Quantum Verification of Matrix Products

Harry Buhrman* Robert Špalek*

Abstract

We present a quantum algorithm that verifies a product of two $n \times n$ matrices over any integral domain with bounded error in worst-case time $O(n^{5/3})$ and expected time $O(n^{5/3}/\min(w, \sqrt{n})^{1/3})$, where w is the number of wrong entries. This improves the previous best algorithm [3] that runs in time $O(n^{7/4})$. We also present a quantum matrix multiplication algorithm that is efficient when the result has few nonzero entries.

1 Introduction

The computational complexity of matrix multiplication is a subject of extensive study. Matrix multiplication is the central algorithmic part of many applications like for example solving linear systems of equations and computing the transitive closure. A fast algorithm for matrix multiplication thus implies a fast algorithm for a variety of computational tasks. Strassen [20] was the first to show that surprisingly two $n \times n$ matrices can be multiplied in time $O(n^\omega)$ for $\omega < 3$. His result was improved by many subsequent papers. The best known bound to date is an algorithm with $\omega \approx 2.376$ by Coppersmith and Winograd [9]. It is a major open problem to determine the true value of ω. Freivalds showed [10] that *verifying* whether the product of two $n \times n$ matrices is equal to a third can be done with high probability in time proportional to n^2. We refer to this latter problem as *matrix verification*.

We study the computational complexity of matrix multiplication and verification on a quantum computer. The first to study matrix verification in the quantum mechanical setting were Ambainis, Buhrman, Høyer, Karpinski, and Kurur [3] who used a clever recursive version of Grover's algorithm to verify whether two $n \times n$ matrices equal a third in time $O(n^{7/4})$, thereby improving the optimal classical bound of Freivalds.

In this paper we construct a bounded-error quantum algorithm for the matrix verification problem that runs in time $O(n^{5/3})$. Suppose we are verifying whether $A \times B = C$. When the number of "wrong" entries in C is w, our algorithm runs in expected time $O(n^{5/3}/\min(w, \sqrt{n})^{1/3})$. For $w = \sqrt{n}$ we have a matching lower bound.

Our algorithm uses the quantum random walk formalism by Szegedy [21] that he developed as a generalization of the quantum random walk technique of Ambainis [2]. Ambainis used a quantum random walk to obtain an optimal quantum algorithm for the element distinctness problem. If one were to adapt that method directly to the setting of matrix verification, one does obtain an $O(n^{5/3})$ algorithm in terms of queries to the input. However that algorithm still requires $\Omega(n^2)$ time, because it computes several times a matrix product of sub-matrices that are loaded into the memory. This costs no additional queries, but it takes an additional time. The rest of the paper is devoted to improve the time complexity of the quantum algorithm to $O(n^{5/3})$.

We perform a quantum random walk on the product of two Johnson graphs, analyze its spectral gap, that is the second smallest eigenvalue of its Laplacian, and estimate that in our setting enough of the nodes are marked if $A \times B \neq C$. See § 3 for a detailed description of our algorithm. We next introduce a combinatorial tool to analyze the behavior of our algorithm when many of the entries are wrong. Finally we use our fast quantum matrix verification algorithm as a building block to construct a quantum algorithm for computing the actual matrix product $A \times B$ that is substantially faster than any known classical method, when there are not too many nonzero entries in the final product.

2 Preliminaries

2.1 Quantum query complexity We assume familiarity with quantum computing [18] and sketch the model of quantum query complexity. Suppose we want to compute some function f. For input $x \in \{0,1\}^N$, a *query* gives us access to the input bits. It corresponds to the unitary transformation

$$O : |i, b, z\rangle \mapsto |i, b \oplus x_i, z\rangle.$$

Here $i \in \{1, \ldots, N\}$ and $b \in \{0, 1\}$; the z-part corresponds to the workspace, which is not affected by the query. We assume the input can be accessed only via such queries. A t-query quantum algorithm has the form $A = U_t O U_{t-1} \cdots O U_1 O U_0$, where the U_k are fixed uni-

*CWI and University of Amsterdam. Supported in part by the EU fifth framework project QAIP, IST-1999-11234, and RESQ, IST-2001-37559, and an NWO grant. E-mail: {buhrman, sr}@cwi.nl.

tary transformations independent of x. This A depends on x via the t applications of O. The algorithm starts in the initial state $|0^k\rangle$ and its *output* is the result of measuring a dedicated part of the final state $A|0^k\rangle$, where k is the total amount of space used by the algorithm.

2.2 Quantum search One of the most interesting quantum algorithms is Grover's search algorithm [11, 5]. It finds an index of an input bit x_i in an n-bit input such that $x_i = 1$ in expected number of $O(\sqrt{n/(|x|+1)})$ queries, where $|x|$ is the Hamming weight (number of ones) in the input. Grover's algorithm can be cast in more general terms as amplitude amplification: given a quantum algorithm A that accepts with probability p, then it can be amplified to have constant success probability with $\sqrt{1/p}$ iterations of A.

Given n numbers $x_1, \ldots, x_n$ as input, the element distinctness problem is the task to determine whether there are two distinct indices i and j such that $x_i = x_j$. Ambainis in a very nice paper [2] applied quantum random walks in a novel way and constructed a quantum algorithm that solves element distinctness in $O(n^{2/3})$ queries. This algorithm is faster than the algorithm [7] which is based on amplitude amplification and uses $O(n^{3/4})$ queries. Ambainis's method was generalized by Szegedy [21] for all graphs and even for all symmetric Markov chains (with non-uniform transition probabilities). Szegedy's method can be regarded as a quantum walk version of amplitude amplification [6].

2.3 Previous best algorithm for matrix verification Ambainis et al. [3] discovered a quantum algorithm running in time $O(n^{7/4})$. Since it was never published, we briefly sketch it here. Let A, B, C be $n \times n$ matrices. First, partition the matrices B and C into $\sqrt{n}$ blocks of $\sqrt{n}$ columns each. It holds that $AB = C$ iff $AB_i = C_i$ for every i, where B_i and C_i are the submatrices of size $n \times \sqrt{n}$. The verification of $AB_i = C_i$ can be done with bounded error in time $O(n^{3/2})$ as follows: choose a random vector x of length $\sqrt{n}$, multiply both sides of the equation by x from the right side, compute classically $y = B_i x$ and $z = C_i x$, and verify the matrix-vector product $Ay = z$ by a Grover search. The search over n rows takes $O(\sqrt{n})$ iterations and a verification of one row takes time n. Now, we apply amplitude amplification on the top of this sub-routine V_i, and compute the And of all $\sqrt{n}$ blocks using $n^{1/4}$ calls to V_i.

2.4 Notation Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. Let $A_{n\times m}$ denote a matrix A of dimension $n \times m$. Let A^T denote the transpose of A. For a $R \subseteq [n]$, let $A|_R$ denote the $|R| \times m$ sub-matrix of A restricted to the rows from R. Analogously, for every $S \subseteq [m]$, let $A|^S$ denote the $n \times |S|$ sub-matrix of A restricted to the columns from S. Let $\lambda(M)$ denote the *spectral norm* of a matrix M; it is equal to the largest eigenvalue of M for symmetric M. For a set S, let $\binom{S}{k}$ denote all subsets of S of size k. An *integral domain* is a commutative ring with identity and no divisors of 0.

For a graph G, let V_G denote the vertices of G and let E_G denote the edges of G. The normalized *Laplacian matrix* $\mathcal{L}(G)$ of an undirected graph G is a symmetric $|V_G| \times |V_G|$ matrix defined by $\mathcal{L}_{i,j}(G) = 1$ if $i = j$, it is $-1/\sqrt{d_i d_j}$ if $i \neq j$, and 0 otherwise. A *spectral gap* of a graph G, often called the *Fiedler value* of G, equals to the second smallest eigenvalue of $\mathcal{L}(G)$; it is nonzero if G is connected. The *Johnson graph* $J(n, k)$ is defined as follows: its vertices are subsets of $[n]$ of size k, and two vertices are connected iff they differ in exactly one number. Let $G = G_1 \times G_2$ denote the *strong product graph* of two graphs G_1, G_2, defined as follows: $V_G = V_{G_1} \times V_{G_2}$, and $((g_1, g_2), (g_1', g_2')) \in E_G$ iff $(g_1, g_1') \in E_{G_1}$ and $(g_2, g_2') \in E_{G_2}$.

3 Algorithm for matrix verification

Let A, B, C be $n \times n$ matrices over some integral domain. A *matrix verification* is deciding whether $AB = C$. We construct an efficient quantum walk algorithm for this problem. It is described in Figure 1 on the next page and its expected running time is analyzed in § 4 and § 5.

The intuition behind the algorithm is the following: The sub-routine *Verify Once* performs a quantum walk on subsets R of rows of A and subsets S of columns of B, perturbed by phase flips whenever $C|^S_R$ contains at least one wrong entry. It then estimates the scalar product of the superposition computed by the quantum walk and the uniform superposition. If $AB = C$, then the phase flip is never performed and the diffusion on a uniform superposition is equal to identity. Hence the superposition computed by the quantum walk stays uniform, and the scalar product is 1. On the other hand, if $AB \neq C$ and k is sufficiently large, then for the number of quantum walk steps ℓ drawn uniformly from $\{1, 2, \ldots, k\}$, with high probability, the quantum walk converges to a superposition almost orthogonal to the uniform superposition. Since the optimal value of k is not known beforehand, the main routine *Matrix Verification* tries a sequence of exponentially increasing k. We also use the following trick to improve the performance of the algorithm. Instead of checking all of $A|_R$ and $B|^S$, we multiply the matrices in *Verify Once* from both sides by random vectors $\mathbf{p}, \mathbf{q}$. It allows us to achieve both a better running time and smaller space complexity.

Matrix Verification (input size n, matrices A, B, C) returns 1 when $AB \neq C$:

1. Take any $1 < \lambda < \frac{8}{7}$, for example $\lambda = \frac{15}{14}$.
2. For $i = 0, 1, \ldots, \log_\lambda(n^{2/3}) + 9$, repeat 16 times the following:
   - Run *Verify Once* ($\sqrt[3]{8} \cdot \lambda^i$).
   - If it returns 1, then return "not equal".
3. Return "equal".

Verify Once (number of rows k) returns 1 when $AB \neq C$ is detected:

4. Pick the number of iterations ℓ uniformly at random from $\{1, 2, \ldots, k\}$. Pick a random row vector $\mathbf{p}$ and a random column vector $\mathbf{q}$ of length n.
5. **Initialization.** Put the quantum register into superposition $\sum_{\substack{R\subseteq[n]\\|R|=k}} \sum_{\substack{S\subseteq[n]\\|S|=k}} |R\rangle|S\rangle$. Think of R as a subset of the rows of A and S as a subset of the columns of B. Compute $\mathbf{a}_R = \mathbf{p}|^R \cdot A|_R$, $\mathbf{b}_S = B|^S \cdot \mathbf{q}|_S$, and $c_{R,S} = \mathbf{p}|^R \cdot C|_R^S \cdot \mathbf{q}|_S$ in time $2kn + k^2$. Let $|z\rangle = |+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$. The quantum state is
$$|+\rangle \sum_{R,S} |R, \mathbf{a}_R\rangle\, |S, \mathbf{b}_S\rangle\, |c_{R,S}\rangle.$$
6. **Quantum walk.** Conditioned on $|z\rangle$, perform ℓ iterations of the following:
   (a) **Phase flip.** Multiply the quantum phase by -1 iff $\mathbf{a}_R \cdot \mathbf{b}_S \neq c_{R,S}$. The scalar product is verified in time n using no queries.
   (b) **Diffusion.** Perform one step of quantum walk on (R, S), that is exchange one row and one column. The update of $\mathbf{a}_R$, $\mathbf{b}_S$, and $c_{R,S}$ costs $2n$ queries to A, $2n$ queries to B, and $4k$ queries to C.
7. Apply the Hadamard transform on $|z\rangle$, measure it, and return the outcome.

Figure 1: Quantum algorithm for matrix verification

4 Analysis of the algorithm

In this section, we prove that *Matrix Verification* has one-sided bounded error and estimate its expected running time depending on the set of wrong entries. We use a recent result by Szegedy [21], which can be regarded as a quantum walk version of quantum amplitude amplification [6].

THEOREM 4.1. (SZEGEDY [21]) *Let G be an undirected graph on vertex set X, and let δ_G be the spectral gap of G. Some set of vertices $M \subseteq X$ are marked with the promise that $|M|$ is either zero or at least $\varepsilon|X|$. Let $k_0 = \mathrm{O}(1/\sqrt{\delta_G \varepsilon})$. For every $k \geq k_0$, the following quantum algorithm decides whether M is non-empty with one-sided error $\frac{1}{2} \leq \gamma \leq \frac{7}{8}$ in time $\mathrm{O}(T_X + k \cdot (T_M + T_G))$:*

1. *Initialization. Compute a uniform superposition over X; assume this takes time T_X.*
2. *Pick $\ell \in \{1, 2, \ldots, k\}$ uniformly at random. Repeat ℓ times the following:*
   - *Phase flip: Flip the phase if an element is marked; assume this takes time T_M.*
   - *Diffusion: Perform one step of quantum walk on G; assume this takes time T_G.*
3. *Return "non-empty" if the distribution computed by the quantum walk has small scalar product with the uniform distribution over X.*

4.1 Multiplication by random vectors Let *Verify Full* denote a modified version of *Verify Once* that does not use the random vectors $\mathbf{p}, \mathbf{q}$, but instead reads all the sub-matrices into memory and verifies $A|_R \cdot B|^S = C|_R^S$ in the phase-flip step (6a). *Verify Full* has the same query complexity as *Verify Once*, but its space complexity and running time are bigger. Although the phase-flip costs no additional queries, the time needed to compute classically $A|_R \cdot B|^S$ is at least kn, whereas the scalar product $\mathbf{a}_R \cdot \mathbf{b}_S$ is computed in time n.[1]

The multiplication by $\mathbf{p}, \mathbf{q}$ complicates the analysis. For example, if there is exactly one wrong entry and the multiplication is over $\mathbb{GF}(2)$, then the wrong entry is completely hidden by a multiplication by zero with probability $\frac{3}{4}$. However, we prove that 16 independent calls to *Verify Once* have a better success probability than one call to *Verify Full*.

[1] It seems that this slowdown "time $\gg$ #queries" is a typical property of algorithms using quantum walks: the quantum algorithm for element distinctness [2] needs to use random hash functions to remedy it, and it is open whether triangle finding [17] can be improved this way.

Let $R, S \subseteq [n]$ denote subsets of rows of A and columns of B, and let

$$W = \{(i,j) \,|(AB - C)_{i,j} \neq 0\} \tag{4.1}$$

be the *set of wrong entries* of the matrix product. We *mark* (R,S) iff $C|_R^S$ contains a wrong entry, formally $A|_R \cdot B|^S \neq C|_R^S$, or equivalently $W \cap R \times S \neq \emptyset$. Let

$$\varepsilon(W,k) = \Pr_{R,S}[(R,S) \text{ is marked}], \tag{4.2}$$

be the *fraction of marked pairs* for $|R| = |S| = k$. In § 5, we prove the following combinatorial lower bound on $\varepsilon(W,k)$:

LEMMA 4.1. *Let*

$$q(W) = \max(|W'|, \min(|W|, \sqrt{n})), \tag{4.3}$$

where W' is the largest independent subset of W, that is it contains at most one 1 in every row and column. For every W and $k \leq n^{2/3}/q(W)^{1/3}$, it holds that $\varepsilon(W,k) = \Omega(\frac{k^2}{n^2} q(W))$.

We call $(R,S,\mathbf{p},\mathbf{q})$ *revealing* iff $\mathbf{p}|^R \cdot (A|_R \cdot B|^S) \cdot \mathbf{q}|_S \neq \mathbf{p}|^R \cdot C|_R^S \cdot \mathbf{q}|_S$, because it reveals the existence of a wrong entry in the matrix product. The condition is equivalent to $(\mathbf{p}|^R \cdot A|_R) \cdot (B|^S \cdot \mathbf{q}|_S) = \mathbf{a}_R \cdot \mathbf{b}_S \neq c_{R,S}$ due to the associativity of matrix multiplication. The performance of the algorithm depends on the *fraction of revealing pairs*

$$\zeta_{\mathbf{p},\mathbf{q}}(W,k) = \Pr_{R,S}[(R,S,\mathbf{p},\mathbf{q}) \text{ is revealing}], \tag{4.4}$$

where $|R| = |S| = k$. By the above example, not every marked pair is revealing. However, the fraction of revealing pairs is expected to be not too far from the fraction of marked pairs. We prove the following in § A:

LEMMA 4.2. *Let $\mathbf{p},\mathbf{q}$ be picked uniformly at random. Then $\Pr[\zeta_{\mathbf{p},\mathbf{q}}(W,k) \geq \frac{1}{8}\varepsilon(W,k)] > \frac{1}{8}$.*

The constant probability $\frac{1}{8}$ of picking "good" random vectors is compensated by a constant number of repetitions. Using the following lemma, it is sufficient to analyze the error of the algorithm as if it is performing *Verify Full* in each step.

LEMMA 4.3. *Let $AB \neq C$. The probability that* Verify Once $(\sqrt[3]{8} \cdot k)$ *outputs 1 at least once in 16 independent trials, each time with new random $\mathbf{p},\mathbf{q}$, is bigger than the success probability of one call to* Verify Full (k).

Proof. By LEMMA 4.2, the success probability of *Verify Once* is at least $\frac{p}{8}$, where p is the success probability of *Verify Once* given that it guesses "good" vectors $\mathbf{p},\mathbf{q}$ with $\zeta(W,k) \geq \frac{1}{8}\varepsilon(W,k)$. By THEOREM 4.1, $p = 1 - \gamma$ satisfies $\frac{1}{2} \geq p \geq \frac{1}{8}$ for every sufficiently large $k \geq k_0$. The factor $\frac{1}{8}$ between $\varepsilon(W,k)$ and $\zeta(W,k)$ is compensated by taking a $\sqrt[3]{8}$-times bigger k_0; the exact dependence is given by equation (4.5) in the proof of THEOREM 4.2. The success probability of 16 independent trials is at least $1-(1-\frac{p}{8})^{16} \geq 1-(e^{-p})^2 \geq 1-(1-0.64p)^2 \geq 1.28p-0.4p^2 \geq p$, because $1-x \leq e^{-x}$, $e^{-x} \leq 1 - 0.64x$ for $x \in [0,1]$, and $p < 0.7$.

4.2 Analysis of *Matrix Verification* In this section, we analyze the expected running time of the main algorithm. We need the following two statements:

LEMMA 4.4. *Let $1 \leq k \leq \frac{n}{2}$. The spectral gap of $G = J(n,k) \times J(n,k)$ is $\delta_G = \Theta(1/k)$.*

Proof. The spectral gap of the Johnson graph $J(n,k)$ is $\frac{n}{(n-k)k}$ [15], which is $\Theta(1/k)$ for $1 \leq k \leq \frac{n}{2}$. It is simple to prove that $\delta_{G_1 \times G_2} = \min(\delta_{G_1}, \delta_{G_2})$. We conclude that $\delta_G = \Theta(1/k)$.

LEMMA 4.5. *Let $|\varphi\rangle$ and $|\psi\rangle$ be quantum states, let $|X\rangle = \frac{\sqrt{2}}{2}(|0,\varphi\rangle + |1,\psi\rangle)$, and let $|Y\rangle = (H \otimes I)|X\rangle$. If the first qubit of $|Y\rangle$ is measured in the computational basis, then $\Pr[Y=1] = \frac{1}{2}(1 - \langle\varphi|\psi\rangle)$.*

Proof. $|Y\rangle = \frac{1}{2}(|0,\varphi\rangle + |1,\varphi\rangle) + \frac{1}{2}(|0,\psi\rangle - |1,\psi\rangle) = |0\rangle\frac{|\varphi\rangle+|\psi\rangle}{2} + |1\rangle\frac{|\varphi\rangle-|\psi\rangle}{2}$, hence $\Pr[Y=1] = \|\frac{|\varphi\rangle-|\psi\rangle}{2}\|^2 = \frac{1}{4}(\langle\varphi| - \langle\psi|)(|\varphi\rangle - |\psi\rangle) = \frac{1}{4}(\langle\varphi|\varphi\rangle + \langle\psi|\psi\rangle - 2\langle\varphi|\psi\rangle) = \frac{1}{2}(1 - \langle\varphi|\psi\rangle)$.

THEOREM 4.2. Matrix Verification *always returns "equal" if $AB = C$. If $AB \neq C$, then it returns "not equal" with probability at least $\frac{2}{3}$. Its worst-case running time is $O(n^{5/3})$, its expected running time is $O(n^{5/3}/q(W)^{1/3})$, and its space complexity is $O(n)$.*

Proof. We compute an upper bound on the expected number of iterations of *Verify Full*. By LEMMA 4.3, the number of calls to *Verify Once* achieving the same success probability is less than 16 with k at most $\sqrt[3]{8}$ times bigger. *Verify Once* walks ℓ quantum steps on the strong product graph of two Johnson graphs $G = J(n,k) \times J(n,k)$. The marked vertices of G correspond to marked pairs (R,S), that is the pairs such that $A|_R \cdot B|^S \neq C|_R^S$. The initialization costs time $T_X = O(kn)$, a phase flip costs time $T_M = n$, and one step of the quantum walk costs time $T_G = 4n + 4k = O(n)$. The running time of *Verify Once* is thus $O((k+\ell)n) = O(kn)$. The scalar product of the two quantum distributions is estimated using LEMMA 4.5, where the first qubit of $|Y\rangle$ corresponds to the conditioning qubit $|z\rangle$.

Let $W \neq \emptyset$. By THEOREM 4.1, *Verify Once* recognizes a wrong matrix product with bounded error for every $k \geq \mathrm{O}(1/\sqrt{\delta_G\,\varepsilon(W,k)})$. Note that the same k is used twice: once as an upper bound on the number of iterations ℓ, and once as the number of rows/columns read into memory, which determines $\varepsilon(W,k)$. Plug in $\varepsilon(W,k) = \Omega(\frac{k^2}{n^2}q(W))$ by LEMMA 4.1 and $\delta_G = \Theta\left(\frac{1}{k}\right)$ by LEMMA 4.4. We get that $k \geq \mathrm{O}(n/\sqrt{kq(W)})$, which gives the condition

$$k \geq k_0 = \mathrm{O}(n^{2/3}/q(W)^{1/3}). \tag{4.5}$$

Hence for every $k \geq k_0$, *Verify Once* makes only small one-sided error. The algorithm *Matrix Verification* does not know $q(W)$ and k_0, but it instead tries different values of k from the exponentially increasing sequence $1 \ldots \lambda^i$.

Let $T_i = \mathrm{O}(\lambda^i n)$ denote the running time of the i-th call to *Verify Once*, and let L be the number of the last call. The expected running time can be written as a telescoping sum

$$\mathrm{E}[T] = \sum_{j=0}^{\infty}\left(\sum_{i=0}^{j} T_i\right)\cdot\Pr[L=j] = \sum_{i=0}^{\infty} T_i\cdot\Pr[L\geq i].$$

Each call after $k \geq k_0$ fails with probability $\gamma \leq \frac{7}{8}$, so

$$\begin{aligned}\mathrm{E}[T] &= \sum_i \lambda^i n\cdot\Pr[L\geq i]\\ &\leq \sum_{i=0}^{(\log_\lambda k_0)-1}\lambda^i n\cdot 1 + \sum_{i=\log_\lambda k_0}^{\infty}\lambda^i n\cdot\gamma^{i-\log_\lambda k_0}\\ &\leq \mathrm{O}(k_0 n)\left(1+\sum_{i=0}^{\infty}(\lambda\gamma)^i\right) = \mathrm{O}\left(\frac{n^{5/3}}{q(W)^{1/3}}\right),\end{aligned}$$

because $\lambda\gamma < \frac{8}{7}\cdot\frac{7}{8} = 1$. The probability that a wrong product is never recognized is $\leq \gamma^9 < \frac{1}{3}$, where 9 is the number of calls after $k \geq n^{2/3}$.

Matrix Verification never makes an error when $AB = C$. In this case, the phase flip is equal to the identity operation. The diffusion is also equal to identity on the uniform distribution, hence the whole quantum walk in *Verify Once* does nothing and the qubit $|z\rangle = |+\rangle$ stays untouched. Finally, *Matrix Verification* always terminates when $k \geq \lambda^9 n^{2/3}$, hence its worst-case running time is $\mathrm{O}(n^{5/3})$.

4.3 Comparison with other quantum walk search algorithms *Matrix Verification* resembles a few other algorithms. The first quantum algorithm of this type was the quantum walk algorithm for element distinctness [2]. The same technique was subsequently successfully applied to triangle finding [17] and group commutativity testing [16]. Both algorithms walk on the Johnson graph $J(n,k)$. The analysis of Ambainis [2] relies on the fact that the quantum state stays in a constant-dimensional subspace. This constraint is satisfied if there is at most one solution; then the subsets can be divided into a constant number of cases. In the non-promise version, the number of cases is, however, not constant. The latter papers [2, 17] solve the non-promise case by projecting the input into a random subspace. With high probability, there is exactly one solution in the subspace; this trick originally comes from Valiant and Vazirani [22]. Since it is not known whether this technique can be used in more than one dimension, we solve the non-promise version of matrix verification using the more general quantum walk by Szegedy [21] instead of the original one by Ambainis [2].

THEOREM 4.1 is quite general because it allows walking on an arbitrary undirected graph. On the other hand, the algorithm *Verify Once* obtained by it is a bit slower than the original Ambainis walk [2, 17]. First, *Verify Once* only solves the decision version of the problem and it does not find the actual position of a wrong entry. We resolve it by a binary search. Second, *Verify Once* does a phase flip after every step of quantum walk instead of doing it once per block of steps. However, for both element distinctness and matrix verification, the additional cost is subsumed by the cost of the quantum walk.

5 Lower bounds on the fraction of marked pairs

In this section, we try to solve the following combinatorial problem:

Problem. Given an $n \times n$ Boolean matrix W and two integers $1 \leq r, s \leq n$, what is the probability $\varepsilon(W,r,s)$ that a random $r \times s$ sub-matrix of W contains a 1? Or, equivalently: Given a bipartite graph on n,n vertices, what is the probability that a randomly chosen induced subgraph with r,s vertices contains at least one edge?

It is simple to prove that $\varepsilon(W,r,s)$ is monotone in all its three parameters. As we have seen in THEOREM 4.2, the expected running time of *Matrix Verification* depends on the fraction of marked pairs, which is $\varepsilon(W,k,k)$, also denoted there by $\varepsilon(W,k)$.[2] Let us compute ε when W contains exactly one 1:

$$\varepsilon(W,r,s) = \binom{n-1}{r-1}\binom{n-1}{s-1}\Big/\binom{n}{r}\binom{n}{s} = \frac{rs}{n^2}.$$

With this bound, monotonicity, and THEOREM 4.2, we conclude that *Matrix Verification* finds the correct

[2] Our algorithm only tries balanced choices $r = s = k$. Since the initialization costs $\mathrm{O}((r+s)n)$, setting one of the variables smaller does not decrease the query complexity, but it decreases the success probability of *Verify Once*.

answer with bounded error in time $O(n^{5/3})$. The rest of this section contains a more detailed analysis of the expected running time of the algorithm for larger W. Unfortunately, we are only able to prove weak lower bounds on ε for general W. However, if one improves them, then we automatically get an improved upper bound on the running time of *the same* algorithm.

Henceforth, let r, s be sufficiently small. The average probability over all sets W with t ones is $E_{W:|W|=t}[\varepsilon(W,r,s)] = \Omega(|W|\frac{rs}{n^2})$ (LEMMA 5.1). We are able to prove the same bound for all $|W| \leq \sqrt{n}$ (LEMMA 5.2), when W is an independent set, that is it does not contain two ones in the same row or column (LEMMA 5.3), or when the ones in W form a rectangle (again LEMMA 5.2). However, the latter rectangle bound only holds for a limited class of r, s, which does not include the balanced case $r = s = k$ in the range used by our algorithm. As a consequence, if the ones in W form a full row or a column, our algorithm is slower than what would follow from this formula. We, however, show that in this case our algorithm is optimal (THEOREM 5.1); this is the only known tight lower bound for our algorithm. Most of the proofs are postponed to § B.

LEMMA 5.1. *Let $rs \leq \frac{n^2}{t}$. Then $E_{W:|W|=t}[\varepsilon(W,r,s)] = \Omega(|W|\frac{rs}{n^2})$.*

LEMMA 5.2. *Let w be the number of nonzero rows of W, and let w' be maximal the number of nonzero entries in a row. Then for every $r \leq \frac{n}{w}$ and $s \leq \frac{n}{w'}$, $\varepsilon(W,r,s) = \Omega(|W|\frac{rs}{n^2})$.*

LEMMA 5.3. *Let W have at most one entry in every row and column. Then for every r, s satisfying $rs \leq n^{4/3}/|W|^{2/3}$, $\varepsilon(W,r,s) = \Omega(|W|\frac{rs}{n^2})$.*

The main LEMMA 4.1 is a direct corollary of LEMMA 5.2 and LEMMA 5.3.

Proof. of LEMMA 4.1. LEMMA 5.2 implies that $\varepsilon(W,k) = \Omega(\frac{k^2}{n^2}\min(|W|,\sqrt{n}))$: First, assume that $|W| \leq \sqrt{n}$ and verify the restrictions on $r = s = k$. For every $t \leq \sqrt{n}$ it holds that $n^{2/3}/t^{1/3} \leq n/t$. Hence if $|W| \leq \sqrt{n}$, then for every $k \leq n^{2/3}/|W|^{1/3}$ it holds that $k \leq n/|W|$ and, since $w, w' \leq |W|$, also $k \leq \frac{n}{w}$ and $k \leq \frac{n}{w'}$. Hence the lower bound $\varepsilon(W,k) = \Omega(\frac{k^2}{n^2}|W|)$ given by LEMMA 5.2 holds for every k in the range required by LEMMA 4.1. Now, if $|W| > \sqrt{n}$, the bound follows from the monotonicity of $\varepsilon(W,k)$ in W.

LEMMA 5.3 says that $\varepsilon(W',k) = \Omega(\frac{k^2}{n^2}|W'|)$ for every independent W' and k in the range required by LEMMA 4.1. The bound on W follows from the monotonicity of $\varepsilon(W,k)$ in W. If we put these two bounds together, we obtain that $\varepsilon(W,k) = \Omega(\frac{k^2}{n^2}q(W))$, as desired.

The bound cannot be strengthened to $\varepsilon(W,k) = \Omega(\frac{k^2}{n^2}|W|)$ for a general W and the full range of k. We show that *no* quantum algorithm can be fast if the n ones in W form a full row. A straightforward calculation shows that $q(W)$ for this W can be at most $\sqrt{n}$ if we want the bound on ε to hold for all $k \leq O(n^{2/3}/q(W)^{1/3})$.

THEOREM 5.1. *Any bounded-error quantum algorithm distinguishing a correct matrix product and a matrix product with one wrong row has query complexity $\Omega(n^{3/2})$.*

Proof. We reduce Or of n parities of length $n+1$ to matrix verification. Let

$$z = (x_{1,1} \oplus \cdots \oplus x_{1,n} \oplus y_1) \vee \cdots \vee (x_{n,1} \oplus \cdots \oplus x_{n,n} \oplus y_n).$$

Using the quantum adversary lower bound method [1], it follows that computing z requires $\Omega(n^{3/2})$ quantum queries, and the lower bounds holds even if we promise that at most one parity is equal to 1. Since $z = 1$ iff $\exists i : y_i \neq \bigoplus_{\ell=1}^{n} x_{i,\ell}$, one can reduce this problem to the matrix verification $AB = C$ over $\mathbb{GF}(2)$, where $A_{i,j} = x_{i,j}$, $B_{i,j} = 1$, and $C_{i,j} = y_i$. The promise is transformed into that at most one row is wrong.

6 Concluding remarks

6.1 Algorithm for matrix multiplication Let $m \geq n^{2/3}$. One can modify the algorithm to verify the product $A_{n\times m}B_{m\times n} = C_{n\times n}$ in time proportional to $n^{2/3}m$. The quantum walk stays the same and only the inner scalar products are of length m instead of n.

Using the rectangular matrix verification algorithm and binary search, we construct a quantum algorithm that outputs the position of a wrong entry. By iterating this and correcting the wrong entries, we compute the matrix product $AB = C$ whenever a good approximation to C is known. One can always start by guessing $C = 0$, hence the following bound holds:

THEOREM 6.1. *Let $m \geq n^{2/3}$. The matrix product $A_{n\times m}B_{m\times n} = C_{n\times n}$ can be computed with polynomially small error probability in expected time*

$$T_M \leq O(1) \cdot \begin{cases} m\log n \cdot n^{2/3}w^{2/3}; & 1 \leq w \leq \sqrt{n}, \\ m\log n \cdot \sqrt{n}w; & \sqrt{n} \leq w \leq n, \\ m\log n \cdot n\sqrt{w}; & n \leq w \leq n^2, \end{cases} \tag{6.6}$$

where $w = |W|$ is the number of nonzero entries of C.

The algorithm and its analysis are presented in § C. Let us compare our algorithm to the fastest known classical algorithms. Let $w = |W|$ be the number of nonzero entries of C and let $\gamma = \sqrt{n}/(\log n)^{3/2}$. Our algorithm computes sparse matrix products with $w = o(\gamma)$ in subquadratic time $o(nm)$. Classically, *dense* square matrix products can be computed in time $O(n^{2.376})$ [9], which is faster than our algorithm for $w = \Omega(n^{0.876})$, and dense rectangular matrix products can be computed in time $O(n^{1.844+o(1)}m^{0.533} + n^{2+o(1)})$ [8]. Very recently, Indyk independently discovered a classical algorithm for sparse rectangular matrix products running in time $\tilde{O}(m(n + w))$ [13], which is faster than ours for $w = \Omega(\gamma)$. The fastest known algorithm for sparse square *input* matrices runs in time $O(n^{1.2}z^{0.7} + n^{2+o(1)})$ [23], where A, B have each at most z nonzero elements.

6.2 Boolean matrices The algorithm *Verify Once* relies on the fact that arithmetical operations are over some integral domain. If the matrices are over the Boolean algebra $\{\vee, \&\}$, then the multiplication by random vectors from both sides does not work. However, Boolean matrix products can be verified even faster by the following simple algorithm:

THEOREM 6.2. *There exists a quantum Boolean matrix verification algorithm running in time* $O(n\sqrt{m})$ *and space* $O(\log n + \log m)$.

Proof. The condition that three given matrices form a valid product can be written as an And-Or tree: And of n^2 equalities, each being an Or of m products. There is a bounded-error quantum algorithm [12] running in time $O(\sqrt{n^2 m}) = O(n\sqrt{m})$ and space $O(\log(n^2 m))$.

By standard techniques [5], one can speed up the verification to time $O(n\sqrt{m/t})$, if the number of wrong entries t is known beforehand. If t is unknown, then the verification can be done in *expected time* $O(n\sqrt{m/t})$ and the worst-case time stays $O(n\sqrt{m})$. The Boolean matrix product with t nonzero entries can be thus computed in expected time $O(n\sqrt{tm})$.

6.3 Open problems It would be interesting to strengthen the lower bound on the fraction $\varepsilon(W, k)$ of marked pairs and thus also the upper bound on matrix verification. As we have shown, this cannot be done in full generality, but perhaps one can show a stronger lower bound using some density argument.

The time complexity of our algorithm goes up if the space complexity is bounded. Can one prove a time-space tradeoff for the verification problem similar to the tradeoff for matrix multiplication [14]? Note that we currently cannot show time-space tradeoffs for *any* decision problem.

Can one prove a better lower bound on matrix verification than $\Omega(n^{3/2})$? This lower bound is tight when there are $\sqrt{n}$ wrong entries. Is the true bound higher with only one wrong entry? Due to the small certificate complexity of this problem, one cannot prove such a bound using any of the adversary methods [19], but it might be provable by the polynomial method [4].

Acknowledgments

We thank Ronald de Wolf and Troy Lee for useful discussions. We thank anonymous referees for their valuable comments.

References

[1] A. Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64:750–767, 2002. Earlier version in STOC'00.

[2] A. Ambainis. Quantum walk algorithm for element distinctness. In *Proc. of 45th IEEE FOCS*, pages 22–31, 2004.

[3] A. Ambainis, H. Buhrman, P. Høyer, M. Karpinski, and P. Kurur. Quantum matrix verification. Unpublished Manuscript, 2002.

[4] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001. Earlier version in FOCS'98.

[5] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4–5):493–505, 1998. Earlier version in Physcomp'96.

[6] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. In *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *AMS Contemporary Mathematics Series*, pages 53–74. 2002.

[7] H. Buhrman, Ch. Dürr, M. Heiligman, P. Høyer, F. Magniez, M. Santha, and R. de Wolf. Quantum algorithms for element distinctness. In *Proc. of 16th IEEE Complexity*, pages 131–137, 2001.

[8] D. Coppersmith. Rectangular matrix multiplication revisited. *Journal of Complexity*, 13:42–49, 1997.

[9] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of symbolic computation*, 9:251–280, 1990. Earlier version in STOC'87.

[10] R. Freivalds. Fast probabilistic algorithms. In *Proc. of 8th Symp. on Math. Foundations of Computer Science*, pages 57–69. Springer Verlag, 1979. LNCS 74.

[11] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. of 28th ACM STOC*, pages 212–219, 1996.

[12] P. Høyer, M. Mosca, and R. de Wolf. Quantum search on bounded-error inputs. In *Proc. of 30th ICALP*, pages 291–299, 2003. LNCS 2719.

[13] P. Indyk. Output-sensitive algorithm for matrix multiplication. Manuscript, 2005.
[14] H. Klauck, R. Špalek, and R. de Wolf. Quantum and classical strong direct product theorems and optimal time-space tradeoffs. In *Proc. of 45th IEEE FOCS*, pages 12–21, 2004.
[15] D. E. Knuth. Combinatorial matrices. In *Selected Papers on Discrete Mathematics*, volume 106 of *CSLI Lecture Notes*. Stanford University, 2003.
[16] F. Magniez and A. Nayak. Quantum complexity of testing group commutativity. In *Proc. of 32nd ICALP*, pages 1312–1324, 2005. LNCS 3580.
[17] F. Magniez, M. Santha, and M. Szegedy. Quantum algorithms for the triangle problem. In *Proc. of 16th ACM-SIAM SODA*, pages 1109–1117, 2005.
[18] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
[19] R. Špalek and M. Szegedy. All quantum adversary methods are equivalent. In *Proc. of 32nd ICALP*, pages 1299–1311, 2005. LNCS 3580.
[20] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.
[21] M. Szegedy. Quantum speed-up of Markov chain based algorithms. In *Proc. of 45th IEEE FOCS*, pages 32–41, 2004.
[22] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47(1):85–93, 1986.
[23] R. Yuster and U. Zwick. Fast sparse matrix multiplication. In *Proc. of 12th RSA*, pages 604–615, 2004.

A Proofs for the fraction of revealing pairs

LEMMA A.1. *Let G be an integral domain with g elements, let (R,S) be marked, and let $\mathbf{p},\mathbf{q}$ be picked uniformly at random from G^n. The probability that $(R,S,\mathbf{p},\mathbf{q})$ is revealing is $\geq (1-\frac{1}{g})^2 \geq \frac{1}{4}$.*

Proof. Let $D = AB - C$, that is the wrong entries are exactly the nonzero entries of D. Assume that (R,S) is marked and pick any $D_{i_0,j_0} \neq 0$. Now, $(R,S,\mathbf{p},\mathbf{q})$ is not revealing iff

$$0 = \sum_{i\in R, j\in S} \mathbf{p}_i\mathbf{q}_j D_{i,j} = \sum_{\substack{i\in R, j\in S \\ D_{i,j}\neq 0}} \mathbf{p}_i\mathbf{q}_j D_{i,j}$$
$$= \mathbf{p}_{i_0}(\mathbf{q}_{j_0} D_{i_0,j_0} + c_1) + c_2\mathbf{q}_{j_0} + c_3,$$

where c_1, c_2, c_3 are some constants depending on D and other coordinates of $\mathbf{p},\mathbf{q}$. Fix these constants and pick $\mathbf{p}_{i_0}, \mathbf{q}_{j_0}$ at random from G. Since G is an integral domain with g elements, $p = \Pr[\mathbf{q}_{j_0} D_{i_0,j_0} + c_1 = 0] \leq \frac{1}{g}$. For every $\mathbf{q}_{j_0}$ such that $\mathbf{q}_{j_0} D_{i_0,j_0} + c_1 \neq 0$, the equality is equivalent to $c_4\mathbf{p}_{i_0} + c_5 = 0$ for another constants $c_4 \neq 0$ and c_5, which is again satisfied by at most 1 value of $\mathbf{p}_{i_0} \in G$. Hence the probability of having equality by chance is at most

$$p\cdot 1 + (1-p)\cdot\frac{1}{g} = \frac{1}{g} + p\cdot\frac{g-1}{g} \leq \frac{1}{g} + \frac{g-1}{g^2} = \frac{2g-1}{g^2}.$$

The probability that $(R,S,\mathbf{p},\mathbf{q})$ is revealing is thus at least $1 - \frac{2g-1}{g^2} = (1-\frac{1}{g})^2 \geq \frac{1}{4}$ and the equality holds when $g = 2$.

LEMMA A.2. *Let $0 \leq X \leq 1$ and $\mathrm{E}[X] \geq \alpha$. Then $\Pr[X \geq \beta] > \alpha - \beta$.*

Proof. Decompose the expected value of X conditioned on $X \geq \beta$: $\mathrm{E}[X] = \mathrm{E}[X|X<\beta]\cdot(1-\Pr[X\geq\beta]) + \mathrm{E}[X|X\geq\beta]\cdot\Pr[X\geq\beta]$. Rearrange, plug in $0 \leq X \leq 1$, and obtain $\Pr[X\geq\beta] = \frac{\mathrm{E}[X]-\mathrm{E}[X|X<\beta]}{\mathrm{E}[X|X\geq\beta]-\mathrm{E}[X|X<\beta]} > \frac{\alpha-\beta}{1-0} = \alpha-\beta$.

Proof. of LEMMA 4.2. Consider Boolean random variables $V_{R,S,\mathbf{p},\mathbf{q}} = 1$ iff $(R,S,\mathbf{p},\mathbf{q})$ is revealing. Let $v_{\mathbf{p},\mathbf{q}}$ be the *fraction of marked sets that are also revealing* when $\mathbf{p},\mathbf{q}$ multiply the equation, formally $v_{\mathbf{p},\mathbf{q}} = \mathrm{E}_{\mathrm{marked}\,(R,S)}[V_{R,S,\mathbf{p},\mathbf{q}}]$. By LEMMA A.1, for every marked (R,S), $\mathrm{E}_{\mathbf{p},\mathbf{q}}[V_{R,S,\mathbf{p},\mathbf{q}}] \geq \frac{1}{4}$. It follows that $\mathrm{E}_{\mathrm{marked}\,(R,S)}[\mathrm{E}_{\mathbf{p},\mathbf{q}}[V_{R,S,\mathbf{p},\mathbf{q}}]] \geq \frac{1}{4}$ and hence $\mathrm{E}_{\mathbf{p},\mathbf{q}}[\mathrm{E}_{\mathrm{marked}\,(R,S)}[V_{R,S,\mathbf{p},\mathbf{q}}]] = \mathrm{E}_{\mathbf{p},\mathbf{q}}[v_{\mathbf{p},\mathbf{q}}] \geq \frac{1}{4}$. By LEMMA A.2, when $\mathbf{p},\mathbf{q}$ is picked uniformly at random, $\Pr[v_{\mathbf{p},\mathbf{q}} \geq \frac{1}{8}] > \frac{1}{8}$. Hence in this "good" case, $\zeta_{\mathbf{p},\mathbf{q}}(W,k) \geq \frac{1}{8}\varepsilon(W,k)$.

B Proofs for the fraction of marked pairs

Proof. of LEMMA 5.1. Consider Boolean random variables $V_{R,S,W} = 1$ iff $W \cap R\times S \neq \emptyset$. Then for every $|R| = r$ and $|S| = s$, it holds that $\mathrm{E}_{W:|W|=t}[V_{R,S,W}] = \Pr_W[W\cap R\times S\neq\emptyset]$ and

$$\Pr_{W:|W|=t}[W\cap R\times S\neq\emptyset]$$
$$= 1 - \frac{n^2-rs}{n^2}\cdot\frac{n^2-rs-1}{n^2-1}\cdots\frac{n^2-rs-t+1}{n^2-t+1}$$
$$\geq 1 - \left(\frac{n^2-rs}{n^2}\right)^t \geq 1 - e^{-\frac{rs}{n^2}t} = \Omega\left(t\frac{rs}{n^2}\right),$$

because $1-x \leq e^{-x}$ and, on any fixed interval $x\in[0,A]$, also $e^{-x} \leq 1 - \frac{1-e^{-A}}{A}x$. The claim is now proved using standard arguments. Since $\forall R,S : \mathrm{E}_W[V_{R,S,W}] \geq t\frac{rs}{n^2}$, also $\mathrm{E}_{R,S}[\mathrm{E}_W[V_{R,S,W}]] \geq t\frac{rs}{n^2}$. Exchange the order of summation and obtain $\mathrm{E}_W[\mathrm{E}_{R,S}[V_{R,S,W}]] = \mathrm{E}_W[\varepsilon(W,r,s)] \geq t\frac{rs}{n^2}$.

Proof. of LEMMA 5.2. Let Z denote the random event "$W\cap R\times S\neq\emptyset$". For $j = 0,1,\ldots,w$, let Z_j denote the random event "$W\cap R\times[n]$ has exactly j nonempty rows". Since $\{Z_j\}$ are disjoint and $\sum_{j=0}^{w}\Pr[Z_j] = 1$, we

decompose the probability

$$\Pr[Z] = \sum_{j=0}^{w} \Pr[Z \mid Z_j] \cdot \Pr[Z_j] \geq \sum_{j=1}^{w} \Pr[Z_j] \cdot \Pr[Z \mid Z_1]$$
$$= (1 - \Pr[Z_0]) \cdot \Pr[Z \mid Z_1],$$

because $\Pr[Z \mid Z_j] \geq \Pr[Z \mid Z_1]$ for $j \geq 1$. Now, $\Pr[Z_0] = \Pr[W \cap R \times [n] = \emptyset] = \frac{n-w}{n} \cdot \frac{n-w-1}{n-1} \cdots \frac{n-w-r+1}{n-r+1} \leq (\frac{n-w}{n})^r = (1 - \frac{w}{n})^r \leq e^{-\frac{rw}{n}}$, because $1 - x \leq e^{-x}$. For every constant A, it holds that $e^{-x} \leq 1 - \frac{1-e^{-A}}{A}x$ on $x \in [0, A]$. By our assumption, $r \leq \frac{n}{w}$, hence $\frac{rw}{n} \leq 1$ and $e^{-\frac{rw}{n}} \leq 1 - \frac{1-e^{-1}}{1}\frac{rw}{n} = 1 - \alpha\frac{rw}{n}$ for $\alpha = 1 - e^{-1}$. We conclude that $1 - \Pr[Z_0] \geq \alpha\frac{rw}{n}$.

To lower-bound the other term, we decompose Z_1. For $i = 1, 2, \ldots, n$, let Y_i denote the random event "$W \cap R \times [n]$ has the i-th row nonempty and all other rows are empty". Let w_i be the number of entries in the i-th row of W. Since $\{Y_i\}$ are disjoint and $Y_1 \cup \cdots \cup Y_n = Z_1$,

$$\Pr[Z \mid Z_1] = \sum_{i:w_i \neq 0} \Pr[Z \mid Y_i] \cdot \Pr[Y_i \mid Z_1]$$
$$= \frac{1}{w} \sum_{i:w_i \neq 0} \Pr[Z \mid Y_i].$$

$\Pr[Z \mid Y_i]$ is easy to evaluate, since the i-th row of W contains w_i entries and S is picked uniformly at random. Let W_i be the i-th row of W. By the same arguments as above, $\Pr[Z \mid Y_i] = \Pr[W_i \cap S \neq \emptyset] = 1 - \Pr[W_i \cap S = \emptyset] \geq 1 - e^{-\frac{sw_i}{n}}$. By our assumption, $s \leq \frac{n}{w_i}$, hence $e^{-\frac{sw_i}{n}} \leq 1 - \alpha\frac{sw_i}{n}$ and $\Pr[Z \mid Y_i] \geq \alpha\frac{sw_i}{n}$.

Put both lower bounds together and obtain

$$\Pr[Z] \geq \alpha\frac{rw}{n} \cdot \frac{1}{w} \sum_i \alpha\frac{sw_i}{n} = \alpha^2\frac{rs}{n^2} \sum_i w_i = \Omega(|W|\frac{rs}{n^2}),$$

which is what we had to prove.

Proof. of Lemma 5.3. If $t \leq \sqrt{n}$, then the result follows from Lemma 5.2. Let us assume that $t > \sqrt{n}$. Again, let Z denote the random event "$W \cap R \times S \neq \emptyset$" and, for $j = 0, 1, \ldots, r$, let Z_j denote the random event "$W \cap R \times [n]$ has exactly j nonempty rows". Then

$$1 - \Pr[Z \mid Z_j] = \Pr[W \cap R \times S = \emptyset \mid Z_j]$$
$$= \frac{n-s}{n} \cdot \frac{n-s-1}{n-1} \cdots \frac{n-s-j+1}{n-j+1}$$
$$\leq \left(\frac{n-s}{n}\right)^j \leq e^{-\frac{sj}{n}}.$$

Since $j \leq r$ and $t > \sqrt{n}$, we get that $sj \leq rs \leq n^{4/3}/t^{2/3} \leq n^{4/3}/(\sqrt{n})^{2/3} = n$. Hence $\frac{sj}{n} \leq 1$ and by upper-bounding the exponential we get that $\Pr[Z \mid Z_j] \geq 1 - e^{-\frac{sj}{n}} \geq 1 - (1 - \alpha\frac{sj}{n}) = \alpha\frac{sj}{n}$ for $\alpha = 1 - e^{-1}$. Now, $\{Z_j\}$ are disjoint and $\sum_{j=0}^{r} \Pr[Z_j] = 1$, hence we decompose the probability

$$\Pr[Z] = \sum_{j=0}^{r} \Pr[Z \mid Z_j] \cdot \Pr[Z_j] \geq \sum_{j=0}^{r} \alpha\frac{sj}{n}\Pr[Z_j]$$
$$= \alpha\frac{s}{n} \sum_{j=0}^{r} j \cdot \Pr[Z_j] = \alpha\frac{s}{n} \cdot \mathrm{E}[Y],$$

where Y is the number of nonempty rows. There are r rows among n in R and we pick t entries without returning uniformly at random. An application of $\mathrm{E}[Y] = \frac{rt}{n}$ completes the proof.

C Matrix multiplication

In this section, we show how to use (the rectangular version of) *Matrix Verification* to obtain the actual position of a wrong entry. Furthermore, we present an algorithm for matrix multiplication. The algorithms are described in Figure 2.

Theorem C.1. Find Wrong Entry *has one-sided polynomially small error, worst-case running time* $\mathrm{O}(n^{2/3}m \log n)$*, and expected running time* $\mathrm{O}(n^{2/3}m \log n/q(W)^{1/3})$ *for the set of wrong entries* W*.*

Proof. Assume that $AB \neq C$. Let W^ℓ be the set of wrong entries in the ℓ-th recursion level of the binary search. From the definition of $q(W)$, if $q(W^\ell) = q$, then $q(W^\ell_{i,j}) \geq \frac{q}{4}$ for at least one quadrant $i, j \in \{1, 2\}$. *Find Wrong Entry* descends into the first quadrant it finds a solution in, hence it chooses $W^{\ell+1} = W^\ell_{i,j}$ with high probability and then $(\frac{n}{2})^2/q(W^{\ell+1}) \leq n^2/q(W^\ell)$. There are $\log n$ levels of the recursion. Hence its expected running time is at most

$$\sum_{\ell=1}^{\log n} 4\sqrt[3]{\frac{(n/2^\ell)^2}{q(W^\ell)}}m \leq \sum_{\ell=1}^{\log n} 4\sqrt[3]{\frac{n^2}{q(W)}}m = \frac{n^{2/3}m}{q(W)^{1/3}} \sum_{\ell=1}^{\log n} 4$$
$$= \mathrm{O}\left(\frac{n^{2/3}m}{q(W)^{1/3}} \log n\right),$$

as claimed. By Theorem 4.2, the probability that a wrong matrix product is not recognized in one iteration is at most $\frac{1}{3}$. The probability that it is not recognized in $\mathrm{O}(\log n)$ iterations is $1/\mathrm{poly}(n)$. If $AB = C$, then the first iteration of binary search is repeated $\mathrm{O}(\log n)$ times and the worst-case running time is $\mathrm{O}(n^{2/3}m \log n)$.

Remark C.1. *It might be that the position of the wrong entry can be obtained from just one call to* Matrix Verification *in the same way as in the quantum walk algorithm for element distinctness [2] – by measuring*

Matrix Multiplication (input size n, m, matrices $A_{n\times m}, B_{m\times n}$) returns $C_{n\times n} = AB$:

1. Initialize $C = 0$.
2. Run *Find Wrong Entry* (n, m, A, B, C). If it returns "equal", return C.
3. Otherwise let (r, c) be the wrong position. Recompute $C_{r,c}$. Find and recompute all wrong entries in the r-th row using the *Grover Search*. Find and recompute all wrong entries in the c-th column using the *Grover Search*.
4. Go to step 2.

Find Wrong Entry (input size n, m, matrices $A_{n\times m}, B_{m\times n}, C_{n\times n}$) returns a position (r, c) if $C_{r,c} \neq \sum_i A_{r,i}B_{i,c}$, or "equal" if $AB = C$:

1. If $n = 1$, verify the scalar product and exit.
2. Let A_1, A_2 denote the top and bottom half of A, let B_1, B_2 denote the left and right half of B, and let $C_{1,1}, C_{1,2}, C_{2,1}, C_{2,2}$ denote the four quadrants of C.
3. Repeat at most $\mathrm{O}(\log n)$ times the following:
   - Run in parallel *Matrix Verification* $(\frac{n}{2}, m, A_i, B_j, C_{i,j})$ for $i, j \in \{1, 2\}$.
   - If some of them returns "not equal", stop the other threads of computation and cancel the loop.
4. If the matrix verification was always successful, return "equal".
5. Let $C_{i,j} \neq A_iB_j$ be the found wrong submatrix. Let $(r', c') =$ *Find Wrong Entry* $(\frac{n}{2}, m, A_i, B_j, C_{i,j})$.
6. If $i = 1$, set $r = r'$, otherwise set $r = r' + \frac{n}{2}$. If $j = 1$, set $c = c'$, otherwise set $c = c' + \frac{n}{2}$. Return (r, c).

Figure 2: Quantum algorithm for matrix multiplication

the subsets R, S instead of the quantum coin register $|z\rangle$. However, this is only known to follow from Theorem 4.1 *for exactly one wrong entry, that is $|W| = 1$ [21, Section 10]. The log-factor in the total running time is necessary for having a polynomially small error.*

We prove the upper bound on matrix multiplication.

Proof. of Theorem 6.1. Finding all r_ℓ wrong entries in the ℓ-th row is done by the Grover search with an unknown number of solutions [5], and it takes time $\sum_{i=1}^{r_\ell} \sqrt{\frac{n}{i}}m = \mathrm{O}(\sqrt{nr_\ell}m)$, where the scalar products of length m are computed on-line. We ensure that there are no wrong entries left with probability polynomially close to one in an additional time $\mathrm{O}(\sqrt{n}m\log n)$. Let us condition the rest of the analysis by that the Grover searches indeed find all ones.

Let W' be the largest independent subset of W. Clearly, *Matrix Multiplication* finishes in at most $|W'|$ iterations, otherwise there would exist an independent set larger than $|W'|$. The total running time is the sum of the time spent in *Find Wrong Entry*

$$T_F \leq \sum_{\ell=1}^{|W'|} \frac{n^{2/3}m\log n}{|W'|^{1/3}} = \mathrm{O}((n|W'|)^{2/3}m\log n),$$

and the time spent in the Grover searches. By applying a Cauchy-Schwarz inequality several times,

$$\begin{aligned} T_G &\leq \sum_{\ell=1}^{|W'|} (\sqrt{nr_\ell}m + \sqrt{nc_\ell}m)\log n \\ &= m\sqrt{n}\log n \left(\sum_{\ell=1}^{|W'|} 1\cdot\sqrt{r_\ell} + \sum_{\ell=1}^{|W'|} 1\cdot\sqrt{c_\ell}\right) \\ &\leq m\sqrt{n}\log n\sqrt{\sum_{\ell=1}^{|W'|} 1}\left(\sqrt{\sum_{\ell=1}^{|W'|} r_\ell} + \sqrt{\sum_{\ell=1}^{|W'|} c_\ell}\right) \\ &= \mathrm{O}(m\sqrt{n}\log n\sqrt{|W'|}\sqrt{|W|}). \end{aligned}$$

The algorithm is bounded-error, because both *Find Wrong Entry* and the iterated Grover searches have polynomially small error. Put the bounds together and obtain:

$$T_F + T_G \leq m\log n\sqrt{n}\sqrt{|W'|}\cdot(n^{1/6}|W'|^{1/6} + \sqrt{|W|}).$$

Evaluate separately the three cases $|W| \in [1, \sqrt{n}]$, $|W| \in [\sqrt{n}, n]$, and $|W| \in [n, n^2]$, use that $|W'| \leq |W|$ and $|W'| \leq n$, and obtain inequality (6.6), which we had to prove.