On Nondeterministic Derandomization of Freivalds' Algorithm: Consequences, Avenues and Algorithmic Progress

Marvin Künnemann
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
marvin@mpi-inf.mpg.de

Abstract

Motivated by studying the power of randomness, certifying algorithms and barriers for fine-grained reductions, we investigate the question whether the multiplication of two $n \times n$ matrices can be performed in near-optimal *nondeterministic* time $\tilde{\mathcal{O}}(n^2)$. Since a classic algorithm due to Freivalds verifies correctness of matrix products probabilistically in time $\mathcal{O}(n^2)$, our question is a relaxation of the open problem of derandomizing Freivalds' algorithm.

We discuss consequences of a positive or negative resolution of this problem and provide potential avenues towards resolving it. Particularly, we show that sufficiently fast deterministic verifiers for 3SUM or univariate polynomial identity testing yield faster deterministic verifiers for matrix multiplication. Furthermore, we present the partial algorithmic progress that distinguishing whether an integer matrix product is correct or contains between 1 and n erroneous entries can be performed in time $\tilde{\mathcal{O}}(n^2)$ – interestingly, the difficult case of deterministic matrix product verification is not a problem of "finding a needle in the haystack", but rather cancellation effects in the presence of many errors.

Our main technical contribution is a deterministic algorithm that corrects an integer matrix product containing at most t errors in time $\tilde{\mathcal{O}}(\sqrt{t}n^2 + t^2)$. To obtain this result, we show how to compute an integer matrix product with at most t nonzeroes in the same running time. This improves upon known deterministic output-sensitive integer matrix multiplication algorithms for $t = \Omega(n^{2/3})$ nonzeroes, which is of independent interest.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases matrix product verification, certifying computation, fine-grained complexity and algorithms

Digital Object Identifier 10.4230/LIPIcs.ESA.2018.56

Related Version A full version of the paper is available at [27], `https://arxiv.org/abs/1806.09189`.

Acknowledgements The author wishes to thank Markus Bläser, Russell Impagliazzo, Kurt Mehlhorn, Ramamohan Paturi, and Michael Sagraloff for early discussions on this work and Karl Bringmann for comments on a draft of this paper.

1 Introduction

Fast matrix multiplication algorithms belong to the most exciting algorithmic developments in the realm of low-degree polynomial-time problems. Starting with Strassen's polynomial speedup [38] over the naive $\mathcal{O}(n^3)$-time algorithm, extensive work (see, e.g., [13, 41, 29]) has brought down the running time to $\mathcal{O}(n^{2.373})$ (we refer to [8] for a survey). This leads to substantial improvements over naive solutions for a wide range of applications; for many

26th Annual European Symposium on Algorithms (ESA 2018).
Editors: Yossi Azar, Hannah Bast, and Grzegorz Herman; Article No. 56; pp. 56:1–56:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

problems, the best known algorithms make crucial use of fast multiplication of square or rectangular matrices. To name just a few examples, we do not only obtain polynomial improvements for numerous tasks in linear algebra (computing matrix inverses, determinants, etc.), graph theory (finding large cliques in graphs [33], All-Pairs Shortest Path for bounded edge-weights [4]), stringology (context free grammar parsing [40], RNA folding and language edit distance [9]) and many more, but also strong subpolynomial improvements such as a $2^{\Omega(\sqrt{\log n})}$-factor speed-up for the All-Pairs Shortest Path problem (APSP) [46] or similar improvements for the orthogonal vectors problem (OV) [3]. It is a famous open question whether the matrix multiplication exponent ω is equal to 2.

Matrix multiplication is the search version of the MM-Verification problem: given $n \times n$ matrices A, B and a candidate C for the product matrix, verify whether $AB = C$. There is a surprisingly simple randomized algorithm due to Freivalds [15] that is correct with probability at least 1/2: Pick a random vector $v \in \{0,1\}^n$, compute the matrix-vector products Cv and $A(Bv)$, and declare $AB = C$ if and only if $Cv = ABv$. Especially given the simplicity of this algorithm and the widely-shared hope that $\omega = 2$, one might conjecture that a deterministic version of Freivalds' algorithm exists. Alas, while refined ways to pick the random vector v reduce the required number of random bits to $\log n + \mathcal{O}(1)$ [32, 26], a $\tilde{\mathcal{O}}(n^2)$-time deterministic algorithms for matrix product verification remains elusive.

The motivation of this paper is the following question:

Can we solve Boolean, integer or real matrix multiplication in nondeterministic $\tilde{\mathcal{O}}(n^2)$ *time?*

Here we say that a functional problem f is in nondeterministic time $t(n)$ if f admits a $t(n)$-*time verifier*: there is a function v, computable in deterministic time $t(n)$, where n denotes the problem size of x, such that for all x, y there exists a certificate c with $v(x, y, c) = 1$ if and only $y = f(x)$.[1]

Note that a $\tilde{\mathcal{O}}(n^2)$-time derandomization of Freivalds' algorithm would yield an affirmative answer: guess C, and verify $AB = C$ using the deterministic verification algorithm. In contrast, a nondeterministic algorithm may guess additional information, a *certificate* beyond a guess C on the matrix product, and use it to verify that $C = AB$. Surprising faster algorithms in such settings have recently been found for 3SUM and all problems subcubic equivalent to APSP under deterministic reductions [11]; see [43, 42] for an overview over subcubic equivalences to APSP.

In this paper, we discuss consequences of positive or negative resolutions of this question, propose potential avenues for an affirmative answer and present partial algorithmic progress. In particular, we show that (1) sufficiently fast verifiers for 3SUM or univariate polynomial identity testing yield faster nondeterministic matrix multiplication algorithms, (2) in the integer case we can detect existence of between 1 and n erroneous entries in C in deterministic time $\tilde{\mathcal{O}}(n^2)$ and (3) we provide a novel deterministic output-sensitive integer matrix multiplication algorithm that improves upon previous deterministic algorithms if AB has at least $n^{2/3}$ nonzeroes.

1.1 Further Motivation and Consequences

Our motivation stems from studying the power of randomness, as well as algorithmic applications in certifiable computation, and consequences for the fine-grained complexity of polynomial-time problems.

[1] Throughout the paper, we view any decision problem P as a binary-valued functional problem. Thus a $t(n)$-time verifier for P shows that P is in nondeterministic *and* co-nondeterministic time $t(n)$.

Power of Randomness: Matrix-product verification has one of the simplest randomized solution for which no efficient derandomization is known – the currently best known deterministic algorithm simply computes the matrix product AB in deterministic time $\mathcal{O}(n^{\omega})$ and checks whether $C = AB$. Exploiting nondeterminism instead of randomization may yield insights into when and under which conditions we can derandomize algorithms without polynomial increases in the running time.

A very related case is that of univariate polynomial identity testing (UPIT): it has a similar status with regards to randomized and deterministic algorithms. As we will see, finding $\tilde{\mathcal{O}}(n^2)$-time nondeterministic derandomizations for UPIT is a more difficult problem, so that resolving our main question appears to be a natural intermediate step towards nondeterministic derandomizations of UPIT, see Section 1.2.

Practical Applications – Deterministic Certifying Algorithms: Informally, a *certifying algorithm* for a functional problem f is an algorithm that computes, for each input x, besides the desired output $y = f(x)$ also a certificate c such that there is a *simple* verifier that checks whether c proves that $y = f(x)$ indeed holds [31]. If we fix our notion of *simplicity* to be that of being computable by a fast deterministic algorithm, then our notion of verifiers turns out to be a suitable notion to study existence of certifying algorithms – it only disregards the running time needed to compute the certificate c.

Having a fast verifier for matrix multiplication would certainly be desirable – while Freivalds' algorithm yields a solution that is sufficient for many practical applications, it can never *completely* remove doubts on the correctness. Since matrix multiplication is a central ingredient for many problems, fast verifiers for matrix multiplication imply fast verifiers for many more problems.

In fact, even if $\omega = 2$, finding *combinatorial*[2] strongly subcubic verifiers is of interest, as these are more likely to yield practical advantages over more naive solutions. In particular, the known subcubic verifiers for all problems subcubic equivalent to APSP (under deterministic reductions) [11] all rely on fast matrix multiplication, and might not yet be relevant for practical applications.

Barriers for SETH-based Lower Bounds: Given the widely-shared hope that $\omega = 2$, can we rule out conditional lower bounds of the form $n^{c-o(1)}$ with $c > 2$ for matrix multiplication, e.g., based on the Strong Exponential Time Hypothesis (SETH) [19]? Carmosino et al. [11] proposed the Nondeterministic Strong Exponential Time Hypothesis (NSETH) that effectively postulates that there is no $\mathcal{O}(2^{(1-\varepsilon)n})$-time co-nondeterministic algorithm for k-SAT for all constant k. Under this assumption, we can rule out fast nondeterministic or co-nondeterministic algorithms for all problems that have *deterministic* fine-grained reductions from k-SAT. Conversely, if we find a nondeterministic matrix multiplication algorithm running in time $n^{c+o(1)}$, then NSETH implies that there is no SETH-based lower bound of $n^{c'-o(1)}$, with $c' > c$, for matrix multiplication using deterministic reductions.

Barriers for Reductions in Case of a Negative *Resolution*: Suppose that there is a negative resolution of our main question, specifically that Boolean matrix multiplication has no $n^{c-o(1)}$-time verifier for some $c > 2$ (observe that this would imply $\omega > 2$). Then by a simple $\mathcal{O}(n^2)$-time nondeterministic reduction from Boolean matrix multiplication to triangle finding (implicit in the proof of Theorem 1.1 below) and a known $\mathcal{O}(n^2)$-time reduction from triangle finding to Radius [1], Radius has no $n^{c-o(1)}$-time verifier. This state of affairs would rule out certain kinds of subcubic reductions from Radius to Diameter, e.g., deterministic

[2] Throughout this paper, we call an algorithm *combinatorial*, if it does not use sophisticated algebraic techniques underlying the fastest known matrix multiplication algorithms.

many-one-reductions, since these would transfer a simple $\mathcal{O}(n^2)$-time verifier for Diameter[3] to Radius. Note that finding a subcubic reduction from Radius to Diameter is an open problem in the fine-grained complexity community [1].

1.2 Structural Results: Avenues Via Other Problems

We present two particular avenues for potential subcubic or even near-quadratic matrix multiplication verifiers: finding fast verifiers for either 3SUM or univariate polynomial identity testing.

3SUM

One of the core hypotheses in the field of hardness in P is the 3SUM problem [16]. Despite the current best time bound of $\mathcal{O}(n^2 \cdot \frac{\operatorname{poly}\log\log n}{\log^2 n})$ [6, 12] being only slightly subquadratic, recently a strongly subquadratic verifier running in time $\tilde{\mathcal{O}}(n^{3/2})$ was found [11]. We have little indication to believe that this verification time is optimal; for the loosely related computational model of decision trees, a remarkable near-linear time bound has been obtained just this year [25].

By a simple reduction, we obtain that any polynomial speedup over the known 3SUM verifier yields a subcubic Boolean matrix multiplication verifier. In particular, establishing a near-linear 3SUM verifier would yield a positive answer to our main question in the Boolean setting.

▶ **Theorem 1.1.** *Any $\mathcal{O}(n^{3/2-\varepsilon})$-time verifier for* 3SUM *yields a $\mathcal{O}(n^{3-2\varepsilon})$-time verifier for Boolean matrix multiplication.*

Under the BMM hypothesis, which asserts that there is no combinatorial $\mathcal{O}(n^{3-\varepsilon})$-time algorithm for Boolean matrix multiplication (see, e.g., [2]), a $n^{3/2-o(1)}$-time lower bound (under randomized reductions) for combinatorial 3SUM algorithms is already known [22, 43]. The above result, however, establishes a stronger, non-randomized relationship between the verifiers' running times by a simple proof exploiting nondeterminism.

UPIT

Univariate polynomial identity testing (UPIT) asks to determine, given two degree-n polynomials p, q over a finite field of polynomial order, represented as arithmetic circuits with $\mathcal{O}(n)$ wires, whether p is identical to q. By evaluating and comparing p and q at $n+1$ distinct points or $\tilde{\mathcal{O}}(1)$ random points, we can solve UPIT deterministically in time $\tilde{\mathcal{O}}(n^2)$ or with high probability in time $\tilde{\mathcal{O}}(n)$, respectively. A nondeterministic derandomization, more precisely, a $\mathcal{O}(n^{2-\varepsilon})$-time verifier, would have interesting consequences [47]: it would refute the Nondeterministic Strong Exponential Time Hypothesis posed by Carmosino et al. [11], which in turn would prove novel circuit lower bounds, deemed difficult to prove. We observe that a sufficiently strong nondeterministic derandomization of UPIT would also give a faster matrix multiplication verifier.

---

[3] We verify that a graph G has diameter d as follows: For every vertex v, we guess the shortest path tree originating in v. It is straightforward to use this tree to verify that all vertices v' have distance at most d from v in time $\mathcal{O}(n)$. Thus, we can prove that the diameter is at most d in time $\mathcal{O}(n^2)$. For the lower bound, guess some vertex pair u, v and verify that their distance is indeed d using a single-source shortest path computation in time $\mathcal{O}(m + n\log n) = \mathcal{O}(n^2)$.

▶ **Theorem 1.2.** *Any $\mathcal{O}(n^{3/2-\varepsilon})$-time verifier for* UPIT *yields a $\mathcal{O}(n^{3-2\varepsilon})$-time verifier for integer matrix multiplication.*

Note that this avenue might seem more difficult to pursue than a direct attempt at resolving our main question, due to its connection to NSETH and circuit lower bounds. Alternatively, however, we can view the specific arithmetic circuit obtained in our reductions as an interesting intermediate testbed for ideas towards derandomizing UPIT. In fact, our algorithmic results were obtained by exploiting the connection to UPIT, and exploiting the structure of the resulting specialized circuits/polynomials.

1.3 Algorithmic Results: Progress on Integer Matrix Product Verification

Our main result is partial algorithmic progress towards the conjecture in the integer setting. Specifically, we consider a restriction of MM-Verification to the case of detecting a bounded number t of errors. Formally, let MM-Verification$_t$ denote the following problem: given $n \times n$ integer matrices A, B, C with polynomially bounded entries, produce an output "$C = AB$" or "$C \neq AB$", where the output must always be correct if C and AB differ in at most t entries.

Our main result is an algorithm that solves MM-Verification$_t$ in near-quadratic time for $t = \mathcal{O}(n)$ and in strongly subcubic time for $t = \mathcal{O}(n^c)$ with $c < 2$.

▶ **Theorem 1.3.** *For any $1 \leq t \leq n^2$,* MM-Verification$_t$ *can be solved deterministically in time $O((n^2 + tn)\log^{2+o(1)} n)$.*

Interestingly, this shows that detecting the presence of very few errors is not a difficult case. Instead of a needle-in-the-haystack problem, we rather need to find a way to deal with cancellation effects in the presence of at least $\Omega(n)$ errors.

As a corollary, we obtain a different near-quadratic-time randomized algorithm for MM-Verification than Freivalds' algorithm: Run the algorithm of Theorem 1.3 for $t = n$ in time $\tilde{\mathcal{O}}(n^2)$. Afterwards, either $C = AB$ holds or C has at least $\Omega(n)$ erroneous entries. Thus it suffices to sample $\Theta(n)$ random entries i, j and to check whether $C_{i,j} = (AB)_{i,j}$ for all sampled entries (by naive computation of $(AB)_{i,j}$ in time $\mathcal{O}(n)$ each) to obtain an $\tilde{\mathcal{O}}(n^2)$-time algorithm that correctly determines $C = AB$ or $C \neq AB$ with constant probability. Potentially, this alternative to Freivalds' algorithm might be simpler to derandomize.

Finally, our algorithm for *detecting* up to t errors can be extended to a more involved algorithm that also *finds* all erroneous entries (if no more than t errors are present) and *correct* them in time $\tilde{\mathcal{O}}(\sqrt{t}n^2 + t^2)$. In fact, this problem turns out to be equivalent to the notion of output-sensitive matrix multiplication os-MM$_t$: Given $n \times n$ matrices A, B of polynomially bounded integer entries with the promise that AB contains at most t nonzeroes, compute AB.

▶ **Theorem 1.4.** *Let $1 \leq t \leq n^2$. Given $n \times n$ matrices A, B, C of polynomially bounded integers, with the property that C differs from AB in at most t entries, we can compute AB in time $\mathcal{O}(\sqrt{t}n^2 \log^{2+o(1)} n + t^2 \log^{3+o(1)} n)$. Equivalently, we can solve* os-MM$_t$ *in time $\mathcal{O}(\sqrt{t}n^2 \log^{2+o(1)} n + t^2 \log^{3+o(1)} n)$.*

Previous work by Gasieniec et al. [17] gives a $\tilde{\mathcal{O}}(n^2 + tn)$ *randomized* solution, as well as a $\tilde{\mathcal{O}}(tn^2)$ deterministic solution. Because of the parameter-preserving equivalence between t error correction and os-MM$_t$, this task is also solved by the randomized $\tilde{\mathcal{O}}(n^2 + tn)$-time algorithm due to Pagh [34][4] and the deterministic $\mathcal{O}(n^2 + t^2 n \log^5 n)$-time algorithm due to Kutzkov [28]. Note that our algorithm improves upon Kutzkov's algorithm for $t = \Omega(n^{2/3})$, in particular, our algorithm is strongly subcubic for $t = \mathcal{O}(n^{3/2-\varepsilon})$ and even improves upon the best known fast matrix multiplication algorithm for $t = \mathcal{O}(n^{0.745})$.

1.4 Further Related Work

There is previous work that claims to have resolved our main question in the affirmative. Unfortunately, the approach is flawed; we detail the issue in the full version of this article [27].

Other work considers MM-Verification and os-MM in quantum settings, e.g., [10, 23]. Furthermore, better running times can be obtained if we restrict the distribution of the errors over the guessed matrix/nonzeroes over the matrix product: Using rectangular matrix multiplication, Iwen and Spencer [20] show how to compute AB in time $\mathcal{O}(n^{2+\varepsilon})$ for any $\varepsilon > 0$, if no column (or no row) of AB contains more than $n^{0.29462}$ nonzeroes. Furthermore, Roche [35] gives a randomized algorithm refining the bound of Gasieniec et al. [17] using, as additional parameters, the total number of nonzeroes in A, B, C and the number of distinct columns/rows containing an error.

For the case of Boolean matrix multiplication, several output-sensitive algorithms are known [36, 48, 5, 30], including a simple deterministic $\mathcal{O}(n^2 + tn)$-time algorithm [36] and, exploiting fast matrix multiplication, a randomized $\tilde{\mathcal{O}}(n^2 t^{\omega/2-1})$-time solution [30]. Note that in the Boolean setting, our parameter-preserving reduction from error correction to output-sensitive multiplication (Proposition 3.1) no longer applies, so that these algorithms unfortunately do not immediately yield error correction algorithms.

1.5 Paper Organization

After collecting notational conventions and introducing polynomial multipoint evaluation as our main algorithmic tool in Section 2, we give a high-level description over the main ideas behind our results in Section 3. We prove our structural results in Section 4. Our first algorithmic result on error detection is proven in Section 5. Unfortunately, the details for our technically most demanding result, i.e., Theorem 1.4, had to be omitted due to space constraints – they are available in the full version of this article [27]. We conclude with open questions in Section 6.

2 Preliminaries

Recall the definition of a $t(n)$-time verifier for a functional problem f: there is a function v, computable in deterministic time $t(n)$ with n being the problem size of x, such that for all x, y there exists a certificate c with $v(x, y, c) = 1$ if and only $y = f(x)$. Here, we assume the word RAM model of computation with a word size $w = \Theta(\log n)$.

For n-dimensional vectors a, b over the integers, we write their inner product as $\langle a, b\rangle = \sum_{k=1}^{n} a[k] \cdot b[k]$, where $a[k]$ denotes the k-th coordinate of a. For any matrix X, we write $X_{i,j}$ for its value at row i, column j. We typically represent the $n \times n$ matrix A by its n-dimensional row vectors $a_1, \ldots, a_n$, and the $n \times n$ matrix B by its n-dimensional column vectors $b_1, \ldots, b_n$ such that $(AB)_{i,j} = \langle a_i, b_j\rangle$. For any $I \subseteq [n], J \subseteq [n]$, we obtain a submatrix $(AB)_{I,J}$ of AB by deleting from AB all rows not in I and all columns not in J.

[4] For $t = \omega(n)$, Jacob and Stöckel [21] give an improved randomized $\tilde{\mathcal{O}}(n^2 (t/n)^{\omega-2})$-time algorithm.

Fast Polynomial Multipoint Evaluation

Consider any finite field $\mathbb{F}$ and let $M(d)$ be the number of additions and multiplications in $\mathbb{F}$ needed to multiply two degree-d univariate polynomials. Note that $M(d) = \mathcal{O}(d \log d \log\log d) = \mathcal{O}(d \log^{1+o(1)} n)$, see, e.g. [44].

▶ **Lemma 2.1** (Multipoint Polynomial Evaluation [14])**.** *Let $\mathbb{F}$ be an arbitrary field. Given a degree-d polynomial $p \in \mathbb{F}[X]$ given by a list of its coefficients $(a_0, \ldots, a_d) \in \mathbb{F}^{d+1}$, as well as input points $x_1, \ldots, x_d \in \mathbb{F}$, we can determine the list of evaluations $(p(x_1), \ldots, p(x_d)) \in \mathbb{F}^n$ using $\mathcal{O}(M(d) \log d)$ additions and multiplications in $\mathbb{F}$.*

Thus, we can evaluate p on any list of inputs $x_1, \ldots, x_n$ in time $\mathcal{O}((n + d) \log^{2+o(1)} d)$.

3 Technical Overview

We first observe a simple parameter-preserving equivalence of the following problems,

MM-Verification$_t$ Given $\ell \times n, n \times \ell, \ell \times \ell$ matrices A, B, C such that AB and C differ in $0 \le z \le t$ entries, determine whether $AB = C$, i.e., $z = 0$,

AllZeroes$_t$ Given $\ell \times n, n \times \ell$ matrices A, B such that AB has $0 \le z \le t$ nonzeroes, determine whether $AB = 0$, i.e., $z = 0$.

We also obtain a parameter-preserving equivalence of their "constructive" versions,

MM-Correction$_t$ Given $\ell \times n, n \times \ell, \ell \times \ell$ matrices A, B, C such that AB and C differ in $0 \le z \le t$ entries, determine AB,

os-MM$_t$ Given $\ell \times n, n \times \ell$ matrices A, B such that AB has $0 \le z \le t$ nonzeroes, determine AB.

For any problem P_t among the above, let $T_P(n, \ell, t)$ denote the optimal running time to solve P_t with parameters n, ℓ and t.

▶ **Proposition 3.1.** *Let $\ell \le n$ and $1 \le t \le n^2$. We have*

$$T_{\text{MM-Verification}}(n, \ell, t) = \Theta(T_{\text{AllZeroes}}(n, \ell, t))$$
$$T_{\text{MM-Correction}}(n, \ell, t) = \Theta(T_{\text{os-MM}}(n, \ell, t)).$$

Proof. By setting $C = 0$, we can reduce AllZeroes$_t$ and os-MM$_t$ to MM-Verification$_t$ and MM-Correction$_t$, respectively, achieving the lower bounds of the claim.

For the other direction, let $a_1, \ldots, a_\ell \in \mathbb{Z}^n$ be the row vectors of A, $b_1, \ldots, b_\ell \in \mathbb{Z}^n$ be the column vectors of B and $c_1, \ldots, c_\ell \in \mathbb{Z}^\ell$ be the column vectors of C. Let e_i denote the vector whose i-th coordinate is 1 and whose other coordinates are 0. We define $\ell \times (n+\ell), (n+\ell) \times \ell$ matrices A', B' by specifying the row vectors of A' as

$$a'_i = (a_i, -e_i),$$

and the column vectors of B' as

$$b'_j = (b_j, c_j).$$

Note that $(A'B')_{i,j} = \langle a'_i, b'_j \rangle = \langle a_i, b_j \rangle - c_j[i]$, thus $(A'B')_{i,j} = 0$ if and only if $(AB)_{i,j} = C_{i,j}$. Consequently, $A'B'$ has at most t nonzeroes, and checking equality of $A'B'$ to the all-zero matrix is equivalent to checking $AB = C$. The total time to solve MM-Verification$_t$ is thus bounded by $\mathcal{O}((n + \ell)\ell) + T_{\text{AllZeroes}}(n + \ell, \ell, t) = \mathcal{O}(T_{\text{AllZeroes}}(n, \ell, t))$, as desired.

Furthermore, by computing $C' = A'B'$ (which contains at most t nonzero entries), we can also correct the matrix product C by updating $C_{i,j}$ to $C_{i,j} + C'_{i,j}$. This takes time $\mathcal{O}((n + \ell)\ell) + T_{\text{os-MM}}(n + \ell, \ell, t) = \mathcal{O}(T_{\text{os-MM}}(n, \ell, t))$, as desired. ◀

Because of the above equivalence, we can focus on solving $\textsc{AllZeroes}_t$ and $\textsc{os-MM}_t$ in the remainder of the paper. The key for our approach is the following multilinear polynomial

$$f_{\mathrm{MM}}^{A,B}(x_1,\ldots,x_\ell;y_1,\ldots,y_\ell) := \sum_{i,j\in[\ell]} x_i \cdot y_j \cdot \langle a_i, b_j\rangle,$$

where again the $a_1,\ldots,a_\ell$ denote the row vectors of A and the $b_1,\ldots,b_\ell$ denote the column vectors of B. Note that the nonzero monomials of $f_{\mathrm{MM}}^{A,B}$ correspond directly to the nonzero entries of AB. We introduce a univariate variant

$$g(X) = g^{A,B}(X) := f_{\mathrm{MM}}^{A,B}(1, X, \ldots, X^{\ell-1}; 1, X^\ell, \ldots, X^{\ell(\ell-1)}),$$

which has the helpful property that monomials $x_i y_j$ of f_{MM} are mapped to the monomial $X^{(i-1)+\ell(j-1)}$ in a one-to-one manner, preserving coefficients. To obtain a more efficient representation of g than to explicitly compute all coefficients $\langle a_i, b_j\rangle$, we can exploit linearity of the inner product: we have $g(X) = \sum_{k=1}^{n} q_k(X) r_k(X^\ell)$, where $q_k(Z) = \sum_{i=1}^{\ell} a_i[k] Z^{i-1}$ and $r_k(Z) = \sum_{j=1}^{\ell} b_j[k] Z^{j-1}$. This representation is more amenable for efficient evaluation, and immediately yields a reduction to univariate polynomial identity testing (UPIT) (see Theorem 4.2 in Section 4).

To solve the detection problem, we use an idea from sparse polynomial interpolation [7, 49]: If AB has at most t nonzeroes, then for any root of unity ω of sufficiently high order, $g(\omega^0) = g(\omega^1) = g(\omega^2) = \cdots = g(\omega^{t-1}) = 0$ is equivalent to $AB = 0$. By showing how to do fast batch evaluation of g using the above representation, we obtain an $\tilde{\mathcal{O}}((\ell+t)n)$-time algorithm for $\textsc{AllZeroes}_t$ in Section 5, proving Theorem 1.3.

Towards solving the correction problem, the naive approach is to use the $\tilde{\mathcal{O}}((\ell+t)n)$-time $\textsc{AllZeroes}_t$ algorithm in combination with a self-reduction to obtain a fast algorithm for finding a nonzero position (i,j) of AB: If the $\textsc{AllZeroes}$ algorithm determines that AB contains at least one nonzero entry, we split the product matrix AB into four submatrices, detect any one of them containing a nonzero entry, and recurse on it. After finding such an entry, one can compute the correct nonzero value $(AB)_{i,j} = \langle a_i, b_j\rangle$ in time $\mathcal{O}(n)$. One can then "remove" this nonzero from further search (analogously to Proposition 3.1) and iterate this process. Unfortunately, this only yields an algorithm of running time $\tilde{\mathcal{O}}(tn^2)$, even if $\textsc{AllZeroes}$ would take near-optimal time $\tilde{\mathcal{O}}(n^2)$. A faster alternative is to use the self-reduction such that we find *all* nonzero entries whenever we recurse on a submatrix containing at least one nonzero value. However, this process only leads to a running time of $\tilde{\mathcal{O}}(\sqrt{t}n^2 + nt^2)$. Here, the bottleneck $\tilde{\mathcal{O}}(nt^2)$ term stems from the fact that performing an $\textsc{AllZeroes}$ test for t submatrices (e.g., when t nonzeroes are spread evenly in the matrix) takes time $t \cdot \tilde{\mathcal{O}}(nt)$.

We still obtain a faster algorithm by a rather involved approach: The intuitive idea is to test submatrices for appropriately smaller number of nonzeroes $z \ll t$. At first sight, such an approach might seem impossible, since we can only be certain that a submatrix contains no nonzeroes if we test it for the full number t of potential nonzeroes. However, by showing how to reuse and quickly update previously computed information after finding a nonzero, we make this approach work by obtaining "global" information at a small additional cost of $\tilde{\mathcal{O}}(t^2)$. Doing these dynamic updates quickly crucially relies on the efficient representation of the polynomial g. The details are given in the full version of this article [27].

4 Structural Results: Avenues Via Other Problems

In this section, we show the simple reductions translating verifiers for 3SUM or UPIT to matrix multiplication.

4.1 3SUM

We consider the following formulation of the 3SUM problem: given sets S_1, S_2, S_3 of polynomially bounded integers, determine whether there exists a triplet $s_1 \in S_1, s_2 \in S_2, s_3 \in S_3$ with $s_1 + s_2 = s_3$. It is known that a combinatorial $\mathcal{O}(n^{3/2-\varepsilon})$-time algorithm for 3SUM (for any $\varepsilon > 0$) yields a combinatorial $\mathcal{O}(n^{3-\varepsilon'})$-time Boolean matrix multiplication (BMM) algorithm (for some $\varepsilon' > 0$). This follows by combining a reduction from Triangle Detection to 3SUM of [22] and using the combinatorial subcubic equivalence of Triangle Detection and BMM [43][5]. While this only yields a *nontight* BMM-based lower bound for 3SUM for *deterministic or randomized* combinatorial algorithms, we can establish a *tight* relationship for the current state of knowledge of combinatorial verifiers. In fact, allowing nondeterminism, we obtain a very simple direct proof of a stronger relationship of the running times than known for deterministic reductions.

▶ **Theorem 4.1.** *If 3SUM admits a ("combinatorial") $\mathcal{O}(n^{3/2-\varepsilon})$-time verifier, then* BMM *admits a ("combinatorial") $\mathcal{O}(n^{3-2\varepsilon})$-time verifier.*[6]

Thus, significant combinatorial improvements over Carmosino et al.'s 3SUM verifier yield strongly subcubic combinatorial BMM verifiers. In particular, a $\tilde{\mathcal{O}}(n)$-time verifier for 3SUM would yield an affirmative answer to our main question in the Boolean setting. Note that an analogous improvement of the $\mathcal{O}(n^{3/2}\sqrt{\log n})$ [18] size bound in the decision tree model to a size of $\mathcal{O}(n \log^2 n)$ has recently been obtained [25].

To establish this strong relationship, our reduction exploits the nondeterministic setting – without nondeterminism, no reduction is known that would give a $\mathcal{O}(n^{\frac{8}{3}-\varepsilon})$-time BMM algorithm even if 3SUM could be solved in an optimal $\mathcal{O}(n)$ time bound.

Proof of Theorem 4.1. Given the $n \times n$ Boolean matrices A, B, C, we first check whether all entries (i,j) with $C_{i,j} = 1$ are correct. For this, for each such i, j, we guess a witness k and check that $A_{i,k} = B_{k,j} = 1$, which verifies that $C_{i,j} = (AB)_{i,j} = 1$.

To check the remaining zero entries $Z = \{(i,j) \in [n]^2 \mid C_{i,j} = 0\}$, we construct a 3SUM instance S_1, S_2, S_3 as follows. Let $W = 2(n+1)$. For each $(i,j) \in Z$, we include $iW^2 + jW$ in our set S_3. For every (i,k) with $A_{i,k} = 1$, we include $iW^2 + k$ in our set S_1, and, for every (k,j) with $B_{k,j} = 1$, we include $jW - k$ in our set S_2. Clearly, any witness $A_{i,k} = B_{k,j} = 1$ for $(AB)_{i,j} = 1$, $(i,j) \in Z$ yields a triplet $a = iW^2 + k \in S_1, b = jW - k \in S_2, c = iW^2 + jW \in S_3$ with $a + b = c$. Conversely, any 3SUM triplet $a \in S_1, b \in S_2, c \in S_3$ yields a witness for $(AB)_{i,j} = 1$, where $(i,j) \in Z$ is the zero entry represented by c, since $(iW^2 + k) + (jW - k') = i'W^2 + j'W$ for $i, i', j, j', k, k' \in [n]$ if only if $i = i', j = j'$ and $k = k'$ by choice of W. Thus, the 3SUM instance is a NO instance if and only if no $(i,j) \in Z$ has a witness for $(AB)_{i,j} = 1$, i.e., all $(i,j) \in Z$ satisfy $C_{i,j} = (AB)_{i,j} = 0$.

Note that reduction runs in nondeterministic time $\mathcal{O}(n^2)$, using an oracle call of a 3SUM instance of size $\mathcal{O}(n^2)$, which yields the claim. ◀

[5] K. G. Larsen obtained an independent proof of this fact, see `https://simons.berkeley.edu/talks/kasper-larsen-2015-12-01`.

[6] Strictly speaking, the notion of a "combinatorial" algorithm is not well-defined, hence we use quotes here. However, our reductions are so simple that they should qualify under any reasonable exact definition.

4.2 UPIT

Univariate Polynomial Identity Testing (UPIT) is the following problem: Given arithmetic circuits Q, Q' on a single variable, with degree n and $O(n)$ wires, over a field of order poly(n), determine whether $Q \equiv Q'$, i.e., the outputs of Q and Q' agree on all inputs. Using evaluation on $n+1$ distinct points, we can deterministically solve UPIT in time $\tilde{\mathcal{O}}(n^2)$, while evaluating on $\tilde{\mathcal{O}}(1)$ random points yields a randomized solution in time $\tilde{\mathcal{O}}(n)$. Williams [47] proved that a $\mathcal{O}(n^{2-\varepsilon})$-time deterministic UPIT algorithm refutes the Nondeterministic Strong Exponential Time Hypothesis posed by Carmosino et al. [11]. We establish that a sufficiently strong (nondeterministic) derandomization of UPIT also yields progress on MM-Verification.

▶ **Theorem 4.2.** *If* UPIT *admits a ("combinatorial") $\mathcal{O}(n^{3/2-\varepsilon})$-time verifier for some $\varepsilon > 0$, then there is a ("combinatorial") $\mathcal{O}(n^{3-2\varepsilon})$-time verifier for matrix multiplication over polynomially bounded integers and over finite fields of polynomial order.*

Proof. We only give the proof for matrix multiplication over a finite field $\mathbb{F}$ of polynomial order. Using Chinese Remaindering, we can easily extend the reduction to the integer case (see Proposition 5.3 below).

Consider $g(X) = \sum_{i,j\in[n]} \langle a_i, b_j\rangle X^{(i-1)+n(j-1)}$ over $\mathbb{F}$ as defined in Section 3 (with $\ell = n$). As described there, we can write $g(X) = \sum_{k=1}^{n} q_k(X) r_k(X^n)$ with $q_k(Z) = \sum_{i=1}^{n} a_i[k] Z^{i-1}$ and $r_k(Z) = \sum_{j=1}^{n} b_j[k] Z^{j-1}$. Let $k \in [n]$ and note that q_k, r_k and X^n have arithmetic circuits with $\mathcal{O}(n)$ wires using Horner's scheme. Chaining the circuits of X^n and r_k, and multiplying with the output of the circuit for q_k, we obtain a degree-$\mathcal{O}(n^2)$ circuit Q_k with $\mathcal{O}(n)$ wires. It remains to sum up the outputs of the circuits $Q_1, \ldots, Q_n$. We thus obtain a circuit Q with $\mathcal{O}(n^2)$ wires and degree $\mathcal{O}(n^2)$. Since by construction $AB = 0$ if and only $Q \equiv 0$, we obtain an UPIT instance Q, Q', with Q' being a constant-sized circuit with output 0, that is equivalent to our MM-Verification instance. Thus, any $\mathcal{O}(n^{3/2-\varepsilon})$-time algorithm for UPIT would yield a $\mathcal{O}(n^{2(3/2-\varepsilon)})$-time MM-Verification algorithm, as desired. ◀

It is known that refuting NSETH implies strong circuit lower bounds [11], so pursuing this route might seem much more difficult than attacking MM-Verification directly. However, to make progress on MM-Verification, we only need to nondeterministically derandomize UPIT for very specialized circuits. In this direction, our algorithmic results exploit that we can derandomize UPIT for these specialized circuits, as long as they represent *sparse* polynomials.

5 Deterministically Detecting Presence of $0 < z \le t$ Errors

In this section we prove the first of our main algorithmic results, i.e., Theorem 1.3.

▶ **Theorem 5.1.** *For any $1 \le t \le n^2$,* MM-Verification$_t$ *can be solved deterministically in time $O((n^2 + tn)\log^{2+o(1)}(n))$.*

We prove the claim by showing how to solve the following problem in time $\tilde{\mathcal{O}}((\ell + t)n)$.

▶ **Lemma 5.2.** *Let $\mathbb{F}_p$ be a prime field with a given element $\omega \in \mathbb{F}_p$ of order at least ℓ^2. Let A, B be $\ell \times n, n \times \ell$-matrices over $\mathbb{F}_p$. There is an algorithm running in time $\mathcal{O}((\ell + t)n \log^{2+o(1)} n)$ with the following guarantees:*

1. *If $AB = 0$, the algorithm outputs "$AB = 0$".*
2. *If AB has $0 < z \le t$ nonzeroes, the algorithm outputs "$AB \neq 0$".*

Given such an algorithm working over finite fields, we can check matrix products of integer matrices using the following proposition.

▶ **Proposition 5.3.** *Let A, B be $n \times n$ matrices over the integers of absolute values bounded by n^c for some $c \in \mathbb{N}$. Then we can find, in time $\mathcal{O}(n^2 \log n)$, distinct primes $p_1, p_2, \dots, p_d$ and corresponding elements $\omega_1 \in \mathbb{F}_{p_1}, \omega_2 \in \mathbb{F}_{p_2}, \dots, \omega_d \in \mathbb{F}_{p_d}$, such that*

i) *$AB = 0$ if and only if $AB = 0$ over $\mathbb{F}_{p_i}$ for all $1 \le i \le d$,*
ii) *$d = \mathcal{O}(1)$, and*
iii) *for each $1 \le i \le d$, we have $p_i = \mathcal{O}(n^2)$ and ω_i has order at least n^2 in $\mathbb{F}_{p_i}$.*

Note that the obvious approach of choosing a single prime field $\mathbb{F}_p$ with $p \ge n^{2c+1}$ is not feasible for our purposes: the best known deterministic algorithm to find such a prime takes time $n^{c/2+o(1)}$ (see [39] for a discussion), quickly exceeding our desired time bound of $\mathcal{O}(n^2)$.

Proof of Proposition 5.3. Let $d = c + 1$ and note that any entry $(AB)_{i,j} = \sum_{k=1}^{n} A_{i,k} B_{k,j}$ is in $[-n^{2c+1}, n^{2c+1}]$. Thus for any number $m > n^{2c+1}$, we have $(AB)_{ij} \equiv 0 \pmod m$ if and only if $(AB)_{i,j} = 0$. By Chinese Remaindering, we obtain that any distinct primes $p_1, \dots, p_d$ with $p_i \ge n^2$ satisfy i) and ii), as $AB = 0$ if and only if $AB = 0$ over $\mathbb{F}_{p_i}$ for all $1 \le i \le d$, using the fact that $\prod_{i=1}^{d} p_i \ge n^{2d} > n^{2c+1}$.

By Bertrand's postulate, there are at least d primes in the range $\{n^2 + 1, \dots, 2^d(n^2 + 1)\}$, thus using the sieve of Eratosthenes, we can find $p_1, \dots, p_d$ with $p_i \ge n^2 + 1$ and $p_i \le 2^d(n^2 + 1)$ in time $\mathcal{O}(n^2 \log \log n)$ (see [44, Theorem 18.10]). It remains to find elements $\omega_1 \in \mathbb{F}_{p_1}, \dots, \omega_d \in \mathbb{F}_{p_d}$ of sufficiently high order. For each $1 \le j \le d$, this can be achieved in time $\mathcal{O}(n^2 \log n)$ by exhaustive testing: We keep a list $L \subseteq \mathbb{F}_{p_j}^\times = \mathbb{F}_{p_j} \setminus \{0\}$ of "unencountered" elements, which we initially set to $\mathbb{F}_{p_j}^\times$. Until there are no elements in L remaining, we pick any $\alpha \in L$ and delete all elements in the subgroup of $\mathbb{F}_{p_j}^\times$ generated by α from L. We set ω_j to the last α that we picked (which has to generate the complete multiplicative group $\mathbb{F}_{p_j}^\times$) and thus is a primitive $(p_j - 1)$-th root of unity. Since $p_j - 1 \ge n^2$, the order of ω_j is at least n^2, as desired. Observe that the number of iterations is bounded by the number of subgroups of $\mathbb{F}_{p_j}^\times$, i.e., the number of divisors of $p_j - 1$. Thus, we have at most $\mathcal{O}(\log p_j)$ iterations, each taking time at most $\mathcal{O}(p_j)$, yielding a running time of $\mathcal{O}(p_j \log p_j) = \mathcal{O}(n^2 \log n)$. ◀

Combining Proposition 3.1 with the algorithm of Lemma 5.2 and Proposition 5.3, we obtain the theorem.

Proof of Theorem 5.1. Given any instance A, B, C of MM-VERIFICATION$_t$, we convert it to an instance A', B' of ALLZEROES as in Proposition 3.1. We construct primes $p_1, \dots, p_d$ as in Proposition 5.3 in time $\mathcal{O}(n^2 \log n)$. For each $j \in [d]$, we convert A', B' to matrices over $\mathbb{F}_{p_j}$ in time $\mathcal{O}(n^2)$ and test whether $A'B' = 0$ over $\mathbb{F}_{p_j}$ for all $j \in [d]$ using Lemma 5.2 in time $\mathcal{O}((n^2 + tn) \log^{2+o(1)} n)$. We output "$AB = C$" if and only if all tests succeeded. Correctness follows from Proposition 5.3 and Lemma 5.2, and the total running time is $\mathcal{O}((n^2 + tn) \log^{2+o(1)} n)$, as desired. ◀

In the remainder, we prove Lemma 5.2. As outlined in Section 3, define the polynomial $g(X) = \sum_{i,j \in [\ell]} \langle a_i, b_j \rangle X^{(i-1)+\ell(j-1)}$ over $\mathbb{F}_p$. We aim to determine whether $g \equiv 0$. To do so, we use the following idea from Ben-Or and Tiwari's approach to black-box sparse polynomial interpolation (see [7, 49]). Suppose that $\omega \in \mathbb{F}_p$ has order at least ℓ^2. Then the following proposition holds.

▶ **Proposition 5.4.** *Assume AB has $0 \le z \le t$ nonzeroes. Then $g(\omega^0) = g(\omega) = g(\omega^2) = \cdots = g(\omega^{t-1}) = 0$ if and only if $g \equiv 0$, i.e., $z = 0$.*

Proof. By assumption on A, B, we have $g(X) = \sum_{m \in M} c_m X^m$, where $M = \{(i-1)+\ell(j-1) \mid \langle a_i, b_j\rangle \neq 0\}$ with $|M| = z \leq t$ and $c_{(i-1)+\ell(j-1)} = \langle a_i, b_j\rangle$. Writing $M = \{m_1, \dots, m_z\}$ and defining $v_m = \omega^m$, we see that $g(\omega^0) = \dots = g(\omega^{t-1}) = 0$ is equivalent to

$$\begin{aligned}
c_{m_1} \quad &+ \dots + c_{m_z} &&= 0,\\
c_{m_1} v_{m_1} &+ \dots + c_{m_z} v_{m_z} &&= 0,\\
c_{m_1} v_{m_1}^2 &+ \dots + c_{m_z} v_{m_z}^2 &&= 0,\\
& &&\dots\\
c_{m_1} v_{m_1}^{t-1} &+ \dots + c_{m_z} v_{m_z}^{t-1} &&= 0.
\end{aligned}$$

Since ω has order at least ℓ^2, we have that $v_m = \omega^m \neq \omega^{m'} = v_{m'}$ for all $m, m' \in M$ with $m \neq m'$. Thus the above system is a Vandermonde system with unique solution $(c_{m_1}, \dots, c_{m_z}) = (0, \dots, 0)$, since $z \leq t$. This yields the claim. ◀

It remains to compute $g(\omega^0), \dots, g(\omega^{t-1})$ in time $\tilde{\mathcal{O}}((\ell + t)n)$.

▶ **Proposition 5.5.** *For any $\sigma_1, \dots, \sigma_t \in \mathbb{F}_p$, we can compute $g(\sigma_1), \dots, g(\sigma_t)$ in time $\mathcal{O}((\ell + t)n \log^{2+o(1)} \ell)$.*

Proof. Recall that $g(X) = \sum_{k=1}^n q_k(X) \cdot r_k(X^\ell)$, where $q_k(Z) = \sum_{i=1}^\ell a_i[k] Z^{i-1}$ and $r_k(Z) = \sum_{j=1}^\ell b_j[k] Z^{j-1}$. Let $1 \leq k \leq n$. Using fast multipoint evaluation (Lemma 2.1), we can compute $q_k(\sigma_1), \dots, q_k(\sigma_t)$ using $\mathcal{O}((\ell + t) \log^{2+o(1)} \ell)$ additions and multiplications in $\mathbb{F}_p$. Furthermore, since we can compute $\sigma_1^\ell, \dots, \sigma_t^\ell$ using $\mathcal{O}(t \log \ell)$ additions and multiplications in $\mathbb{F}_p$, we can analogously compute $r_k(\sigma_1^\ell), \dots, r_k(\sigma_t^\ell)$ in time $\mathcal{O}((\ell + t) \log^{2+o(1)} \ell)$. Doing this for all $1 \leq k \leq n$ yields all values $q_k(\sigma_u), r_k(\sigma_u^\ell)$ with $k \in [n], u \in [t]$ in time $\mathcal{O}((\ell + t)n \log^{2+o(1)} \ell)$. We finally aggregate these values to obtain the desired outputs $g(\sigma_u) = \sum_{k=1}^n q_k(\sigma_u) \cdot r_k(\sigma_u^\ell)$ with $u \in [t]$. The aggregation only uses $\mathcal{O}(tn)$ multiplications and additions in $\mathbb{F}_p$, thus the claim follows. ◀

Together with Proposition 5.4, this yields Lemma 5.2 and thus the remaining step of the proof of Theorem 5.1.

6 Open Questions

It remains to answer our main question. To this end, can we exploit any of the avenues presented in this work? In particular: Can we (1) find a faster 3SUM verifier, (2) find a faster UPIT algorithm for the circuits given in Theorem 4.2, or (3) instead of derandomizing Freivalds' algorithm, nondeterministically derandomize the sampling-based algorithm following from our main algorithmic result (which detects up to $\mathcal{O}(n)$ errors using Theorem 1.3, and then samples and checks $\Theta(n)$ random entries)?

A further natural question is whether we can use the sparse polynomial interpolation technique by Ben-Or and Tiwari [7] (see also [49, 24] for alternative descriptions of their approach) to give a more efficient deterministic algorithm for output-sensitive matrix multiplication. Indeed, they show how to use $\mathcal{O}(t)$ evaluations of a t-sparse polynomial p to efficiently interpolate p (for $p = g^{A,B}$, this corresponds to determining AB). Specifically, the $\mathcal{O}(t)$ evaluations define a certain Toeplitz system whose solution yields the coefficients of a polynomial $\zeta(Z) = \prod_{i=1}^z (Z - r_i)$ where r_i is the value of the i-th monomial of p evaluated at a certain known value. By factoring ζ into its linear factors, we can determine the monomials of p (i.e., for $p = g^{A,B}$, the nonzero entries of AB). In our case, we can then obtain AB by naive computations of the inner products at the nonzero positions in time $\mathcal{O}(nt)$. The bottleneck in this approach appears to be deterministic polynomial factorization into linear factors: In our setting, we would need to factor a degree-($\leq t$) polynomial over a prime field $\mathbb{F}_p$ of size $p = \Theta(n^2)$. We are not aware of deterministic algorithms faster than Shoup's $\mathcal{O}(t^{2+\varepsilon} \cdot \sqrt{p} \log^2 p)$-time algorithm [37], which would yield an $\mathcal{O}(n^2 + nt^{2+\varepsilon})$-time algorithm at best. However, such an algorithm would be dominated by Kutzkov's algorithm [28]. Can we sidestep this bottleneck? Note that some works improve on Shoup's running time for suitable primes (assuming the Extended Riemann Hypothesis; see [44, Chapter 14] for references).

References

1 Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'15)*, pages 1681–1697, 2015. doi:10.1137/1.9781611973730.112.

2 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proc. 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS'14)*, pages 434–443, 2014. doi:10.1109/FOCS.2014.53.

3 Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'15)*, pages 218–230, 2015. doi:10.1137/1.9781611973730.17.

4 Noga Alon, Zvi Galil, and Oded Margalit. On the exponent of the all pairs shortest path problem. *Journal of Computer and System Sciences*, 54(2):255–262, 1997. doi:10.1006/jcss.1997.1388.

5 Rasmus Resen Amossen and Rasmus Pagh. Faster join-projects and sparse matrix multiplications. In *Proc. 12th International Conference on Database Theory (ICDT'09)*, pages 121–126, 2009. doi:10.1145/1514894.1514909.

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

7 Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynominal interpolation (extended abstract). In *Proc. 20th Annual ACM Symposium on Theory of Computing (STOC'88)*, pages 301–309, 1988. doi:10.1145/62212.62241.

8 Markus Bläser. Fast matrix multiplication. *Theory of Computing, Graduate Surveys*, 5:1–60, 2013. doi:10.4086/toc.gs.2013.005.

9 Karl Bringmann, Fabrizio Grandoni, Barna Saha, and Virginia Vassilevska Williams. Truly sub-cubic algorithms for language edit distance and RNA-folding via fast bounded-difference min-plus product. In *Proc. 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS'16)*, pages 375–384, 2016. doi:10.1109/FOCS.2016.48.

10 Harry Buhrman and Robert Spalek. Quantum verification of matrix products. In *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'06)*, pages 880–889, 2006. URL: http://dl.acm.org/citation.cfm?id=1109557.1109654.

11 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the Strong Exponential Time Hypothesis and consequences for non-reducibility. In *Proc. 2016 ACM Conference on Innovations in Theoretical Computer Science (ITCS'16)*, pages 261–270, 2016. doi:10.1145/2840728.2840746.

12 Timothy M. Chan. More logarithmic-factor speedups for 3SUM, (median, +)-convolution, and some geometric 3SUM-hard problems. In *Proc. 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'18)*, pages 881–897, 2018. doi:10.1137/1.9781611975031.57.

13 Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal on Symbolic Computation*, 9(3):251–280, 1990. `doi:10.1016/S0747-7171(08)80013-2`.

14 Charles M. Fiduccia. Polynomial evaluation via the division algorithm: The fast Fourier transform revisited. In *Proc. 4th Annual ACM Symposium on Theory of Computing (STOC'72)*, pages 88–93, 1972. `doi:10.1145/800152.804900`.

15 Rusins Freivalds. Fast probabilistic algorithms. In *Proc. 8th International Symposium on Mathematical Foundations of Computer Science (MFCS'79)*, pages 57–69, 1979. `doi:10.1007/3-540-09526-8_5`.

16 Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry*, 5:165–185, 1995. `doi:10.1016/0925-7721(95)00022-2`.

17 Leszek Gasieniec, Christos Levcopoulos, Andrzej Lingas, Rasmus Pagh, and Takeshi Tokuyama. Efficiently correcting matrix products. *Algorithmica*, 79(2):428–443, 2017. `doi:10.1007/s00453-016-0202-3`.

18 Allan Grønlund and Seth Pettie. Threesomes, degenerates, and love triangles. In *Proc. 55th IEEE Annual Symposium on Foundations of Computer Science (FOCS'14)*, pages 621–630, 2014. `doi:10.1109/FOCS.2014.72`.

19 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. `doi:10.1006/jcss.2000.1727`.

20 Mark A. Iwen and Craig V. Spencer. A note on compressed sensing and the complexity of matrix multiplication. *Information Processing Letters*, 109(10):468–471, 2009. `doi:10.1016/j.ipl.2009.01.010`.

21 Riko Jacob and Morten Stöckel. Fast output-sensitive matrix multiplication. In *Proc. 23rd Annual European Symposium on Algorithms (ESA'15)*, pages 766–778, 2015. `doi:10.1007/978-3-662-48350-3_64`.

22 Zahra Jafargholi and Emanuele Viola. 3SUM, 3XOR, triangles. *Algorithmica*, 74(1):326–343, 2016. `doi:10.1007/s00453-014-9946-9`.

23 Stacey Jeffery, Robin Kothari, and Frédéric Magniez. Improving quantum query complexity of boolean matrix multiplication using graph collision. In *Proc. 39th International Colloquium on Automata, Languages, and Programming (ICALP'12)*, pages 522–532, 2012. `doi:10.1007/978-3-642-31594-7_44`.

24 Erich Kaltofen and Yagati N. Lakshman. Improved sparse multivariate polynomial interpolation algorithms. In *Proc. 1st International Symposium on Symbolic and Algebraic Computation (ISSAC'88)*, pages 467–474, 1988. `doi:10.1007/3-540-51084-2_44`.

25 Daniel M. Kane, Shachar Lovett, and Shay Moran. Near-optimal linear decision trees for k-SUM and related problems. *CoRR*, abs/1705.01720, 2017. To appear in STOC'18. `arXiv:1705.01720`.

26 Tracy Kimbrel and Rakesh K. Sinha. A probabilistic algorithm for verifying matrix products using $O(n^2)$ time and $\log_2 n + O(1)$ random bits. *Information Processing Letters*, 45(2):107–110, 1993. `doi:10.1016/0020-0190(93)90224-W`.

27 Marvin Künnemann. On nondeterministic derandomization of Freivalds' algorithm: Consequences, avenues and algorithmic progress. *CoRR*, abs/1806.09189, 2018. `arXiv:1806.09189`.

28 Konstantin Kutzkov. Deterministic algorithms for skewed matrix products. In *Proc. 30th International Symposium on Theoretical Aspects of Computer Science (STACS'13)*, pages 466–477, 2013. `doi:10.4230/LIPIcs.STACS.2013.466`.

29 François Le Gall. Powers of tensors and fast matrix multiplication. In *Proc. 39th International Symposium on Symbolic and Algebraic Computation (ISSAC'14)*, pages 296–303, 2014. `doi:10.1145/2608628.2608664`.

30 Andrzej Lingas. A fast output-sensitive algorithm for boolean matrix multiplication. In *Proc. 17th Annual European Symposium on Algorithms (ESA'09)*, pages 408–419, 2009. doi:10.1007/978-3-642-04128-0_37.

31 Ross M. McConnell, Kurt Mehlhorn, Stefan Näher, and Pascal Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119–161, 2011. doi:10.1016/j.cosrev.2010.09.009.

32 Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993. doi:10.1137/0222053.

33 Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 026(2):415–419, 1985. URL: http://eudml.org/doc/17394.

34 Rasmus Pagh. Compressed matrix multiplication. *ACM Transactions on Computation Theory*, 5(3):9:1–9:17, 2013. doi:10.1145/2493252.2493254.

35 Daniel S. Roche. Error correction in fast matrix multiplication and inverse. *CoRR*, abs/1802.02270, 2018. arXiv:1802.02270.

36 Claus-Peter Schnorr and C. R. Subramanian. Almost optimal (on the average) combinatorial algorithms for boolean matrix product witnesses, computing the diameter (extended abstract). In *Proc. 2nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM'98)*, pages 218–231, 1998. doi:10.1007/3-540-49543-6_18.

37 Victor Shoup. On the deterministic complexity of factoring polynomials over finite fields. *Information Processing Letters*, 33(5):261–267, 1990. doi:10.1016/0020-0190(90)90195-4.

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

39 Terence Tao, Ernest Croot III, and Harald Helfgott. Deterministic methods to find primes. *Mathematics of Computation*, 81(278):1233–1246, 2012. doi:10.1090/S0025-5718-2011-02542-1.

40 Leslie G. Valiant. General context-free recognition in less than cubic time. *Journal of Computer and System Sciences*, 10(2):308–315, 1975. doi:10.1016/S0022-0000(75)80046-8.

41 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proc. 44th Annual ACM Symposium on Theory of Computing Conference (STOC'12)*, pages 887–898, 2012. doi:10.1145/2213977.2214056.

42 Virginia Vassilevska Williams. Fine-grained algorithms and complexity. In *Proc. 21st International Conference on Database Theory (ICDT'18)*, pages 1:1–1:1, 2018. doi:10.4230/LIPIcs.ICDT.2018.1.

43 Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *Proc. 51th Annual IEEE Symposium on Foundations of Computer Science (FOCS'10)*, pages 645–654, 2010. doi:10.1109/FOCS.2010.67.

44 Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra (3. ed.)*. Cambridge University Press, 2013.

45 Jirí Wiedermann. Fast nondeterministic matrix multiplication via derandomization of Freivalds' algorithm. In *Proc. 8th IFIP International Conference on Theoretical Computer Science (TCS'14)*, pages 123–135, 2014. doi:10.1007/978-3-662-44602-7_11.

46 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proc. 46th Annual ACM Symposium on Theory of Computing (STOC'14)*, pages 664–673, 2014. doi:10.1145/2591796.2591811.

47 Ryan Williams. Strong ETH breaks with Merlin and Arthur: Short non-interactive proofs of batch evaluation. In *Proc. 31st Conference on Computational Complexity (CCC'16)*, pages 2:1–2:17, 2016. doi:10.4230/LIPIcs.CCC.2016.2.

48 Raphael Yuster and Uri Zwick. Fast sparse matrix multiplication. *ACM Transactions on Algorithms*, 1(1):2–13, 2005. `doi:10.1145/1077464.1077466`.

49 Richard Zippel. Interpolating polynomials from their values. *Journal of Symbolic Computation*, 9(3):375–403, 1990. `doi:10.1016/S0747-7171(08)80018-1`.