\ell \in [L]$, B_ℓ^T is ϵ-min-degree and $(\epsilon, 2, d)$-regular. The other properties require more work. We start with the following claim:

Claim. *The algorithm only recurses on subgraphs of B with density at least $(1 + \frac{\epsilon\gamma}{4})\,\mathbb{E}[B]$.*

Algorithm 5.4 Implements the algorithm from Lemma 5.3.

```
1: procedure GoodCube($X, Y, Z, A, B, \epsilon, \gamma, d$)
2:     Compute $Y' \leftarrow$ MinDegree($Y, Z, B, \frac{\epsilon\gamma}{2}, \frac{1}{2}$) and let $A' \leftarrow A[X, Y']$ and $B' \leftarrow B[Y', Z]$
3:     if $\mathbb{E}[B'] \geq (1 + \frac{\epsilon\gamma}{4})\,\mathbb{E}[B]$ then
4:         return GoodCube($X, Y', Z, A', B', \epsilon, \gamma, d$)
5:     Compute $\{(X_\ell, Y_\ell, A_\ell)\}_{\ell=1}^{L} \leftarrow$ ADecomposition($X, Y', A', \epsilon, d$)
6:     for each $\ell \in [L]$ do
7:         Compute $Z_\ell \leftarrow$ MinDegree($Z, Y_\ell, B^T[Z, Y_\ell], \epsilon, \gamma$) and let $B_\ell \leftarrow B[Y_\ell, Z_\ell]$
8:         if $\mathbb{E}[B_\ell] \geq (1 + \epsilon\gamma)\,\mathbb{E}[B[Y_\ell, Z]]$ then
9:             return GoodCube($X, Y_\ell, Z_\ell, A[X, Y_\ell], B_\ell, \epsilon, \gamma, d$)
10:        if Sift($Z_\ell, Y_\ell, B_\ell^T, \epsilon, 2, d$) returns a denser rectangle $Z'' \times Y'' \subseteq Z_\ell \times Y_\ell$ then
11:            return GoodCube($X, Y'', Z'', A[X, Y''], B[Y'', Z''], \epsilon, \gamma, d$)
12:    return $Y', Z, \{(X_\ell, Y_\ell, Z_\ell, A_\ell)\}_{\ell=1}^{L}$
```

Proof. If the algorithm recurses in Lines 3 and 4, then the claim is immediate. After passing Line 4, Lemma 5.1 guarantees that the graph B' is $\frac{\epsilon\gamma}{2}$-min-degree. From this min-degree condition we know that, for any set $Y'' \subseteq Y'$, the subgraph $B[Y'', Z]$ has density at least $(1 - \frac{\epsilon\gamma}{2})\,\mathbb{E}[B]$. In particular, if the algorithm recurses in Line 4 then we recurse on a subgraph of density

$$\mathbb{E}[B_\ell] \geq (1 + \epsilon\gamma)\,\mathbb{E}[B[Y_\ell, Z]] \geq (1 + \epsilon\gamma)(1 - \tfrac{\epsilon\gamma}{2})\,\mathbb{E}[B] \geq (1 + \tfrac{\epsilon\gamma}{4})\,\mathbb{E}[B];$$

here in the last step we used that $\gamma \in (0, \frac{1}{2})$ and $\epsilon \in (0, 1)$. Similarly, if the algorithm recurses in Line 11 then the density is at least $(1 + \frac{\epsilon}{2})\,\mathbb{E}[B_\ell] \geq (1 + \frac{\epsilon}{2})\,\mathbb{E}[B[Y_\ell, Z]] \geq (1 + \frac{\epsilon\gamma}{4})\,\mathbb{E}[B]$. □

Correctness of Property 3. With the previous claim in mind we can easily conclude that Property 3 holds. If the algorithm terminates without recurring, then $\mathbb{E}[B[Y^*, Z^*]] \geq \mathbb{E}[B]$ follows directly from Lemma 5.1. If the algorithm recurses, then the statement follows by induction using that the density never decreases.

Correctness of Properties 4 and 5. Observe that Property 4 is immediate by Theorem 3.3. Furthermore, to prove Property 5 it suffices to check that $|Z^* \setminus Z_\ell| \leq \gamma |Z^*|$ for all $\ell \in [L]$. This indeed holds by Lemma 5.1.

Correctness of Property 6. First note that in the base case, if the algorithm terminates without recurring, then $Y^* = Y'$, $Z^* = Z$ and thus $|Y^*|\,|Z^*| \geq \frac{1}{2} \cdot |Y|\,|Z|$. Next consider the recursive cases. In Line 4 we possibly recurse on the subgraph $B[Y', Z]$, where $|Y'| \geq \frac{1}{2} \cdot |Y|$. In Line 9 we possibly recurse on the subgraph $B[Y_\ell, Z_\ell]$, where $|Y_\ell| \geq \exp(-d^3 \operatorname{poly}(\epsilon^{-1})) \cdot |Y'|$ (by Theorem 3.3) and $|Z_\ell| \geq (1 - \gamma) \cdot |Z| \geq \frac{1}{2} \cdot |Z|$ (by Lemma 5.1). Finally, in Line 11 we possibly recurse on a subgraph $B[Y'', Z'']$ of size at least $|Y''|\,|Z''| \geq \frac{\epsilon}{16} \cdot 2^{-2d^2} \cdot |Y_\ell|\,|Z_\ell|$ (by Theorem 3.2). In either case, the size of the subgraph is at least $\exp(-d^3 \operatorname{poly}(\epsilon^{-1})) \cdot |X|\,|Y|$. Recall further that by the claim, the density increases multiplicatively by $1 + \frac{\epsilon\gamma}{4}$ with every recursive call. Therefore, and since the initial density is $\mathbb{E}[B] \geq 2^{-d}$, the algorithm returns sets Y^*, Z^* with size

$$\exp(-d^3 \operatorname{poly}(\epsilon^{-1}))^{\log_{1+\frac{\epsilon\gamma}{4}}(\mathbb{E}[B]^{-1})} \cdot |Y|\,|Z| \geq \exp(-d^4 \gamma^{-1} \operatorname{poly}(\epsilon^{-1})) \cdot |Y|\,|Z|.$$

Here we used that $\log(1 + \epsilon) \geq \epsilon$ for all $\epsilon \in (0, 1)$.

Running Time. We finally analyze the running time of the algorithm. As argued before, the recursion depth is bounded by $O(d\epsilon^{-1}\gamma^{-1})$. In each execution of Algorithm 5.4 we call ADecomposition once which takes time $n^2 \cdot \exp(d^3 \operatorname{poly}(\epsilon^{-1}))$. In addition we call Sift L times taking time $L \cdot n^2 \cdot \exp(d^2 \operatorname{poly}(\epsilon^{-1}))$. The total running time becomes $n^2 \cdot \gamma^{-1} \exp(d^3 \operatorname{poly}(\epsilon^{-1}))$. ☐

Theorem 3.1 (AB-Decomposition). *Let $A \in \{0,1\}^{X \times Y}$, $B \in \{0,1\}^{Y \times Z}$, let $\epsilon \in (0,1)$ and $d \geq 1$. There is an algorithm* ABDecomposition$(X, Y, Z, A, B, \epsilon, d)$ *that computes a collection of tuples $\{(X_k, Y_k, Z_k, A_k, B_k)\}_{k=1}^K$, where $X_k \subseteq X$, $Y_k \subseteq Y$, $Z_k \subseteq Z$, $A_k \in \{0,1\}^{X_k \times Y_k}$, $B_k \in \{0,1\}^{Y_k \times Z_k}$ such that*

1. *$AB = \sum_{k=1}^K A_k B_k$.*
2. *For all $k \in [K]$:*
   (i) $\mathbb{E}[A_k] \leq 2^{-d}$ or $\mathbb{E}[B_k] \leq 2^{-d}$, or
   (ii) A_k and B_k^T are both $(\epsilon, 2, d)$-regular and ϵ-min-degree.
3. *$\sum_{k=1}^K |X_k|\,|Y_k|\,|Z_k| \leq 2(d+2)^2 \cdot |X|\,|Y|\,|Z|$.*
4. *$K \leq \exp(d^7 \operatorname{poly}(\epsilon^{-1}))$.*

The algorithm is deterministic and runs in time $n^2 \cdot \exp(d^7 \operatorname{poly}(\epsilon^{-1}))$ (where $n = |X| + |Y| + |Z|$).

Proof. We start with the description of the algorithm; see Algorithm 5.5 for the pseudocode. Throughout we assume an additional input parameter $0 \leq h \leq d$ which acts somewhat as the recursion depth of the algorithm. For the initial call, we set $h = 0$.

The algorithm has two bases cases. If B is sufficiently sparse, $\mathbb{E}[B] \leq 2^{-d}$, then we return the trivial decomposition $\{(X, Y, Z, A, B)\}$ (Lines 2 and 3). Moreover, if the algorithm has reached recursion depth $h = d$, then we return a trivial decomposition of size at most 2^d. Specifically, we split B into submatrices $B_1, \dots, B_{2^d} \in \{0,1\}^{Y \times Z}$ each of density at most 2^{-d} and return the decomposition $\{(X, Y, Z, A, B_i)\}_{i=1}^{2^d}$ (Lines 4 to 6).

Otherwise, run GoodCube$(X, Y, Z, A, B, \epsilon, \gamma, d)$ (Lemma 5.3) with the parameter $\gamma = \frac{1}{2(d+2)^2}$. This output consists of sets $Y^* \subseteq Y$, $Z^* \subseteq Z$ and a set $\{(X_\ell, Y_\ell, Z_\ell, A_\ell)\}_{\ell=1}^L$ (Line 7). For each $\ell \in [L]$ we write $B_\ell \leftarrow B[Y_\ell, Z_\ell]$ and $B'_\ell \leftarrow B[Y_\ell, Z^* \setminus Z_\ell]$. We then recursively compute, for each $\ell \in [L]$, the decomposition $\mathcal{S}_\ell$ of $(X_\ell, Y_\ell, Z^* \setminus Z_\ell, A_\ell, B'_\ell)$ (Lines 8 to 10). Moreover, we recursively compute the decomposition $\mathcal{S}^*$ of $(X, Y, Z, A, B - B[Y^*, Z^*])$ (Line 11; here, we denote by $B - B[Y^*, Z^*]$ the matrix obtained from B after zeroing out the entries in $B[Y^*, Z^*]$). Finally, we return $\bigcup_{\ell=1}^L \{(X_\ell, Y_\ell, Z_\ell, A_\ell, B_\ell)\} \cup \bigcup_{\ell=1}^L \mathcal{S}_\ell \cup \mathcal{S}^*$ (Line 12).

Correctness of Property 1. For a set $\mathcal{S}$ as returned by the algorithm, we write

$$\Sigma(\mathcal{S}) = \sum_{(X',Y',Z',A',B') \in \mathcal{S}} A'B',$$

where, as in the theorem statement, we interpret each term $A'B'$ in the sum as an $X \times Z$-matrix by extending with zeros. The goal is to prove that $\Sigma(\mathcal{S}) = AB$, where $\mathcal{S}$ is the set returned by the

Algorithm 5.5 Implements the algorithm from Theorem 3.1.

```
1: procedure ABDecomposition'($X, Y, Z, A, B, \epsilon, d, h$)
2:     if $\mathbb{E}[B] \le 2^{-d}$ then
3:         return $\{(X, Y, Z, A, B)\}$
4:     if $h = d$ then
5:         Arbitrarily partition $B$ into submatrices $B_1, \dots, B_{2^d} \in \{0,1\}^{Y \times Z}$ of density at most $2^{-d}$
6:         return $\{(X, Y, Z, A, B_i)\}_{i=1}^{2^d}$
7:     Compute $Y^*, Z^*, \{(X_\ell, Y_\ell, Z_\ell, A_\ell)\}_{\ell=1}^{L} \leftarrow$ GoodCube($X, Y, Z, A, B, \epsilon, \frac{1}{2(d+2)^2}, d$)
8:     for each $\ell \in [L]$ do
9:         Let $B_\ell \leftarrow B[Y_\ell, Z_\ell]$ and $B'_\ell \leftarrow B[Y_\ell, Z^* \setminus Z_\ell]$
10:        Compute $\mathcal{S}_\ell \leftarrow$ ABDecomposition'($X_\ell, Y_\ell, Z^* \setminus Z_\ell, A_\ell, B'_\ell, \epsilon, d, h+1$)
11:    Compute $\mathcal{S}^* \leftarrow$ ABDecomposition'($X, Y, Z, A, B - B[Y^*, Z^*], \epsilon, d, h$)
12:    return $\bigcup_{\ell=1}^{L} \{(X_\ell, Y_\ell, Z_\ell, A_\ell, B_\ell)\} \cup \bigcup_{\ell=1}^{L} \mathcal{S}_\ell \cup \mathcal{S}^*$

13: procedure ABDecomposition($X, Y, Z, A, B, \epsilon, d$)
14:     return ABDecomposition'($X, Y, Z, A, B, \epsilon, d, 0$)
```

algorithm on input (X, Y, Z, A, B). This is clear in both base cases. So assume that the algorithm recurses. Then by induction:

$$\begin{aligned}
\Sigma(\mathcal{S}) &= \sum_{\ell=1}^{L} A_\ell B_\ell + \sum_{\ell=1}^{L} \Sigma(\mathcal{S}_\ell) + \Sigma(\mathcal{S}^*) \\
&= \sum_{\ell=1}^{L} A_\ell B_\ell + \sum_{\ell=1}^{L} A_\ell B'_\ell + AB - A[X, Y^*]B[Y^*, Z^*] \\
&= \sum_{\ell=1}^{L} A_\ell B[Y_\ell, Z^*] + AB - A[X, Y^*]B[Y^*, Z^*] \\
&= A[X, Y^*]B[Y^*, Z^*] + AB - A[X, Y^*]B[Y^*, Z^*] \\
&= AB;
\end{aligned}$$

here, in the second-to-last step we have applied Property 1 of Lemma 5.3.

Correctness of Property 2. In both base cases we return partitions in which all parts B_k are sparse, $\mathbb{E}[B_k] \le 2^{-d}$. In the recursive case the output consists of the union of three different sets: For each $(X_\ell, Y_\ell, Z_\ell, A_\ell, B_\ell)$ the claim follows from Property 2 of Lemma 5.3, and for each element in $\mathcal{S}_\ell$ or $\mathcal{S}^*$ the claim holds by induction.

Correctness of Property 3. For a collection $\mathcal{S}$ of tuples as returned by the algorithm, let us write

$$C(\mathcal{S}) = \sum_{(X', Y', Z', A', B') \in \mathcal{S}} |X'|\,|Y'|\,|Z'|.$$

Our goal is to prove that $C(\mathcal{S}) \leq 2^{h+1} \cdot (d+2)^2 \cdot |X|\,|Y|\,|Z|$, where $\mathcal{S}$ is the output of our algorithm with inputs $(X, Y, Z, A, B, \epsilon, d, h)$. In the sparse case, if $\mathbb{E}[B] \leq 2^{-d}$, then we return the trivial decomposition with $C(\mathcal{S}) = |X|\,|Y|\,|Z|$. We prove by induction that otherwise the following bound applies:

$$C(\mathcal{S}) \leq 2^{h+1} \cdot (d+2)(d+1-\log(\mathbb{E}[B]^{-1})) \cdot |X|\,|Y|\,|Z|$$

Clearly, this upper bound is true if $h = d$, so suppose that $h < d$. Then the algorithm recurses and we report $\mathcal{S}$ with $C(\mathcal{S}) = \sum_{\ell=1}^{L} |X_\ell|\,|Y_\ell|\,|Z_\ell| + \sum_{\ell=1}^{L} C(\mathcal{S}_\ell) + C(\mathcal{S}^*)$. In the following we bound these three contributions individually.

For the first contribution we readily exploit Property 4 of Lemma 5.3:

$$\sum_{\ell=1}^{L} |X_\ell|\,|Y_\ell|\,|Z_\ell| \leq (d+2) \cdot |X|\,|Y^*|\,|Z^*| \leq 2^h \cdot (d+2) \cdot |X|\,|Y^*|\,|Z^*|. \tag{1}$$

For the second contribution we exploit Property 5 of Lemma 5.3:

$$\begin{aligned}\sum_{\ell=1}^{L} C(\mathcal{S}_\ell) &\leq \sum_{\ell=1}^{L} 2^{h+1} \cdot (d+2)^2 \cdot |X_\ell|\,|Y_\ell|\,|Z^* \setminus Z_\ell| \\ &\leq 2^{h+1} \cdot \gamma \cdot (d+2)^3 \cdot |X|\,|Y^*|\,|Z^*| \\ &\leq 2^h \cdot (d+2) \cdot |X|\,|Y^*|\,|Z^*|. \end{aligned} \tag{2}$$

Here, in the last step, we have used our choice of $\gamma = \frac{1}{2(d+2)^2}$. For the third contribution, we distinguish two cases. If the density δ^* of the subgraph $B - B[Y^*, Z^*]$ is smaller than 2^{-d}, then $C(\mathcal{S}^*) \leq |X|\,|Y|\,|Z|$. Otherwise, we have $\delta^* \leq \mathbb{E}[B] \cdot (1 - \frac{|Y^*|\,|Z^*|}{|Y|\,|Z|})$ (since $\mathbb{E}[B[Y^*, Z^*]] \geq \mathbb{E}[B]$ by Property 3 of Lemma 5.3) and therefore:

$$\begin{aligned} C(\mathcal{S}^*) &\leq 2^{h+1} \cdot (d+2)(d+1-\log(\mathbb{E}[B]^{-1} \cdot (1 - \tfrac{|Y^*|\,|Z^*|}{|Y|\,|Z|})^{-1})) \cdot |X|\,|Y|\,|Z| \\ &\leq 2^{h+1} \cdot (d+2)(d+1-\log(\mathbb{E}[B]^{-1}) + \log(1 - \tfrac{|Y^*|\,|Z^*|}{|Y|\,|Z|})) \cdot |X|\,|Y|\,|Z| \\ &\leq 2^{h+1} \cdot (d+2)(d+1-\log(\mathbb{E}[B]^{-1}) - \tfrac{|Y^*|\,|Z^*|}{|Y|\,|Z|}) \cdot |X|\,|Y|\,|Z| \\ &\leq 2^{h+1} \cdot (d+2)(d+1-\log(\mathbb{E}[B]^{-1})) \cdot |X|\,|Y|\,|Z| - 2^{h+1} \cdot (d+2) \cdot |X|\,|Y^*|\,|Z^*|. \end{aligned} \tag{3}$$

Summing over all three contributions (1), (2) and (3) yields the claimed bound on $C(\mathcal{S})$.

Correctness of Property 4. Let $M = \exp(d^4 \gamma^{-1} \operatorname{poly}(\epsilon^{-1}))$ be such that $|Y^*|\,|Z^*| \geq |Y|\,|Z| \,/\, M$ for the sets Y^*, Z^* returned by Lemma 5.3. Moreover, let $L = \exp(d^3 \operatorname{poly}(\epsilon^{-1}))$ be as in Lemma 5.3. We prove by induction that

$$K = |\mathcal{S}| \leq 2^d M \cdot (2^d M L + L)^{d+1-h} \cdot \mathbb{E}[B],$$

where $\mathcal{S}$ is the set returned by the algorithm. This bound is easily verified in the two base cases. If the algorithm recurses instead then $|\mathcal{S}| \leq L + \sum_{\ell=1}^{L} |\mathcal{S}_\ell| + |\mathcal{S}^*|$. By induction we can bound

$$|\mathcal{S}_\ell| \leq 2^d M \cdot (2^d M L + L)^{d-h},$$

and

$$\begin{aligned}|\mathcal{S}^*| &\leq 2^d M \cdot (2^d ML + L)^{d+1-h} \cdot (\mathbb{E}[B] - \mathbb{E}[B[Y^*, Z^*]] \cdot \tfrac{|Y^*|\,|Z^*|}{|Y|\,|Z|}) \\ &\leq 2^d M \cdot (2^d ML + L)^{d+1-h} \cdot \big(\mathbb{E}[B] - \mathbb{E}[B[Y^*, Z^*]] \cdot \tfrac{1}{M}\big) \\ &\leq 2^d M \cdot (2^d ML + L)^{d+1-h} \cdot \big(\mathbb{E}[B] - \tfrac{1}{2^d M}\big),\end{aligned}$$

using in the last step that $\mathbb{E}[B[Y^*, Z^*]] \geq \mathbb{E}[B]$ by Property 3 of Lemma 5.3. Combining these bounds, it follows that

$$\begin{aligned}|\mathcal{S}| &\leq L + L \cdot 2^d M \cdot (2^d ML + L)^{d-h} + 2^d M \cdot (2^d ML + L)^{d+1-h} \cdot (\mathbb{E}[B] - \tfrac{1}{2^d M}) \\ &\leq (2^d ML + L)^{d+1-h} - (2^d ML + L)^{d+1-h} + 2^d M \cdot (2^d ML + L)^{d+1-h} \cdot \mathbb{E}[B] \\ &= 2^d M \cdot (2^d ML + L)^{d+1-h} \cdot \mathbb{E}[B].\end{aligned}$$

In the end we plug in values for L and M to obtain $|S| \leq \exp(d^5 \gamma^{-1} \operatorname{poly}(\epsilon^{-1})) = \exp(d^7 \operatorname{poly}(\epsilon^{-1}))$, as stated.

Running Time. From the previous consideration we also learn that the number of recursive calls is bounded by $\exp(d^7 \operatorname{poly}(\epsilon^{-1}))$. In each recursive call, the dominant step is to call GoodCube in time $n^2 \cdot \exp(d^3 \operatorname{poly}(\epsilon^{-1}))$. Hence, the total time is $n^2 \cdot \exp(d^7 \operatorname{poly}(\epsilon^{-1}))$. □

6 Boolean Matrix Multiplication and Triangle Detection

In this section we formally derive our efficient algorithm for Boolean Matrix Multiplication from the regularity decompositions developed in the previous sections. Our algorithm relies on the following fine-grained reduction from BMM to Triangle Detection due to Vassilevska Williams and Williams [72]:

Lemma 6.1 (Boolean Matrix Multiplication to Triangle Detection, [72])**.** *If Triangle Detection is in time $O(n^3/f(n))$ (for some nondecreasing function $f(n)$), then Boolean Matrix Multiplication is in time $O(n^3/f(n^{1/3}))$.*

Theorem 6.2 (Triangle Detection)**.** *There is a deterministic combinatorial detecting whether a graph contains a triangle in time $n^3/2^{\Omega(\sqrt[7]{\log n})}$.*

Proof. We assume without loss of generality that the input graph is tripartite, (X, Y, Z, A, B, C). Let $\epsilon = \frac{1}{160}$ and let $d \geq 1$ be a parameter to be determined later. Using Theorem 3.1 we decompose (X, Y, Z, A, B) into pieces $\{(X_k, Y_k, Z_k, A_k, B_k)\}_{k=1}^K$. Further, write $C_k = C[X_k, Z_k]$. For each piece we distinguish two cases:

- If $\mathbb{E}[A_k] \leq 2^{-d}$ or $\mathbb{E}[B_k] \leq 2^{-d}$ or $\mathbb{E}[C_k] \leq 2^{-\epsilon d/2}$, then search in $(X_k, Y_k, Z_k, A_k, B_k, C_k)$ for a triangle. This step takes time $O(|X_k|\,|Y_k|\,|Z_k| \,/\, 2^{\epsilon d/2})$ by exploiting the respective sparseness.
- Otherwise, simply return "yes".

For the correctness first observe that there is a triangle in the given graph if and only if there exists $(x, z) \in X \times Z$ with $(AB)(x, z) \geq 1$ and $C(x, z) = 1$. Since Property 1 of Theorem 3.1 guarantees that $AB = \sum_k A_k B_k$, there is a triangle in the original graph if and only if there is some $k \in [K]$ and $(x, z) \in X_k \times Z_k$ with $(A_k B_k)(x, z) \geq 1$ and $C_k(x, z) = 1$. The correctness of the first case is thus clear. But it remains to argue that if $\mathbb{E}[A_k] > 2^{-d}$ and $\mathbb{E}[B_k] > 2^{-d}$ and $\mathbb{E}[C_k] > 2^{-\epsilon d/2}$, then there exists a triangle in $(X_k, Y_k, Z_k, A_k, B_k, C_k)$. Indeed, by Property 2 of Theorem 3.1 and the first two assumptions, we have that A_k and B_k^T are $(\epsilon, 2, d)$-regular and ϵ-min-degree. In this case, Theorem 2.1 implies that $A \circ B$ is $(\mathbb{E}[A]\,\mathbb{E}[B], 80\epsilon, 2^{-\epsilon d/2})$-uniform. By definition, this means that not more than a $2^{-\epsilon d/2}$-fraction of the entries in $A \circ B$ do not lie in the range $[(1-80\epsilon)\,\mathbb{E}[A]\,\mathbb{E}[B], (1+80\epsilon)\,\mathbb{E}[A]\,\mathbb{E}[B]] = [\frac{1}{2}\,\mathbb{E}[A]\,\mathbb{E}[B], \frac{3}{2}\,\mathbb{E}[A]\,\mathbb{E}[B]]$. In particular, (since we have $\mathbb{E}[A], \mathbb{E}[B] > 0$) it follows that at most a $2^{-\epsilon d/2}$-fraction of the entries in $A \circ B$ are nonzero. Using finally that $\mathbb{E}[C_k] > 2^{-\epsilon d/2}$, we conclude that there exists some common entry $(x, z) \in X_k \times Z_k$ where both $(AB)(x, z) \geq 1$ and $C(x, z) = 1$.

Let us finally analyze the running time. Detecting a triangle in each sparse sub-instance takes time

$$\sum_{k=1}^{K} \frac{|X_k|\,|Y_k|\,|Z_k|}{2^{\Omega(d)}} \leq \frac{|X|\,|Y|\,|Z| \cdot 2(d+2)^2}{2^{\Omega(d)}} = \frac{|X|\,|Y|\,|Z|}{2^{\Omega(d)}},$$

using Property 3 of Theorem 3.1. Furthermore, precomputating the regularity decomposition takes time $n^2 \cdot \exp(d^7 \operatorname{poly}(\epsilon^{-1})) = n^2 \cdot 2^{O(d^7)}$. This running is optimized by picking $d = \Theta(\sqrt[7]{\log n})$, where the constant is sufficiently small such that the preprocessing time becomes $O(n^{2.1})$, say. For this choice, the total running time is indeed $n^3/2^{\Omega(d)} = n^3/2^{\Omega(\sqrt[7]{\log n})}$. □

Our main Theorem 1.2 is immediate by combining Lemma 6.1 and Theorem 6.2.

7 Triangle Enumeration

In this section we give an improved algorithm for enumerating triangles in graphs, based on our previous decomposition theorems:

Theorem 1.3 (Triangle Enumeration Algorithm)**.** *There is a deterministic algorithm that preprocesses a given graph in time $n^3/(\log n)^6 \cdot (\log\log n)^{O(1)}$ and then enumerates all triangles with constant delay.*

We make an important distinction: A *triangle listing* algorithm receives as input a graph and returns as output a list of its t triangles—here, we care about triangle listing algorithms with running times of the form $O(n^3/f(n) + t)$. A *triangle enumeration* algorithm first preprocesses a graph in time $O(n^3/f(n))$. Afterwards, it can enumerate all triangles in the graph with constant delay (i.e., upon query, the algorithm spends time $O(1)$ to report the next triangle). For the majority of this section we work with triangle listing algorithms, but in Section 7.3 we show that both types are equivalent in our context.[12]

We structure the remainder of this section as follows: We quickly give the main idea of our algorithm in Section 7.1, with details following in Section 7.2. Finally, in Section 7.4 we include a proof that further improvements to our enumeration algorithm would entail a 3-SUM-algorithm that is faster than what is currently known.

[12]We note that there has been work on developing triangle listing algorithms in time $O(n^3/f(n) + t \cdot g(n))$, i.e., with a super-constant "per-triangle" cost $g(n)$. For this setting, algebraic fast matrix multiplication turns out to be useful and yields an algorithm in time $O(n^{2.3716} + t^{0.4782} \cdot n^{1.5655})$ [17]. Due to the super-constant per-triangle cost, however, this algorithm does not lead to nontrivial constant-delay enumeration algorithms.

7.1 Triangle Listing—The Idea

In this short overview we give the main intuition behind our algorithm. We remark that all non-trivial algorithms for triangle enumeration are based on the Four-Russians technique [12]. The following lemma states a stronger version for *sparse* graphs that has implicitly appeared in [13, 23]:

Lemma 7.1 (Four-Russians)**.** *Let $G = (X, Y, Z, A, B, C)$ be a tripartite graph. There is a deterministic algorithm that lists all t triangles in G in time*

$$O\left(n^{2.3} + \frac{|X|\,|Y|\,|Z|}{(\log n)^{100}} + \frac{|X|\,|Y|\,|Z| \cdot \mathbb{E}[A \circ B] \cdot (\log\log n)^2}{(\log n)^2} + t\right)$$

(where $n \geq |X| + |Y| + |Z|$).

Without further assumptions on the graph we can only use the trivial bound $\mathbb{E}[A \circ B] \leq 1$, which recovers the original improvement of shaving two log-factors. Note that all subsequent works that score more log-shaves nevertheless still rely on this lemma.

To understand how we arrive at our improvement of nearly six log-shaves, and why six log-shaves appears to be a *right* answer, suppose that the input graph is random (in the sense that it includes each edge independently with some probability δ). We claim that by trivial means, Lemma 7.1 now yields an improvement by nearly six log-factors. There are two cases: If the graph is sparse, $\delta \leq \frac{(\log\log n)^2}{(\log n)^2}$, then

$$\mathbb{E}[A \circ B] \approx \mathbb{E}[A]\,\mathbb{E}[B] \approx \delta^2 \leq \frac{(\log\log n)^4}{(\log n)^4},$$

and the improvement is immediate. On the other hand, consider the dense case, $\delta \geq \frac{(\log\log n)^2}{(\log n)^2}$. Intuitively, we exploit that whenever the graph is sufficiently dense, then the number of triangles dominates the running time in Lemma 7.1. Specifically, the number of triangles in the graph is $t \approx \delta^3 \cdot n^3$ (assuming that the graph is random), and therefore the running time of Lemma 7.1 is bounded by

$$O\left(\frac{n^3 \cdot \delta^2 \cdot (\log\log n)^2}{(\log n)^2} + t\right) = O\left(\frac{t \cdot (\log\log n)^2}{\delta \cdot (\log n)^2} + t\right) = O(t).$$

Of course, we cannot assume that the graph is perfectly random. Our natural approach is to apply our regularity decomposition to decompose the given graph into regular pieces that behave randomly (with respect to the quantity $\mathbb{E}[A \circ B]$ and the number of triangles t). The details are significantly more complicated though, as we cannot assume that the edges C behave regularly.

7.2 Triangle Listing—The Details

We now give the details of our algorithm, starting with a proof of the Four-Russians lemma. Recall that we work over a Word RAM model with word size $\Theta(\log n)$. In the proof of Lemma 7.1 we specifically use that this model allows to construct a length-$n^{0.1}$ array that can be accessed *in constant time* via keys of length $0.1 \log n$.

Proof of Lemma 7.1. The goal is to give an algorithm that lists all triangles in a given tripartite graph $G = (X, Y, Z, A, B, C)$. We use the following notation: For a node $y \in Y$, let $N_A(y) \subseteq X$ and $N_B(y) \subseteq Z$ denote the sets of neighbors of y, respectively. Let $s = \lfloor \log n \rfloor^{100}, r = \lfloor \frac{\log n}{1000 \log\log n} \rfloor$, and consider the following steps:

1. We arbitrarily partition X into $I = \lceil |X|/s \rceil$ groups $X_1, \dots, X_I$ of size at most s; similarly partition Z into $J = \lceil |Z|/s \rceil$ groups $Z_1, \dots, Z_J$ of size at most s.

2. We enumerate each tuple (i, j, S, T) where $i \in [I]$, $j \in [J]$ and where $S \subseteq X_i$, $T \subseteq Z_j$ have size $|S|, |T| \leq r$. Note that
$$\log \binom{s}{r} \leq r \log s \leq \frac{\log n}{1000 \log \log n} \cdot 100 \log \log n = \frac{\log n}{10};$$
therefore, there are at most $n \cdot n \cdot n^{0.1} \cdot n^{0.1} = n^{2.2}$ such tuples (i, j, S, T). Moreover, we can encode each such tuple in $O(\log n)$ bits which takes $O(1)$ machine words.[13] For each tuple (i, j, S, T) we prepare a list of all edges in $C[S, T]$ and store a pointer (of constant word size) to this list.

3. Next, we enumerate each tuple (y, i, j) where $y \in Y$, $i \in [I]$ and $j \in [J]$. We partition the set $N_A(y) \cap X_i$ (i.e., the set of neighbors of y in X_i) arbitrarily into subsets S of size r (plus possibly one subset of size less than r); let $\mathcal{S}(y, i)$ denote the resulting partition. Similarly, we partition $N_B(y) \cap Z_j$ into subsets of size at most r (plus possibly one subset of size less than r); let $\mathcal{T}(y, j)$ denote the resulting partition. Now, for each pair $S \in \mathcal{S}(y, i), T \in \mathcal{T}(y, j)$ we query list of edges associated to the tuple (i, j, S, T). We enumerate each edge (x, z) in this list associated to (i, j, S, T) and store the triangle (x, y, z).

It is easy to verify that this algorithm lists all triangles in G. Let us focus on the running time. Step 1 runs in negligible time $O(n)$. In Step 2 we enumerate at most $n^{2.2}$ tuples (i, j, S, T), and for each such tuple we spend time at most $O(s^2)$ to prepare the list of edges in $C[S, T]$. The total time of this step is $O(n^{2.2} \cdot s^2)$ which we loosely bound by $O(n^{2.3})$. In Step 3 we enumerate all $|Y| \cdot I \cdot J$ tuples (y, i, j). For each such tuple, we spend time $O(s)$ to prepare the sets $\mathcal{S}(y, i)$ and $\mathcal{T}(y, j)$. Afterwards, we spend time $O(|\mathcal{S}(y, i)| \cdot |\mathcal{T}(y, j)|)$ plus the time to list all triangles. This listing cost is linear in t, so the running time of Step 3 is thus bounded by

$$\begin{aligned}
&O\left(\sum_{y \in Y} \sum_{\substack{i \in [I] \\ j \in [J]}} (s + |\mathcal{S}(y, i)| \cdot |\mathcal{T}(y, j)|) + t \right) \\
&\quad = O\left(\sum_{y \in Y} \sum_{\substack{i \in [I] \\ j \in [J]}} \left(s + \frac{|N_A(y) \cap X_i|}{r} \cdot \frac{|N_B(y) \cap Z_j|}{r} \right) + t \right) \\
&\quad = O\left(\sum_{y \in Y} \left(IJs + \frac{|N_A(y)| \cdot |N_B(y)|}{r^2} \right) + t \right) \\
&\quad = O\left(\frac{|X|\,|Y|\,|Z|}{s} + \frac{|X|\,|Y|\,|Z| \cdot \mathbb{E}[A \circ B]}{r^2} + t \right).
\end{aligned}$$

The time bound from the lemma statement follows by plugging in the chosen parameters s and r. $\square$

[13]Of course, the exact bit representation matters. The easiest option here is to represent each set S as a (sorted) list of its at most r elements. As each element can be represented using $\log s$ bits, this representation indeed takes $r \log s \leq \frac{1}{10} \log n$ bits in total.

Theorem 7.2 (Triangle Listing)**.** *There is a deterministic algorithm that lists all t triangles in a given graph in time $O(n^3 / (\log n)^6 \cdot (\log\log n)^{O(1)} + t)$.*

Proof. Let $G = (X, Y, Z, A, B, C)$ be a given tripartite graph. We design a recursive algorithm that lists all triangles in G. To this end, we maintain one (global) list of triangles and each recursive call of the algorithm appends triangles to this list. Let n denote the total number of nodes in the original graph G (at the top level of the recursion), and let $\gamma, \delta, \epsilon \in (0, 1)$ and $d \geq 4/\epsilon$ be parameters to be determined later.

The first step is to call Theorem 3.1 on input $(X, Y, Z, A, B, \epsilon, d)$ to compute a decomposition $\{(X_k, Y_k, Z_k, A_k, B_k)\}_{k=1}^K$. For each $k \in [K]$, we write for convenience $C_k = C[X_k, Z_k]$ and $G_k = (X_k, Y_k, Z_k, A_k, B_k, C_k)$. By the guarantee of Theorem 3.1 this decomposition partitions the set of triangles in G, and thus the remaining goal is to separately list all triangles in the graphs G_k. To this end, for each $k \in [K]$, we distinguish the following cases:

1. *If* $\mathbb{E}[A_k] \leq 2^{-\epsilon d/4}$ *or* $\mathbb{E}[B_k] \leq 2^{-\epsilon d/4}$*:* List all triangles in G_k in time $O(|X_k|\,|Y_k|\,|Z_k| / 2^{\epsilon d/4})$.

2. *If* $\mathbb{E}[A_k] \leq \delta$ *and* $\mathbb{E}[B_k] \leq \delta$*:* List all triangles in G_k by Lemma 7.1.

3. *If* $\mathbb{E}[B_k] \geq \delta$*:* We further subdivide the graph G_k based on the approximate degrees in X_k with respect to Z_k. Specifically, let $L = \lceil \epsilon d/2 \rceil$ and split X_k into buckets $X_{k,1}, \ldots, X_{k,L}$ defined by

$$X_{k,\ell} = \{x \in X_k : 2^{-\ell} < \deg_{C_k}(x) \leq 2^{-\ell+1}\} \quad (\text{for } 1 \leq \ell < L),$$
$$X_{k,L} = \{x \in X_k : \deg_{C_k}(x) \leq 2^{-L+1}\}.$$

   For each $\ell \in [L]$, we define the matrix $C_{k,i} \in \{0,1\}^{X_k \times Z_k}$ as the submatrix of C_k obtained by zeroing out all rows not in X_i. (That is, $C_{k,i}[x, z] = C_k[x, z]$ if $x \in X_i$ and $C_k[x, z] = 0$ otherwise.) Writing $G_{k,\ell} = (X_k, Y_k, Z_k, A_k, B_k, C_{k,\ell})$, our remaining goal is to list the disjoint union of triangles in the graphs $G_{k,\ell}$. For each $\ell \in [L]$, we instead distinguish the following three subcases:

   3.1 *If* $\mathbb{E}[C_{k,\ell}] \leq 2^{-L+1}$*:* List all triangles in $G_{k,\ell}$ in time $O(|X_k|\,|Y_k|\,|Z_k| / 2^L)$.

   3.2 *If* $|X_{k,\ell}| < \gamma |X_k|$*:* Recurse on $(X_{k,\ell}, Y_k, Z_k, A_k[X_{k,\ell}, Y_k], B_k, C_k[X_{k,\ell}, Z_k])$. (It is easy to check that the triangles in this graph are exactly the triangles in $G_{k,\ell}$.)

   3.3 *If* $\mathbb{E}[C_{k,\ell}] > 2^{-L+1}$ *and* $|X_{k,\ell}| \geq \gamma |X_k|$*:* List all triangles in $(Y_k, X_k, Z_k, A_k^T, C_{k,\ell}, B_k)$ by Lemma 7.1. (It is easy to check that the triangles in this graph are in one-to-one correspondence to the triangles in $G_{k,\ell}$.) Note that we have exchanged the node and edge sets such that Lemma 7.1 benefits from minimizing $\mathbb{E}[A_k^T \circ C_{k,\ell}]$.

4. If $\mathbb{E}[A_k] \geq \delta$: This case is symmetric to the previous case. More precisely, we can reduce to the previous case by considering instead the graph $(Z_k, Y_k, X_k, B_k^T, A_k^T, C_k^T)$ whose triangles are clearly in one-to-one correspondence with those in G_k.

Finally, we add one more rule to the algorithm: As soon as we reach recursion depth H (for some parameter H to be determined), we simply solve the instance by brute-force in time $O(|X|\,|Y|\,|Z|)$. This completes the description of the algorithm. The correctness should be clear from the in-text explanations.

Running Time. It remains to bound the running time. For simplicity, we already fix all the parameters here, and then analyze the cases individually:

$$\begin{aligned}
\epsilon &= \frac{1}{160},\\
d &= \lceil 64000 \log\log n \rceil,\\
\gamma &= \frac{1}{8L(d+2)^2} = \Theta\bigg(\frac{1}{(\log\log n)^3}\bigg),\\
\delta &= \frac{(\log\log n)^2}{\gamma(\log n)^2} = \Theta\bigg(\frac{(\log\log n)^5}{(\log n)^2}\bigg),\\
H &= \left\lceil \log\frac{(\log n)^6}{(\log\log n)^{14}} \right\rceil = \Theta(\log\log n).
\end{aligned}$$

Claim 7.3 (Cases 1 and 3.1)**.** *The total running time of Cases 1 and 3.1 is* $O(|X|\,|Y|\,|Z|/(\log n)^{99})$*.*

Proof. The algorithm deals with Cases 1 and 3.1 in time $O(|X_k|\,|Y_k|\,|Z_k|/2^{\epsilon d/4})$, which, by our choice of the parameters ϵ, d, is $O(|X_k|\,|Y_k|\,|Z_k|/(\log n)^{100})$. In total, taking into account all $k \in [K]$, and possibly the at most L repetitions of Case 3.1, by Theorem 3.1 this becomes

$$\sum_{k=1}^{K} O\bigg(\frac{L\cdot|X_k|\,|Y_k|\,|Z_k|}{(\log n)^{100}}\bigg) = O\bigg(\frac{d^2L\cdot|X|\,|Y|\,|Z|}{(\log n)^{100}}\bigg) = O\bigg(\frac{|X|\,|Y|\,|Z|}{(\log n)^{99}}\bigg). \qquad \square$$

Claim 7.4 (Case 2)**.** *The total running time of Case 2 is*

$$O\bigg(Kn^{2.3} + |X|\,|Y|\,|Z|\cdot\frac{(\log\log n)^{14}}{(\log n)^6} + t_2\bigg),$$

where t_2 *is the number of triangles listed in Case 2.*

Proof. We can assume that $\mathbb{E}[A_k], \mathbb{E}[B_k] > 2^{-d}$ whenever we enter Case 2 (since Case 1 did not apply). Thus, the regularity decomposition (Theorem 3.1) guarantees that A_k and B_k^T are both $(\epsilon, 2, d)$-regular and ϵ-min-degree. As a corollary of Theorem 2.1 we obtain that

$$\mathbb{E}[A_k \circ B_k] \le (1+80\epsilon)\,\mathbb{E}[A]\,\mathbb{E}[B] + 2^{-\epsilon d/2} \le \tfrac{3}{2}\,\mathbb{E}[A]\,\mathbb{E}[B] + \mathbb{E}[A]\,\mathbb{E}[B] \le \tfrac{5}{2}\delta^2.$$

Write t_{G_k} to denote the number of triangles in G_k. By Lemma 7.1 and Theorem 3.1, it follows that Case 2 indeed takes total time

$$\begin{aligned}
&\sum_{\substack{k\in[K]\\ \text{case 2}}} O\bigg(n^{2.3} + |X_k|\,|Y_k|\,|Z_k|\cdot\bigg(\frac{1}{(\log n)^{100}} + \frac{\delta^2(\log\log n)^2}{(\log n)^2}\bigg) + t_{G_k}\bigg)\\
&\quad= \sum_{\substack{k\in[K]\\ \text{case 2}}} O\bigg(n^{2.3} + |X_k|\,|Y_k|\,|Z_k|\cdot\frac{(\log\log n)^{12}}{(\log n)^6} + t_{G_k}\bigg)\\
&\quad= O\bigg(Kn^{2.3} + |X|\,|Y|\,|Z|\cdot\frac{d^2(\log\log n)^{12}}{(\log n)^6} + t_2\bigg)\\
&\quad= O\bigg(Kn^{2.3} + |X|\,|Y|\,|Z|\cdot\frac{(\log\log n)^{14}}{(\log n)^6} + t_2\bigg). \qquad \square
\end{aligned}$$

Claim 7.5 (Case 3.3)**.** *The total running time of Case 3.3 is*

$$O\Big(KLn^{2.3} + \frac{|X|\,|Y|\,|Z|}{(\log n)^{99}} + t_{3.3}\Big),$$

where $t_{3.3}$ is the number of triangles listed in Case 3.3.

Proof. Focus on some pair (k, ℓ) that falls into Case 3.3, i.e., where $\mathbb{E}[B_k] \geq \delta$, $\mathbb{E}[C_{k,\ell}] > 2^{-L+1}$ and $|X_{k,\ell}| \geq \gamma |X_k|$. We must have $\ell < L$ (as otherwise $\mathbb{E}[C_{k,\ell}] \leq 2^{-L+1}$). The analysis of this case is inspired by the previous algorithm for purely random graphs. Our goal is to prove that both (i) the number of 2-paths in $G_{k,\ell}$ (via the edge parts A_k and $C_{k,\ell}$) and (ii) the number of triangles in $G_{k,\ell}$ behave as if $G_{k,\ell}$ was purely random.

We start with (i). By the construction of $C_{k,\ell}$, it is immediate that the density of $C_{k,\ell}$ is at least

$$\mathbb{E}[C_{k,\ell}] \geq \frac{|X_{k,\ell}|}{|X_k|} \cdot 2^{-\ell} \geq \gamma \cdot 2^{-\ell}.$$

Therefore, we can bound

$$\begin{aligned}
\mathbb{E}[A_k^T \circ C_{k,\ell}] &= \mathop{\mathbb{E}}_{x \in X_k} \deg_{A_k}(x) \cdot \deg_{C_{k,\ell}}(x) \\
&\leq \mathop{\mathbb{E}}_{x \in X_k} \deg_{A_k}(x) \cdot 2^{-\ell+1} \\
&\leq 2\gamma^{-1} \cdot \mathbb{E}[A_k] \cdot \mathbb{E}[C_{k,\ell}].
\end{aligned}$$

Next, we turn to (ii) and bound the number of triangles in $G_{k,\ell}$ from below. Using Theorem 2.1 we infer that at most a $2^{-\epsilon d/2} \leq 2^{-L}$-fraction of the entries in $\mathbb{E}[A_k \circ B_k]$ does not fall into the range $[(1-80\epsilon)\,\mathbb{E}[A_k]\,\mathbb{E}[B_k], (1+80\epsilon)\,\mathbb{E}[A_k]\,\mathbb{E}[B_k]] = [\frac{1}{2}\,\mathbb{E}[A_k]\,\mathbb{E}[B_k], \frac{3}{2}\,\mathbb{E}[A_k]\,\mathbb{E}[B_k]]$. Therefore, the number of triangles in $G_{k,\ell}$ is at least

$$\begin{aligned}
t_{G_{k,\ell}} &\geq |X_k|\,|Y_k|\,|Z_k| \cdot \tfrac{1}{2}\,\mathbb{E}[A_k] \cdot \mathbb{E}[B_k] \cdot (\mathbb{E}[C_{k,\ell}] - 2^{-L}) \\
&\geq |X_k|\,|Y_k|\,|Z_k| \cdot \tfrac{1}{4} \cdot \mathbb{E}[A_k] \cdot \mathbb{E}[B_k] \cdot \mathbb{E}[C_{k,\ell}] \\
&\geq |X_k|\,|Y_k|\,|Z_k| \cdot \tfrac{\gamma}{8} \cdot \mathbb{E}[B_k] \cdot \mathbb{E}[A_k^T \circ C_{k,\ell}] \\
&\geq |X_k|\,|Y_k|\,|Z_k| \cdot \tfrac{\gamma\delta}{8} \cdot \mathbb{E}[A_k^T \circ C_{k,\ell}].
\end{aligned}$$

By combining both statements, we can bound the running time of Lemma 7.1 as follows:

$$\begin{aligned}
&\sum_{\substack{k\in[K],\ell\in[L]\\ \text{case 3.3}}} O\Bigg(n^{2.3} + |X_k|\,|Y_k|\,|Z_k| \cdot \bigg(\frac{1}{(\log n)^{100}} + \frac{\mathbb{E}[A_k^T \circ C_{k,\ell}] \cdot (\log\log n)^2}{(\log n)^2}\bigg) + t_{G_{k,\ell}}\Bigg)\\
&\quad\le \sum_{\substack{k\in[K],\ell\in[L]\\ \text{case 3.3}}} O\Bigg(n^{2.3} + \frac{|X_k|\,|Y_k|\,|Z_k|}{(\log n)^{100}} + t_{G_{k,\ell}} \cdot \bigg(\frac{(\log\log n)^2}{(\gamma\delta/8)(\log n)^2} + 1\bigg)\Bigg)\\
&\quad\le \sum_{\substack{k\in[K],\ell\in[L]\\ \text{case 3.3}}} O\Bigg(n^{2.3} + \frac{|X_k|\,|Y_k|\,|Z_k|}{(\log n)^{100}} + t_{G_{k,\ell}}\Bigg)\\
&\quad= O\Bigg(KLn^{2.3} + \frac{d^2L \cdot |X|\,|Y|\,|Z|}{(\log n)^{100}} + t_{3.3}\Bigg)\\
&\quad= O\Bigg(KLn^{2.3} + \frac{|X|\,|Y|\,|Z|}{(\log n)^{99}} + t_{3.3}\Bigg).
\end{aligned}$$

□

It only remains to analyze Case 3.2, which involves recursive calls to our algorithm. Let us write $T(|X|, |Y|, |Z|, t, h)$ to express the total running time of our algorithm, given an input graph with vertex parts X, Y, Z and with at most t triangles, at recursion depth $0 \le h \le H$. Then finally:

Claim 7.6 (Total Running Time)**.** *The total running time is bounded by*

$$T(|X|, |Y|, |Z|, t, h) \le c \cdot \bigg((2KL)^{H-h} \cdot n^{2.3} + |X|\,|Y|\,|Z| \cdot \frac{(\log\log n)^{14} \cdot 2^{h+1}}{(\log n)^6} + t\bigg),$$

for some sufficiently large constant c.

Proof. In the base case we have $T(|X|, |Y|, |Z|, t, H) = O(|X|\,|Y|\,|Z|)$ which indeed satisfied the claim (as $2^H \ge (\log n)^6/(\log\log n)^{14}$). So focus on the case that $h < H$. Then, given the previous Claims 7.3 to 7.5, the running time of the Cases 1, 2, 3.1 and 3.3 is bounded by

$$c \cdot \bigg(KLn^{2.3} + |X|\,|Y|\,|Z| \cdot \frac{(\log\log n)^{14}}{(\log n)^6} + (t - t_{3.2})\bigg), \tag{4}$$

where c is some sufficiently large constant, and where $t_{3.2}$ is the number of triangles listed in Case 3.2 (such that $t - t_{3.2}$ is the number of triangles listed in all other cases). By induction we can bound the contribution of Case 3.2 as follows; recall that $|X_{k,\ell}| < \gamma|X_k|$ for any pair (k, ℓ) falling into Case 3.2:

$$\begin{aligned}
&\sum_{\substack{k\in[K],\ell\in[L]\\ \text{case 3.2}}} T(|X_{k,\ell}|, |Y_k|, |Z_k|, t_{G_{k,\ell}}, h+1)\\
&\quad\le \sum_{\substack{k\in[K],\ell\in[L]\\ \text{case 3.2}}} c \cdot \bigg((2KL)^{H-h-1} \cdot n^{2.3} + |X_{k,\ell}|\,|Y_k|\,|Z_k| \cdot \frac{(\log\log n)^{14} \cdot 2^{h+2}}{(\log n)^6} + t_{G_{k,\ell}}\bigg)\\
&\quad\le \sum_{\substack{k\in[K],\ell\in[L]\\ \text{case 3.2}}} c \cdot \bigg((2KL)^{H-h-1} \cdot n^{2.3} + \gamma \cdot |X_k|\,|Y_k|\,|Z_k| \cdot \frac{(\log\log n)^{14} \cdot 2^{h+2}}{(\log n)^6} + t_{G_{k,\ell}}\bigg)\\
&\quad c \cdot \bigg(KL(2KL)^{H-h-1} \cdot n^{2.3} + 2\gamma L(d+2)^2 \cdot |X|\,|Y|\,|Z| \cdot \frac{(\log\log n)^{14} \cdot 2^{h+2}}{(\log n)^6} + t_{3.2}\bigg).
\end{aligned} \tag{5}$$

By our choice of parameters, $2\gamma L(d+2)^2 = \frac{1}{4}$. Therefore, the total running time (which is obtained as the sum of Equations (4) and (5)) is at most

$$\begin{aligned} &T(|X|, |Y|, |Z|, t, h) \\ &\quad \le c \cdot \left(2KL(2KL)^{H-h-1} \cdot n^{2.3} + |X|\,|Y|\,|Z| \cdot \frac{(\log\log n)^{14} \cdot 2^{h+1}}{(\log n)^6} + (t - t_{3.2} + t_{3.2}) \right) \\ &\quad = c \cdot \left((2KL)^{H-h} \cdot n^{2.3} + |X|\,|Y|\,|Z| \cdot \frac{(\log\log n)^{14} \cdot 2^{h+1}}{(\log n)^6} + t \right), \end{aligned}$$

as stated. □

To obtain the time bound claimed in the theorem statement, recall that we initially call the algorithm at recursion depth $h = 0$, that $K \le \exp(\text{poly}(d, \epsilon^{-1})) = \exp(\text{poly}(\log\log n))$, and that $H = O(\log\log n)$. Hence, the term $(2KL)^H \cdot n^{2.3} \le n^{2.3+o(1)}$ is negligible in the total running time. □

7.3 Triangle Enumeration versus Triangle Listing

We finally turn our triangle listing algorithm into an enumeration algorithm. In fact, we prove the following equivalence:[14]

Lemma 7.7 (Equivalence of Triangle Enumeration and Listing)**.** *The following equivalences hold (in terms of deterministic algorithms):*

1. *If constant-delay triangle enumeration is possible with preprocessing time $O(n^3/f(n))$ (for some function $f(n)$), then triangle listing is in time $O(n^3/f(n) + t)$.*
2. *If triangle listing is in time $O(n^3/f(n)+t)$ (for some computable function $f(n) = n^{o(1)}$), then constant-delay triangle enumeration is possible with preprocessing time $O(n^3/f(n^{1/2}))$.*

To obtain the equivalence in terms of *deterministic* algorithms we rely on the following combinatorial algorithm by Fox, Lovász and Zhao to approximately count triangles (or in fact, arbitrary subgraphs):

Lemma 7.8 (Approximate Triangle Counting [37])**.** *Let $\epsilon > 0$. There is a deterministic algorithm approximating the number of triangles in a graph up to an additive error of ϵn^3 in time $n^2 \cdot \text{poly}(\epsilon^{-1})$.*

Proof of Lemma 7.7. The first item is trivial: Simply preprocess the graph in time $O(n^3/f(n))$ and then enumerate all triangles in time $O(t)$. We thus focus on the second item, and design a triangle enumeration algorithm with constant delay.

[14]This equivalence can easily be extended to an equivalence for arbitrary subgraphs H. Specifically, if H has k vertices, then listing H-subgraphs in time $O(n^k/f(n) + t)$ is equivalent to enumerating H-subgraphs with constant delay in preprocessing time $O(n^k/f(n))$. However, this equivalence has only limited use, as for many graphs H much faster listing and enumeration algorithms are known. For instance, 4-cycles can be listed in time $O(n^2 + t)$ by a simple folklore algorithm.

Preprocessing. Assume without loss of generality that G is tripartite with vertex sets X, Y, Z of size n each. We partition each vertex part into $g = \lceil n^{1/2} \rceil$ groups $X = X_1 \uplus \cdots \uplus X_g$, $Y = Y_1 \uplus \cdots \uplus Y_g$ and $Z = Z_1 \uplus \cdots \uplus Z_g$ of size at most $\lceil n^{1/2} \rceil$. For each triple $i, j, k \in [g]$, let $G_{i,j,k}$ denote the subgraph induced by $X_i \cup Y_j \cup Z_k$. Clearly the triangles in G are perfectly partitioned into the triangles in $(G_{i,j,k})_{i,j,k}$. Let $t_{i,j,k}$ denote the number of triangles in $G_{i,j,k}$.

Let $\epsilon = \frac{1}{4}/f(n^{1/2})$. For each triple $i, j, k \in [g]$, we compute an approximation $\widetilde{t}_{i,j,k}$ of the number of triangles in $G_{i,j,k}$ with absolute error $\epsilon n^{3/2} = \frac{1}{4} n^{3/2}/f(n^{1/2})$. We call a triple i, j, k with $\widetilde{t}_{i,j,k} \leq n^{3/2}/f(n^{1/2})$ *light* and *heavy* otherwise. For each light triple i, j, k we list all triangles in $G_{i,j,k}$ using the efficient listing algorithm and discard $G_{i,j,k}$ afterwards. We store all triangles discovered in this step in one global list to be enumerated later. We sort the remaining heavy triples in descending order according to their approximate triangle counts $\widetilde{t}_{i,j,k}$; let $G_1, \ldots, G_r$ denote the heavy graphs in the resulting order. As the final step of the preprocessing phase, we list all triangles in G_1 in brute-force time $O(n^{3/2})$.

Preprocessing Time. Before proceeding to the description of the enumeration phase, we quickly analyze the running time of the preprocessing phase. Computing the approximate triangle counts for all subgraphs takes time $O(n^{3/2} \cdot n \cdot \mathrm{poly}(f(n^{1/2}))) = n^{5/2+o(1)}$. Observe that the number of triangles in each light subgraph is at most $n^{3/2}/f(n^{1/2}) + \epsilon n^{3/2} = O(n^{3/2}/f(n^{1/2}))$. Therefore, applying the listing algorithm to all light subgraphs takes time $O(n^{3/2} \cdot n^{3/2}/f(n^{1/2})) = O(n^3/f(n^{1/2}))$. Finally, listing all triangles in G_1 takes negligible time.

Enumeration. In the enumeration phase we first enumerate all triangles from the light subgraphs. The more interesting part is to enumerate the triangles involving the heavy subgraphs. Here we proceed as follows: We maintain an *active* heavy subgraph $i \in [r]$ for which we have prepared a list of all of its triangles. Initially we set $i = 1$—and indeed, we have prepared a list of all triangles in G_1. The idea is that while we enumerate triangles from the active graph G_i, we simultaneously prepare a list of all triangle in the next graph G_{i+1}. To this end, we run the efficient listing algorithm on G_{i+1}, and with each triangle from G_i that we list we advance the algorithm by $O(1)$ computation steps. To prove that this approach succeeds (i.e., that we have completed the execution of the listing algorithm on G_{i+1} when all triangles from G_i have been listed), we show that $t_i \geq \Omega(n^{3/2}/f(n^{1/2}) + t_{i+1})$, where we write t_i to denote the number of triangles in G_i. Indeed, we have that $t_i \geq t_{i+1} - 2\epsilon n^{3/2}$ and $t_i \geq n^{3/2}/f(n^{1/2}) - \epsilon n^{3/2}$ (as G_i is heavy), and thus $t_i \geq \Omega(n^{3/2}/f(n^{1/2}) + t_{i+1})$. After we have enumerated all triangles from G_i, we discard G_i and consider G_{i+1} the next active subgraph. Note that this algorithm indeed lists all triangles with constant delay. $\square$

The proof of Theorem 1.3 is complete by combining Theorem 7.2 and Lemma 7.7.

7.4 Conditional Optimality

The famous 3-SUM problem is to test whether in a given set S of n integers there is a solution to the equation $a + b + c = 0$. This problem can be solved naively in time $O(n^2)$ and the fastest known algorithm, due to Baran, Demaine and Pătraşcu, runs in expected time $O(n^2/(\log n)^2 \cdot (\log \log n)^2)$.

From previous work on the 3-SUM-hardness of triangle listing/enumeration by Pătraşcu [62] and by Kopelowitz, Pettie and Porat [54], it follows that our algorithm is *conditionally optimal* in the following (weak) sense: An algorithm that enumerates all triangles in a graph with preprocessing time $O(n^3/(\log n)^{6+\epsilon})$ and constant delay entails an algorithm for the 3-SUM problem in expected time $O(n^2/(\log n)^{2+\epsilon'})$ (i.e., a $(\log n)^{\epsilon'}$ improvement over the Baran-Demaine-Pătraşcu algorithm). Since this statement is not explicit in [54], we devote this section to an almost-self-contained proof. Specifically, we prove the following statement:

Lemma 7.9 (Reducing 3-SUM to Triangle Listing)**.** *Let $\alpha \geq 0$. If there is a randomized algorithm listing all t triangles in a graph in expected time $O(n^3/f(n)+t)$ (for some computable nondecreasing function $f(n)$), then there is a randomized 3-SUM algorithm in expected time $O(n^2/f(n^{2/3})^{1/3})$.*[15]

The proof relies on the following standard lemma on linear hashing:

Lemma 7.10 (Linear Hashing [28])**.** *Let $n \geq m \geq 1$. There is a family $\mathcal{H} = \{h : [-n \mathinner{..} n] \to [m]\}$ of hash functions that can be sampled in expected time* $\mathrm{poly}(\log n)$ *and evaluated in constant time, and that satisfies the following properties:*

- *(Almost-Linearity) There exists some constant-size set Φ such that for all $h \in \mathcal{H}$ and all keys $a, b \in [-n \mathinner{..} n]$, we have $h(a+b) - h(a) - h(b) + h(0) \in \Phi$.*
- *(Pairwise Independence) For all distinct keys $a, b \in [-n \mathinner{..} n]$ and buckets $x, y \in [m]$: $\mathbb{P}_{h \in \mathcal{H}}[h(a) = x \text{ and } h(b) = y] \leq O(\frac{1}{m^2})$.*

Proof of Lemma 7.9. Let S denote the given 3-SUM-instance, and assume that $S \subseteq [-n^c \mathinner{..} n^c]$ for some constant c. As a first step, we randomly sample $O(n \log n)$ pairs $a, b \in S$ and test whether $-a-b \in S$. If we find a solution in this step, we stop and return "yes".

Otherwise, sample three linear hash functions $h_1, h_2, h_3 : [-n^c \mathinner{..} n^c] \to [m]$ as in the previous lemma, and construct the following tripartite graph $G = (X, Y, Z, A, B, C)$. As vertex parts, we take

$$\begin{aligned} X &= [m] \times [m] \times \{h_3(0)\}, \\ Y &= [m] \times \{h_2(0)\} \times [m], \\ Z &= \{h_1(0)\} \times [m] \times [m]. \end{aligned}$$

We add an edge $(x, y) \in X \times Y$ to A if and only if there is some $a \in S$ such that

$$\begin{aligned} &y_1 - x_1 - h_1(a) + h_1(0) \in \Phi, \\ &y_2 - x_2 - h_2(a) + h_2(0) \in \Phi, \\ &y_3 - x_3 - h_3(a) + h_3(0) \in \Phi. \end{aligned}$$

In this case, we say that the edge (x, y) is *labeled* with a. We add edges to B and C in the analogous way. We then use the efficient triangle listing algorithm to list all triangles in G (viewing G as an unlabeled graph). For each triangle (x, y, z) that is returned, test whether it has three edge labels (a, b, c) that satisfy $a + b + c = 0$. In this case we report "yes", and if no such triangle is found, return "no".

[15] We remark that while this reduction is in terms of *expected* running times, the statement can easily be strengthened such that the running time of the constructed 3-SUM algorithm holds with high probability (e.g., by first applying the known self-reduction for 3-SUM [45, 57] and splitting into polynomially many subinstances).

Correctness. For the correctness it suffices to show that, if there exist $a, b, c \in S$ with $a+b+c = 0$, then there is a triangle in G with edge labels (a, b, c). To see this, consider the triple (x, y, z) where

$$x = (h_1(c), h_2(-a), h_3(0)),$$
$$y = (h_1(-b), h_2(0), h_3(a)),$$
$$z = (h_1(0), h_2(b), h_3(-c)).$$

We claim that (x, y, z) forms a triangle. This involves testing nine constraints. For the sake of brevity, we only prove the first such constraint, namely that $y_1 - x_1 - h_1(a) + h_1(0) \in \Phi$. And indeed, given that $y_1 - x_1 - h_1(a) = h_1(-b) - h_1(c) - h_1(a) = h_1(c+a) - h_1(c) - h_1(a)$, the constraint follows by the almost-linearity property of Lemma 7.10.

Running Time. Sampling solutions and constructing the graph G takes negligible time, so the critical contribution is the running time of the listing algorithm. To this end, we first bound the expected number of triangles in the graph.

Claim. *The expected number of triangles in G is $O(n + \frac{n^3}{m^3})$.*

Proof. Fix a triple $a, b, c \in S$. We analyze how many triangles (x, y, z) are labeled with (a, b, c). Recall that for (x, y, z) to be a triangle, it satisfies nine constraints—three for each edge. Focus on the constraints involving h_1:

$$y_1 - x_1 - h_1(a) + h_1(0) \in \Phi,$$
$$z_1 - y_1 - h_1(b) + h_1(0) \in \Phi,$$
$$x_1 - z_1 - h_1(c) + h_1(0) \in \Phi.$$

Recall that necessarily $z_1 = h_1(0)$. Thus, the second and third constraints imply that there are only $|\Phi| = O(1)$ feasible choices for x_1 and y_1. In particular, it follows that there are at most $O(1)$ triangles labeled with (a, b, c). Next, let us write $k\Phi = \{\phi_1 + \cdots + \phi_k : \phi_1, \ldots, \phi_k \in \Phi\}$. Summing all three constraints, we obtain that $3h_1(0) - h_1(a) - h_1(b) - h_3(c) \in 3\Phi$, and by applying the almost-linearity twice, it follows that $h_1(0) - h_1(a+b+c) \in 5\Phi$. By the analogous argument for h_2 and h_3, we conclude that there is a triangle labeled with (a, b, c) only if

$$h_1(0) - h_1(a + b + c) \in 5\Phi,$$
$$h_2(0) - h_2(a + b + c) \in 5\Phi,$$
$$h_3(0) - h_3(a + b + c) \in 5\Phi.$$

Finally, to bound the expected number of triangles, we distinguish two cases. On the one hand, the number of triples with $a+b+c = 0$ is at most $O(n)$ as otherwise, with high probability, we would have detected a 3-SUM solution in the first step of the algorithm. Each such triple contributes at most $O(1)$ triangles. On the other hand, there are up to n^3 triples with $a + b + c \neq 0$. Each such triple contributes at most $O(1)$ triangles, and only if the final three constraints are satisfied. By pairwise independence, each constraint holds with probability at most $O(\frac{|\Phi|}{m}) = O(\frac{1}{m})$, and the three constraints are independent. It follows that these triples contribute $O(\frac{n^3}{m^3})$ triangles in expectation. □

Observe that the graph G has $O(m^2)$ vertices. Therefore, given the previous claim, the total expected running time is bounded by $O(m^6/f(m^2) + n^3/m^3)$. By choosing $m = \lceil n^{1/3} \cdot f(n^{2/3})^{1/9} \rceil$, this becomes $O(n^2/f(n^{2/3})^{1/3})$ (using that f is nondecreasing). □

References

[1] Amir Abboud, Arturs Backurs, Karl Bringmann, and Marvin Künnemann. Fine-grained complexity of analyzing compressed data: Quantifying improvements over decompress-and-solve. In Chris Umans, editor, *58th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2017)*, pages 192–203. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.26.

[2] Amir Abboud, Karl Bringmann, and Nick Fischer. Stronger 3-sum lower bounds for approximate distance oracles via additive combinatorics. In Barna Saha and Rocco A. Servedio, editors, *55th Annual ACM Symposium on Theory of Computing (STOC 2023)*, pages 391–404. ACM, 2023. doi:10.1145/3564246.3585240.

[3] Amir Abboud, Karl Bringmann, Seri Khoury, and Or Zamir. Hardness of approximation in P via short cycle removal: Cycle detection, distance oracles, and beyond. In Stefano Leonardi and Anupam Gupta, editors, *54th Annual ACM Symposium on Theory of Computing (STOC 2022)*, pages 1487–1500. ACM, 2022. doi:10.1145/3519935.3520066.

[4] Amir Abboud, Nick Fischer, and Yarin Shechter. Faster combinatorial k-clique algorithms. *Personal Communication*, 2023.

[5] Amir Abboud, Loukas Georgiadis, Giuseppe F. Italiano, Robert Krauthgamer, Nikos Parotsidis, Ohad Trabelsi, Przemyslaw Uznanski, and Daniel Wolleb-Graf. Faster algorithms for all-pairs bounded min-cuts. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *LIPIcs*, pages 7:1–7:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.7.

[6] Amir Abboud and Nathan Wallheimer. Worst-case to expander-case reductions. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*, volume 251 of *LIPIcs*, pages 1:1–1:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. URL: https://doi.org/10.4230/LIPIcs.ITCS.2023.1, doi:10.4230/LIPICS.ITCS.2023.1.

[7] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2014)*, pages 434–443. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.53.

[8] Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. *SIAM J. Comput.*, 47(3):1098–1122, 2018. doi:10.1137/15M1050987.

[9] Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM J. Comput.*, 28(4):1167–1181, 1999. doi:10.1137/S0097539796303421.

[10] Rasmus Resen Amossen and Rasmus Pagh. Faster join-projects and sparse matrix multiplications. In Ronald Fagin, editor, *12th International Conference on Database Theory (ICDT 2009)*, volume 361 of *ACM International Conference Proceeding Series*, pages 121–126. ACM, 2009. doi:10.1145/1514894.1514909.

[11] Dana Angluin. The four Russians' algorithm for Boolean matrix multiplication is optimal in its class. *SIGACT News*, 8(1):29–33, 1976. doi:10.1145/1008591.1008593.

[12] Vladimir L. Arlazarov, Yefim A. Dinic, Aleksandr Kronrod, and IgorAleksandrovich Faradžev. On economical construction of the transitive closure of an oriented graph. In *Doklady Akademii Nauk*, volume 194, pages 487–488. Russian Academy of Sciences, 1970.

[13] Nikhil Bansal and Ryan Williams. Regularity lemmas and combinatorial algorithms. *Theory Comput.*, 8(1):69–94, 2012. URL: https://doi.org/10.4086/toc.2012.v008a004, doi:10.4086/TOC.2012.V008A004.

[14] Ilya Baran, Erik D. Demaine, and Mihai Pătraşcu. Subquadratic algorithms for 3SUM. *Algorithmica*, 50(4):584–596, 2008. doi:10.1007/s00453-007-9036-3.

[15] Felix A. Behrend. On sets of integers which contain no three terms in arithmetical progression. *Proc. Natl. Acad. Sci.*, 32(12):331–332, 1946.

[16] Thiago Bergamaschi, Monika Henzinger, Maximilian Probst Gutenberg, Virginia Vassilevska Williams, and Nicole Wein. New techniques and fine-grained hardness for dynamic near-additive spanners. In Dániel Marx, editor, *32nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)*, pages 1836–1855. SIAM, 2021. doi:10.1137/1.9781611976465.110.

[17] Andreas Björklund, Rasmus Pagh, Virginia Vassilevska Williams, and Uri Zwick. Listing triangles. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *41st International Colloquium on Automata, Languages, and Programming (ICALP 2014)*, volume 8572 of *Lecture Notes in Computer Science*, pages 223–234. Springer, 2014. doi:10.1007/978-3-662-43948-7_19.

[18] Karl Bringmann, Nick Fischer, and Marvin Künnemann. A fine-grained analogue of schaefer's theorem in P: Dichotomy of $\exists^k\forall$-quantified first-order graph properties. In Amir Shpilka, editor, *34th Computational Complexity Conference (CCC 2019)*, volume 137 of *LIPIcs*, pages 31:1–31:27. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.CCC.2019.31.

[19] Karl Bringmann, Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Tree edit distance cannot be computed in strongly subcubic time (unless APSP can). *ACM Trans. Algorithms*, 16(4):48:1–48:22, 2020. doi:10.1145/3381878.

[20] Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. A dichotomy for regular expression membership testing. In Chris Umans, editor, *58th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2017)*, pages 307–318. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.36.

[21] Karl Bringmann and Philip Wellnitz. Clique-based lower bounds for parsing tree-adjoining grammars. In Juha Kärkkäinen, Jakub Radoszewski, and Wojciech Rytter, editors, *28th Annual Symposium on Combinatorial Pattern Matching (CPM 2017)*, volume 78 of *LIPIcs*, pages 12:1–12:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. URL: https://doi.org/10.4230/LIPIcs.CPM.2017.12, doi:10.4230/LIPICS.CPM.2017.12.

[22] Katrin Casel and Markus L. Schmid. Fine-grained complexity of regular path queries. In Ke Yi and Zhewei Wei, editors, *24th International Conference on Database Theory (ICDT 2021)*, volume 186 of *LIPIcs*, pages 19:1–19:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. URL: https://doi.org/10.4230/LIPIcs.ICDT.2021.19, doi:10.4230/LIPICS.ICDT.2021.19.

[23] Timothy M. Chan. Speeding up the four russians algorithm by about one more logarithmic factor. In Piotr Indyk, editor, *26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015)*, pages 212–217. SIAM, 2015. doi:10.1137/1.9781611973730.16.

[24] Timothy M. Chan, Saladi Rahul, and Jie Xue. Range closest-pair search in higher dimensions. *Comput. Geom.*, 91:101669, 2020. URL: https://doi.org/10.1016/j.comgeo.2020.101669, doi:10.1016/J.COMGEO.2020.101669.

[25] Yi-Jun Chang. Hardness of RNA folding problem with four symbols. *Theor. Comput. Sci.*, 757:11–26, 2019. URL: https://doi.org/10.1016/j.tcs.2018.07.010, doi:10.1016/J.TCS.2018.07.010.

[26] Benny Chor and Oded Goldreich. On the power of two-point based sampling. *J. Complex.*, 5(1):96–106, 1989. doi:10.1016/0885-064X(89)90015-0.

[27] Raphaël Clifford, Allan Grønlund, Kasper Green Larsen, and Tatiana Starikovskaya. Upper and lower bounds for dynamic data structures on strings. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2018)*, volume 96 of *LIPIcs*, pages 22:1–22:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. URL: https://doi.org/10.4230/LIPIcs.STACS.2018.22, doi:10.4230/LIPICS.STACS.2018.22.

[28] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition.* MIT Press, 2009. URL: http://mitpress.mit.edu/books/introduction-algorithms.

[29] Artur Czumaj, Miroslaw Kowaluk, and Andrzej Lingas. Faster algorithms for finding lowest common ancestors in directed acyclic graphs. *Theor. Comput. Sci.*, 380(1-2):37–46, 2007. URL: https://doi.org/10.1016/j.tcs.2007.02.053, doi:10.1016/J.TCS.2007.02.053.

[30] Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Morten Stöckel. Finding even cycles faster via capped k-walks. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th Annual ACM Symposium on Theory of Computing (STOC 2017)*, pages 112–120. ACM, 2017. doi:10.1145/3055399.3055459.

[31] Mina Dalirrooyfard, Thuy-Duong Vuong, and Virginia Vassilevska Williams. Graph pattern detection: Hardness for all induced patterns and faster noninduced cycles. *SIAM J. Comput.*, 50(5):1627–1662, 2021. doi:10.1137/20M1335054.

[32] Debarati Das, Michal Koucký, and Michael E. Saks. Lower bounds for combinatorial algorithms for boolean matrix multiplication. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2018)*, volume 96 of *LIPIcs*, pages 23:1–23:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. URL: https://doi.org/10.4230/LIPIcs.STACS.2018.23, doi:10.4230/LIPICS.STACS.2018.23.

[33] Ran Duan, Hongxun Wu, and Renfei Zhou. Faster matrix multiplication via asymmetric hashing. In *64th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2023)*. IEEE Computer Society, 2023. To appear. URL: https://doi.org/10.48550/arXiv.2210.10173.

[34] Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theor. Comput. Sci.*, 326(1-3):57–67, 2004. URL: https://doi.org/10.1016/j.tcs.2004.05.009, doi:10.1016/J.TCS.2004.05.009.

[35] Michael J. Fischer and Albert R. Meyer. Boolean matrix multiplication and transitive closure. In *12th Annual Symposium on Switching and Automata Theory (SWAT 1971)*, pages 129–131. IEEE Computer Society, 1971. doi:10.1109/SWAT.1971.4.

[36] Jacob Fox. A new proof of the graph removal lemma. *Annals of Mathematics*, 174(1):561–579, 2011. URL: http://www.jstor.org/stable/23030574.

[37] Jacob Fox, László Miklós Lovász, and Yufei Zhao. A fast new algorithm for weak graph regularity. *Comb. Probab. Comput.*, 28(5):777–790, 2019. doi:10.1017/S0963548319000075.

[38] Alan M. Frieze and Ravi Kannan. Quick approximation to matrices and applications. *Comb.*, 19(2):175–220, 1999. doi:10.1007/s004930050052.

[39] David Gillman. A Chernoff bound for random walks on expander graphs. *SIAM J. Comput.*, 27(4):1203–1220, 1998. doi:10.1137/S0097539794268765.

[40] Oded Goldreich. A sample of samplers: A computational perspective on sampling. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, volume 6650 of *Lecture Notes in Computer Science*, pages 302–332. Springer, 2011. doi:10.1007/978-3-642-22670-0_24.

[41] Oded Goldreich and Avi Wigderson. Tiny families of functions with random properties: A quality-size trade-off for hashing. *Random Struct. Algorithms*, 11(4):315–343, 1997. doi:10.1002/(SICI)1098-2418(199712)11:4\\<315::AID-RSA3\\>3.0.CO;2-1.

[42] Timothy W. Gowers. Lower bounds of tower type for Szemerédi's uniformity lemma. *Geometric & Functional Analysis (GAFA)*, 7(2):322–337, 1997. doi:10.1007/PL00001621.

[43] Timothy W. Gowers. A new proof of Szemerédi's theorem. *Geometric & Functional Analysis (GAFA)*, 11:465–588, 08 2001. doi:10.1007/s00039-001-0332-9.

[44] Timothy W. Gowers. Quasirandomness, counting and regularity for 3-uniform hypergraphs. *Comb. Probab. Comput.*, 15(1-2):143–184, 2006. doi:10.1017/S0963548305007236.

[45] Allan Grønlund and Seth Pettie. Threesomes, degenerates, and love triangles. *J. ACM*, 65(4):22:1–22:25, 2018. doi:10.1145/3185378.

[46] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from parvaresh-vardy codes. *J. ACM*, 56(4):20:1–20:34, 2009. doi:10.1145/1538902.1538904.

[47] Hamed Hatami. Graph norms and Sidorenko's conjecture. *Israel Journal of Mathematics*, 175(1):125–150, 2010. doi:10.1007/s11856-010-0005-1.

[48] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing (STOC 2015)*, pages 21–30. ACM, 2015. doi:10.1145/2746539.2746609.

[49] Zhiyi Huang, Yaowei Long, Thatchaphol Saranurak, and Benyu Wang. Tight conditional lower bounds for vertex connectivity problems. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 1384–1395. ACM, 2023. doi:10.1145/3564246.3585223.

[50] Ce Jin and Yinzhan Xu. Tight dynamic problem lower bounds from generalized BMM and omv. In Stefano Leonardi and Anupam Gupta, editors, *63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2022)*, pages 1515–1528. ACM, 2022. doi:10.1145/3519935.3520036.

[51] Ce Jin and Yinzhan Xu. Removing additive structure in 3sum-based reductions. In Barna Saha and Rocco A. Servedio, editors, *55th Annual ACM Symposium on Theory of Computing (STOC 2023)*, pages 405–418. ACM, 2023. doi:10.1145/3564246.3585157.

[52] Zander Kelley, Shachar Lovett, and Raghu Meka. Explicit separations between randomized and deterministic number-on-forehead communication. *arXiv*, 2023. doi:10.48550/arXiv.2308.12451.

[53] Zander Kelley and Raghu Meka. Strong bounds for 3-progressions. In *64th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2022)*. IEEE, 2023. To appear.

[54] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In Robert Krauthgamer, editor, *27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016)*, pages 1272–1287. SIAM, 2016. doi:10.1137/1.9781611974331.ch89.

[55] Kasper Green Larsen and R. Ryan Williams. Faster online matrix-vector multiplication. In Philip N. Klein, editor, *28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 2182–2189. SIAM, 2017. doi:10.1137/1.9781611974782.142.

[56] Lillian Lee. Fast context-free grammar parsing requires fast boolean matrix multiplication. *J. ACM*, 49(1):1–15, 2002. doi:10.1145/505241.505242.

[57] Andrea Lincoln, Virginia Vassilevska Williams, Joshua R. Wang, and R. Ryan Williams. Deterministic time-space trade-offs for k-sum. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *LIPIcs*, pages 58:1–58:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.58.

[58] Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In Artur Czumaj, editor, *29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*, pages 1236–1252. SIAM, 2018. doi:10.1137/1.9781611975031.80.

[59] Jirí Matousek. Computing dominances in e^n. *Inf. Process. Lett.*, 38(5):277–278, 1991. doi:10.1016/0020-0190(91)90071-O.

[60] J. Ian Munro. Efficient determination of the transitive closure of a directed graph. *Inf. Process. Lett.*, 1(2):56–58, 1971. doi:10.1016/0020-0190(71)90006-8.

[61] Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

[62] Mihai Pǎtraşcu. Towards polynomial lower bounds for dynamic problems. In Leonard J. Schulman, editor, *42nd Annual ACM Symposium on Theory of Computing (STOC 2010)*, pages 603–610. ACM, 2010. doi:10.1145/1806689.1806772.

[63] Omer Reingold, Salil P. Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 3–13. IEEE Computer Society, 2000. doi:10.1109/SFCS.2000.892006.

[64] Liam Roditty and Uri Zwick. On dynamic shortest paths problems. *Algorithmica*, 61(2):389–401, 2011. URL: https://doi.org/10.1007/s00453-010-9401-5, doi:10.1007/S00453-010-9401-5.

[65] Giorgio Satta. Tree-adjoining grammar parsing and boolean matrix multiplication. *Comput. Linguistics*, 20(2):173–191, 1994.

[66] Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969. doi:10.1007/BF02165411.

[67] Endre Szemerédi. On sets of integers containing no k elements in arithmetic progression. *Acta Arith*, 27(199–245), 1975.

[68] Leslie G. Valiant. General context-free recognition in less than cubic time. *J. Comput. Syst. Sci.*, 10(2):308–315, 1975. doi:10.1016/S0022-0000(75)80046-8.

[69] Dirk Van Gucht, Ryan Williams, David P. Woodruff, and Qin Zhang. The communication complexity of distributed set-joins with applications to matrix multiplication. In Tova Milo and Diego Calvanese, editors, *34th ACM Symposium on Principles of Database Systems (PODS 2015)*, pages 199–212. ACM, 2015. doi:10.1145/2745754.2745779.

[70] Virginia Vassilevska, Ryan Williams, and Raphael Yuster. All-pairs bottleneck paths for general graphs in truly sub-cubic time. In David S. Johnson and Uriel Feige, editors, *39th Annual ACM Symposium on Theory of Computing (STOC 2007)*, pages 585–589. ACM, 2007. doi:10.1145/1250790.1250876.

[71] R. Ryan Williams. Faster all-pairs shortest paths via circuit complexity. *SIAM J. Comput.*, 47(5):1965–1985, 2018. doi:10.1137/15M1024524.

[72] Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5):27:1–27:38, 2018. doi:10.1145/3186893.

[73] Virginia Vassilevska Williams and Yinzhan Xu. Monochromatic triangles, triangle listing and APSP. In Sandy Irani, editor, *61st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2020)*, pages 786–797. IEEE, 2020. doi:10.1109/FOCS46700.2020.00078.

[74] Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In *35th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2024)*. SIAM, 2024. To appear. doi:10.48550/ARXIV.2307.07970.

[75] Huacheng Yu. An improved combinatorial algorithm for boolean matrix multiplication. *Inf. Comput.*, 261:240–247, 2018. doi:10.1016/j.ic.2018.02.006.

[76] Yufei Zhao. *Graph Theory and Additive Combinatorics: Exploring Structure and Randomness.* Cambridge University Press, 2023.

[77] David Zuckerman. Randomness-optimal oblivious sampling. *Random Struct. Algorithms*, 11(4):345–367, 1997. doi:10.1002/(SICI)1098-2418(199712)11:4\%3C345::AID-RSA4\%3E3.0.CO;2-Z.