Quantum Algorithms for Matrix Products over Semirings

François Le Gall[1] and Harumichi Nishimura[2]

[1] Graduate School of Information Science and Technology, The University of Tokyo, Japan
[2] Graduate School of Information Science, Nagoya University, Japan

Abstract. In this paper we construct quantum algorithms for matrix products over several algebraic structures called semirings, including the $(\max, \min)$-matrix product, the distance matrix product and the Boolean matrix product. In particular, we obtain the following results.

– We construct a quantum algorithm computing the product of two $n \times n$ matrices over the $(\max, \min)$ semiring with time complexity $O(n^{2.473})$. In comparison, the best known classical algorithm for the same problem has complexity $O(n^{2.687})$. As an application, we obtain a $O(n^{2.473})$-time quantum algorithm for computing the all-pairs bottleneck paths of a graph with n vertices, while classically the best upper bound for this task is $O(n^{2.687})$.
– We construct a quantum algorithm computing the ℓ most significant bits of each entry of the distance product of two $n \times n$ matrices in time $O(2^{0.64\ell} n^{2.46})$. In comparison, prior to the present work, the best known classical algorithm for the same problem had complexity $O(2^{\ell} n^{2.69})$. Our techniques lead to further improvements for classical algorithms as well, reducing the classical complexity to $O(2^{0.96\ell} n^{2.69})$, which gives a sublinear dependency on 2^{ℓ}.

The above two algorithms are the first quantum algorithms that perform better than the $\tilde{O}(n^{5/2})$-time straightforward quantum algorithm based on quantum search for matrix multiplication over these semirings. We also consider the Boolean semiring, and construct a quantum algorithm computing the product of two $n \times n$ Boolean matrices that outperforms the best known classical algorithms for sparse matrices.

1 Introduction

Background. Matrix multiplication over semirings has a multitude of applications in computer science, and in particular in the area of graph algorithms (e.g., [5,18,19,20,21,23]). One example is Boolean matrix multiplication, related for instance to the computation of the transitive closure of a graph, where the product of two $n \times n$ Boolean matrices A and B is defined as the $n \times n$ Boolean matrix $C = A \cdot B$ such that $C[i, j] = 1$ if and only if there exists a $k \in \{1, \ldots, n\}$ such that $A[i, k] = B[k, j] = 1$.

More generally, given a set $R \subseteq \mathbb{Z} \cup \{-\infty, \infty\}$ and two binary operations $\oplus \colon R \times R \to R$ and $\odot \colon R \times R \to R$, the structure $(R, \oplus, \odot)$ is a semiring if it

© Springer International Publishing Switzerland 2014

behaves like a ring except that there is no requirement on the existence of an inverse with respect to the operation $\oplus$. Given two $n \times n$ matrices A and B over R, the matrix product over $(R, \oplus, \odot)$ is the $n \times n$ matrix C defined as $C[i,j] = \bigoplus_{k=1}^{n} (A[i,k] \odot B[k,j])$ for any $(i,j) \in \{1,\ldots,n\} \times \{1,\ldots,n\}$. The Boolean matrix product is simply the matrix product over the semiring $(\{0,1\}, \vee, \wedge)$. The $(\max,\min)$-product and the distance product, which both have applications to a multitude of tasks in graph theory such as constructing fast algorithms for all-pairs paths problems (see, e.g., [19]), are the matrix products over the semiring $(\mathbb{Z} \cup \{-\infty, \infty\}, \max, \min)$ and the semiring $(\mathbb{Z} \cup \{\infty\}, \min, +)$, respectively.

Whenever the operation $\oplus$ is such that a term as $\bigoplus_{k=1}^{n} x_k$ can be computed in $\tilde{O}(\sqrt{n})$ time using quantum techniques (e.g., for $\oplus = \vee$ using Grover's algorithm [8] or for $\oplus = \min$ and $\oplus = \max$ using quantum algorithms for minimum finding [7]) and each operation $\odot$ can be implemented in polylog(n) time, the product of two $n \times n$ matrices over the semiring $(R, \oplus, \odot)$ can be computed in time $\tilde{O}(n^{5/2})$ on a quantum computer.[1] This is true for instance for the Boolean matrix product, and for both the $(\max,\min)$ and distance matrix products.

A fundamental question is whether we can do better than those $\tilde{O}(n^{5/2})$-time straightforward quantum algorithms. For the Boolean matrix product, the answer is affirmative since it can be computed classically in time $\tilde{O}(n^{\omega})$, where $\omega < 2.373$ is the exponent of square matrix multiplication over a field. However, Boolean matrix product appears to be an exception, and for most semirings it is not known if matrix multiplication can be done in $\tilde{O}(n^{\omega})$-time. For instance, the best known classical algorithm for the $(\max,\min)$-product, by Duan and Pettie [5], has time complexity $\tilde{O}(n^{(3+\omega)/2}) = O(n^{2.687})$ while, for the distance product, no truly subcubic classical algorithm is even known.

Our Results. We construct in this paper the first quantum algorithms with exponent strictly smaller than 5/2 for matrix multiplication over several semirings.

We first obtain (in Section 4.1) the following result for multiplication over the $(\max,\min)$ semiring.

Theorem 1. *There exists a quantum algorithm that computes, with high probability, the* $(\max,\min)$*-product of two* $n \times n$ *matrices in time* $O(n^{2.473})$.

In comparison, the best known classical algorithm for the $(\max,\min)$-product, by Duan and Pettie [5], has time complexity $\tilde{O}(n^{(3+\omega)/2}) = O(n^{2.687})$, as mentioned above. The $(\max,\min)$-product has mainly been studied in the field in fuzzy logic [6] under the name *composition of relations* and in the context of computing the all-pairs bottleneck paths of a graph (i.e., computing, for all pairs (s,t) of vertices in a graph, the maximum flow that can be routed between s and t). More precisely, it is well known (see, e.g., [5,18,21]) that if the $(\max,\min)$-product of two $n \times n$ matrices can be computed in time $T(n)$, then the all-pairs bottleneck paths of a graph with n vertices can be computed in time $\tilde{O}(T(n))$. As an application of Theorem 1, we thus obtain a $O(n^{2.473})$-time quantum algorithm

[1] In this paper the notation $\tilde{O}(\cdot)$ suppresses the $n^{o(1)}$ factors.

computing the all-pairs bottleneck paths of a graph of n vertices, while classically the best upper bound for this task is $O(n^{2.687})$, again from [5].

In order to prove Theorem 1, we construct a quantum algorithm that computes the product of two $n \times n$ matrices over the existence dominance semiring (defined in Section 2) in time $\tilde{O}(n^{(5+\omega)/3}) \leq O(n^{2.458})$. The dominance product has applications in computational geometry [17] and graph algorithms [20] and, in comparison, the best known classical algorithm for this product [23] has complexity $O(n^{2.684})$. Computing efficiently the existence dominance product is, nevertheless, not enough for our purpose. We introduce (in Section 3) a new generalization of it that we call the *generalized existence dominance product*, and construct both quantum and classical algorithms that compute this product.

We also show (in Section 4.2) how these results for the generalized existence dominance product can be used to construct classical and quantum algorithms computing the ℓ most significant bits of each entry of the distance product of two $n \times n$ matrices. In the quantum setting, we obtain time complexity $\tilde{O}\left(2^{0.640\ell}n^{(5+\omega)/3}\right) \leq O(2^{0.640\ell}n^{2.458})$. In comparison, prior to the present work, the best known classical algorithm for the same problem by Vassilevska and Williams [20] had time complexity $\tilde{O}\left(2^{\ell}n^{(3+\omega)/2}\right) \leq O(2^{\ell}n^{2.687})$, with a slight improvement on the exponent of n obtained later by Yuster [23]. We obtain an improvement for this classical time complexity as well, reducing it to $\tilde{O}\left(2^{0.960\ell}n^{(3+\omega)/2}\right)$, which gives a sublinear dependency on 2^{ℓ}.

These results are, to the best of our knowledge, the first quantum algorithms for matrix multiplication over semirings other than the Boolean semiring improving over the straightforward $\tilde{O}(n^{5/2})$-time quantum algorithm, and the first nontrivial quantum algorithms offering a speedup with respect to the best classical algorithms for matrix multiplication when no assumptions are made on the sparsity of the matrices involved (sparse matrix multiplication is discussed below). This shows that, while quantum algorithms may not be able to outperform the classical $\tilde{O}(n^{\omega})$-time algorithm for matrix multiplication of (dense) matrices over a ring, they can offer a speedup for matrix multiplication over other algebraic structures.

We finally investigate under which conditions quantum algorithms faster than the best known classical algorithms can be constructed for Boolean matrix multiplication. This question has been recently studied extensively in the output-sensitive scenario [3,10,12,13], for which quantum algorithms multiplying two $n \times n$ Boolean matrices with query complexity $\tilde{O}(n\sqrt{\lambda})$ and time complexity $\tilde{O}(n\sqrt{\lambda} + \lambda\sqrt{n})$ were constructed, where λ denotes the number of non-zero entries in the output matrix. In this work, we focus on the case where the input matrices are sparse (but not necessarily the output matrix), and obtain the following result.

Theorem 2 (simplified version). *Let A and B be two $n \times n$ Boolean matrices each containing at most m non-zero entries. There exists a quantum algorithm that computes, with high probability, the Boolean matrix product $A \cdot B$ and has time complexity*

$$\begin{cases} \tilde{O}(n^2) & \textit{if } m \leq n^{1.151}, \\ \tilde{O}\left(m^{0.517} n^{1.406}\right) & \textit{if } n^{1.151} \leq m \leq n^{\omega-1/2}, \\ \tilde{O}(n^{\omega}) & \textit{if } n^{\omega-1/2} \leq m \leq n^2. \end{cases}$$

In comparison, the best known classical algorithm, by Yuster and Zwick [24], has complexity $\tilde{O}(n^2)$ if $m \leq n^{1.151}$, $\tilde{O}(m^{0.697} n^{1.199})$ if $n^{1.151} \leq m \leq n^{(1+\omega)/2}$, and $\tilde{O}(n^{\omega})$ if $n^{(1+\omega)/2} \leq m \leq n^2$. Our algorithm performs better when $n^{1.151} < m < n^{\omega-1/2}$. For instance, if $m = O(n^{(1+\omega)/2}) = O(n^{1.686\cdots})$, then our algorithm has complexity $O(n^{2.277})$, while the algorithm in [24] has complexity $\tilde{O}(n^{\omega})$. The complete statement of Theorem 2, and its proof, are given in the full version of the present paper [15].

Our main quantum tool is rather standard: quantum enumeration, a variant of Grover's search algorithm. We use this technique in various ways to improve the combinatorial steps in several classical approaches [1,5,21,24] that are based on a combination of algebraic steps (computing some matrix products over a field) and combinatorial steps. Moreover, the speedup obtained by quantum enumeration enables us to depart from these original approaches and optimize the combinatorial and algebraic steps in different ways, for instance relying on rectangular matrix multiplication instead of square matrix multiplication. On the other hand, several subtle but crucial issues appear when trying to apply quantum enumeration, such as how to store and access information computed during the preprocessing steps, which induces complications and requires the introduction of new algorithmic ideas. We end up with algorithms fairly remote from these original approaches, where most steps are tailored for the use of quantum enumeration.

2 Preliminaries

Rectangular Matrix Multiplication over Fields. For any $k_1, k_2, k_3 > 0$, let $\omega(k_1, k_2, k_3)$ represent the minimal value τ such that, over a field, the product of an $n^{k_1} \times n^{k_2}$ matrix by an $n^{k_2} \times n^{k_3}$ matrix can be computed with $\tilde{O}(n^{\tau})$ arithmetic operations. The value $\omega(1,1,1)$ is denoted by ω, and the current best upper bound on ω is $\omega < 2.373$, see [14,22]. Other important quantities are the value $\alpha = \sup\{k \mid \omega(1,k,1) = 2\}$ and the value $\beta = (\omega - 2)/(1-\alpha)$. The current best lower bound on α is $\alpha > 0.302$, see [11]. The following facts are known, and will be used in this paper. We refer to [4,9] for details.

Fact 1. $\omega(1,k,1) = 2$ *for* $k \leq \alpha$ *and* $\omega(1,k,1) \leq 2 + \beta(k-\alpha)$ *for* $\alpha \leq k \leq 1$.

Fact 2. *The following relations hold for any values* $k_1, k_2, k_3 > 0$*: (i) for any* $k > 0$, $\omega(kk_1, kk_2, kk_3) = k\omega(k_1, k_2, k_3)$*; (ii)* $\omega(k_{\pi(1)}, k_{\pi(2)}, k_{\pi(3)}) = \omega(k_1, k_2, k_3)$ *for any permutation* π *over* $\{1,2,3\}$*; (iii)* $\omega(k_1, k_2, 1+k_3) \leq \omega(k_1, k_2, 1) + k_3$*; (iv)* $\omega(k_1, k_2, k_3) \geq \max\{k_1+k_2, k_1+k_3, k_2+k_3\}$.

Matrix Products over Semirings. We define below two matrix products over semirings considered in Sections 3 and 4, respectively, additionally to the

Boolean product, the $(\max, \min)$-product and the distance product defined in the introduction. These products were also used in [5,20,21].

Definition 1. *Let A be an $n \times n$ matrix with entries in $\mathbb{Z} \cup \{\infty\}$ and B be an $n \times n$ matrix with entries in $\mathbb{Z} \cup \{-\infty\}$. The existence dominance product of A and B, denoted $A * B$, is the $n \times n$ Boolean matrix C such that $C[i,j] = 1$ if and only if there exists some $k \in \{1, \ldots, n\}$ such that $A[i,k] \leq B[k,j]$. The product $A \lhd B$ is the $n \times n$ matrix C such that $C[i,j] = -\infty$ if $A[i,k] > B[k,j]$ for all $k \in \{1, \ldots, n\}$, and $C[i,j] = \max_k\{A[i,k] \mid A[i,k] \leq B[k,j]\}$ otherwise.*

It is easy to check, as mentioned for instance in [5,21], that computing the $(\max, \min)$-product reduces to computing the product $\lhd$. Indeed if C denotes the $(\max, \min)$-product of two matrices A and B, then for any $(i,j) \in \{1, \ldots, n\} \times \{1, \ldots, n\}$ we can write $C[i,j] = \max\left\{(A \lhd B)[i,j], (B^T \lhd A^T)[j,i]\right\}$, where A^T and B^T denote the transposes of A and B, respectively. Matrix products over the semirings $(\min, \max)$, $(\min, \leq)$ and $(\max, \geq)$ studied, for instance, in [19], similarly reduce to computing the product $\lhd$.

Quantum Algorithms for Matrix Multiplication. We assume that a quantum algorithm can access any entry of the input matrix in a random access way, similarly to the standard model used in [3,10,12,13] for Boolean matrix multiplication.

We will use variants of Grover's search algorithm, as described for instance in [2], to find elements satisfying some conditions inside a search space of size N. Concretely, suppose that a Boolean function $f\colon \{1, \ldots, N\} \to \{0,1\}$ is given and that we want to find a solution, i.e., an element $x \in \{1, \ldots, n\}$ such that $f(x) = 1$. Consider the quantum search procedure (called safe Grover search in [16]) obtained by repeating Grover's standard search a logarithmic number of times, and checking if a solution has been found. This quantum procedure outputs one solution with probability at least $1 - 1/\mathrm{poly}(N)$ if a solution exists, and always rejects if no solution exists. Its time complexity is $\tilde{O}(\sqrt{N/\max(1,t)})$, where t denotes the number of solutions, if the function f can be evaluated in $\tilde{O}(1)$ time. By repeating this procedure and striking out solutions as soon as they are found, one can find all the solutions with probability at least $1 - 1/\mathrm{poly}(N)$ using $\tilde{O}\left(\sqrt{N/t} + \sqrt{N/(t-1)} + \cdots + \sqrt{N/1}\right) = \tilde{O}(\sqrt{N(t+1)})$ computational steps. We call this procedure *quantum enumeration.*

3 Existence Dominance Matrix Multiplication

In this section we present a quantum algorithm that computes the existence dominance product of two matrices A and B. The underlying idea of our algorithm is similar to the idea in the best classical algorithm for the same problem by Duan and Pettie [5]: use a search step to find some of the entries of $A * B$, and rely on classical algebraic algorithms to find the other entries. We naturally use quantum search to implement the first part, and perform careful modifications of their approach to improve the complexity in the quantum setting, taking advantage of the features of quantum enumeration. There are two notable differences: The first one is that the algebraic part of our quantum algorithms uses rectangular matrix multiplication, while [5] uses square matrix multiplication. The second and crucial difference is that, for applications in later sections, we give a quantum algorithm that can handle a new (and more general) version of the existence dominance product, defined on set of matrices, which we call the *generalized existence dominance product* and define below.

Definition 2. *Let u, v be two positive integers, and S be the set $S = \{1,\ldots,u\} \times \{1,\ldots,v\}$. Let $\prec$ be the lexicographic order over $S \cup \{(0,0)\}$ (i.e., $(i,j) \prec (i',j')$ if and only if $i < i'$ or ($i = i'$ and $j < j'$)). Consider u matrices $A^{(1)},\ldots,A^{(u)}$, each of size $n \times n$ with entries in $\mathbb{Z} \cup \{\infty\}$, and v matrices $B^{(1)},\ldots,B^{(v)}$, each of size $n \times n$ with entries in $\mathbb{Z} \cup \{-\infty\}$. For each $(i,j) \in \{1,\ldots,n\} \times \{1,\ldots,n\}$ define the set $S_{ij} \subseteq S \cup \{(0,0)\}$ as follows:*

$$S_{ij} = \{(x,y) \in S \mid A^{(x)} * B^{(y)}[i,j] = 1\} \cup \{(0,0)\}.$$

The generalized existence dominance product of these matrices is the $n \times n$ matrix C with entries in $S \cup \{(0,0)\}$ defined as follows: for all $(i,j) \in \{1,\ldots,n\} \times \{1,\ldots,n\}$ the entry $C[i,j]$ is the maximum element in S_{ij}, where the maximum refers to the lexicographic order.

Note that the case $u = v = 1$ corresponds to the standard existence dominance product, since $C[i,j] = (1,1)$ if $A^{(1)} * B^{(1)}[i,j] = 1$ and $C[i,j] = (0,0)$ if $A^{(1)} * B^{(1)}[i,j] = 0$.

Proposition 1. *Let $A^{(1)},\ldots,A^{(u)}$ be u matrices of size $n \times n$ with entries in $\mathbb{Z} \cup \{\infty\}$, and $B^{(1)},\ldots,B^{(v)}$ be v matrices of size $n \times n$ with entries in $\mathbb{Z} \cup \{-\infty\}$. Let $m_1 \in \{1,\ldots,n^2u\}$ denote the total number of finite entries in the matrices $A^{(1)},\ldots,A^{(u)}$, and $m_2 \in \{1,\ldots,n^2v\}$ denote the total number of finite entries in the matrices $B^{(1)},\ldots,B^{(v)}$. For any parameter $t \in \{1,\ldots,m_1\}$, there exists a quantum algorithm that computes, with high probability, their generalized existence dominance product in time*

$$\tilde{O}\left(\sqrt{\frac{m_1 m_2 n}{t}} + \sqrt{\frac{m_1 m_2 uv}{tn}} + n^{\omega(1+\log_n u, 1+\log_n t, 1+\log_n v)}\right).$$

Proof. Let $t \in \{1,\ldots,m_1\}$ be a parameter to be chosen later. Let L be the list of all finite entries in $A^{(1)},\ldots,A^{(u)}$ sorted in increasing order. Decompose L into t successive parts $L_1,\ldots,L_t$, each containing at most $\lceil m_1/t \rceil$ entries. For each $x \in \{1,\ldots,u\}$ and each $r \in \{1,\ldots,t\}$ we construct two $n \times n$ matrices $A_r^{(x)}, \bar{A}_r^{(x)}$ as follows: for all $(i,j) \in \{1,\ldots,n\} \times \{1,\ldots,n\}$,

$$A_r^{(x)}[i,j] = \begin{cases} A^{(x)}[i,j] \text{ if } A^{(x)}[i,j] \in L_r, \\ \infty \text{ otherwise,} \end{cases} \qquad \bar{A}_r^{(x)}[i,j] = \begin{cases} 1 \text{ if } A^{(x)}[i,j] \in L_r, \\ 0 \text{ otherwise.} \end{cases}$$

Similarly, for each $y \in \{1,\ldots,v\}$ and each $r \in \{1,\ldots,t\}$ we construct two $n \times n$ matrices $B_r^{(y)}, \bar{B}_r^{(y)}$ as follows: for all $(i,j) \in \{1,\ldots,n\} \times \{1,\ldots,n\}$,

$$B_r^{(y)}[i,j] = \begin{cases} B^{(y)}[i,j] \text{ if } \min L_r \leq B^{(y)}[i,j] < \max L_r, \\ -\infty \text{ otherwise,} \end{cases}$$

$$\bar{B}_r^{(y)}[i,j] = \begin{cases} 1 \text{ if } B^{(y)}[i,j] \geq \max L_r, \\ 0 \text{ otherwise.} \end{cases}$$

The cost of this (classical) preprocessing step is $O(n^2 t(u+v))$ time.

It is easy to see that, for each $x \in \{1,\ldots,u\}$ and $y \in \{1,\ldots,v\}$, the following equality holds (where the operators $+$ and $\sum$ refer to the entry-wise OR):

$$A^{(x)} * B^{(y)} = \sum_{r=1}^{t} \left(\bar{A}_r^{(x)} \cdot \bar{B}_r^{(y)} \right) + \sum_{r=1}^{t} \left(A_r^{(x)} * B_r^{(y)} \right). \tag{1}$$

Indeed, the second term compares entries that are in a same part L_r, while the first term takes into consideration entries in distinct parts. Define two $n \times n$ matrices C_1 and C_2 with entries in $S \cup \{(0,0)\}$ as follows: for all $(i,j) \in \{1,\ldots,n\} \times \{1,\ldots,n\}$,

$$C_1[i,j] = \max \left\{ \{(0,0)\} \cup \{(x,y) \in S \mid \sum_{r=1}^{t} \bar{A}_r^{(x)} \cdot \bar{B}_r^{(y)}[i,j] = 1\} \right\}, \tag{2}$$

$$C_2[i,j] = \max \left\{ \{(0,0)\} \cup \{(x,y) \in S \mid \sum_{r=1}^{t} A_r^{(x)} * B_r^{(y)}[i,j] = 1\} \right\}. \tag{3}$$

From Equation (1), the generalized existence dominance product C satisfies $C[i,j] = \max\{C_1[i,j], C_2[i,j]\}$ for all $(i,j) \in \{1,\ldots,n\} \times \{1,\ldots,n\}$. The matrix C can then be computed in time $O(n^2)$ from C_1 and C_2.

The matrix C_1 can clearly be computed in time $O(n^2 uv)$ if all the terms $\sum_r \bar{A}_r^{(x)} \cdot \bar{B}_r^{(y)}$ are known. We can obtain all these uv terms by computing the following Boolean product of an $nu \times nt$ matrix by an $nt \times nv$ matrix (both matrices can be constructed in time $\tilde{O}(n^2 t(u+v))$).

$$\begin{bmatrix} \bar{A}_1^{(1)} & \cdots & \bar{A}_t^{(1)} \\ \vdots & & \vdots \\ \bar{A}_1^{(u)} & \cdots & \bar{A}_t^{(u)} \end{bmatrix} \cdot \begin{bmatrix} \bar{B}_1^{(1)} & \cdots\cdots & \bar{B}_1^{(v)} \\ \vdots & & \vdots \\ \bar{B}_t^{(1)} & \cdots\cdots & \bar{B}_t^{(v)} \end{bmatrix}$$

The cost of this matrix multiplication is $\tilde{O}\left(n^{\omega(1+\log_n u, 1+\log_n t, 1+\log_n v)}\right)$. From item (iv) of Fact 2, we conclude that the matrix C_1 can be computed in time

$$\tilde{O}\left(n^2 uv + n^2 t(u+v) + n^{\omega(1+\log_n u, 1+\log_n t, 1+\log_n v)}\right)$$

$$= \tilde{O}\left(n^{\omega(1+\log_n u, 1+\log_n t, 1+\log_n v)}\right).$$

We now explain how to compute the matrix C_2. Intuitively, the main difficulty is that Equation (3) cannot be used directly since we do not know how to compute the dominance product $*$ efficiently. Lemma 1 below shows that it is possible to replace this dominance product by a Boolean product if we replace the matrices $A_r^{(x)}$ and $B_r^{(y)}$ by some Boolean matrices $\hat{A}_r^{(x)}$ and $\hat{B}_r^{(y)}$ (compare Equation (3) with Equation (4) below). This lemma further shows that the latter matrices can be computed efficiently by a quantum algorithm (based on quantum search). Actually, for technical reasons we additionally need to replace the term $\{(0,0)\}$ in Equation (3) by the term $\{D[i,j]\}$ in Equation (4), where D is a matrix that can also be computed efficiently using a quantum algorithm. While this lemma is the main technical part of the proof of this proposition, due to space constraints its proof is omitted (we refer to [15] for all details).

Lemma 1. *There exists a quantum algorithm that, with high probability, outputs*

- *tu Boolean matrices $\hat{A}_r^{(x)}$, each of size $n \times 2n$, for all $x \in \{1,\ldots,u\}$ and $r \in \{1,\ldots,t\}$,*
- *tv Boolean matrices $\hat{B}_r^{(y)}$, each of size $2n \times n$, for all $y \in \{1,\ldots,v\}$ and $r \in \{1,\ldots,t\}$,*
- *a matrix D of size $n \times n$ with entries in $S \cup \{(0,0)\} = (\{1,\ldots,u\} \times \{1,\ldots,v\}) \cup \{(0,0)\}$,*

such that

$$C_2[i,j] = \max\left\{\{D[i,j]\} \cup \{(x,y) \in S \mid \sum_{r=1}^{t} \hat{A}_r^{(x)} \cdot \hat{B}_r^{(y)}[i,j] = 1\}\right\} \tag{4}$$

for all $(i,j) \in \{1,\ldots,n\} \times \{1,\ldots,n\}$. The time complexity of this quantum algorithm is

$$\tilde{O}\left(n^2 t(u+v) + \sqrt{\frac{m_1 m_2 n}{t}} + \sqrt{\frac{m_1 m_2 uv}{tn}}\right).$$

After applying the quantum algorithm of Lemma 1, we can obtain the matrix C_2, similarly to the computation of C_1, if we know all the terms $\sum_r \hat{A}_r^{(x)} \cdot \hat{B}_r^{(y)}$. we obtain all these uv terms by computing the following Boolean product of an $nu \times nt$ matrix by an $nt \times nv$ matrix.

$$\begin{bmatrix} \hat{A}_1^{(1)} & \cdots & \hat{A}_t^{(1)} \\ \vdots & & \vdots \\ \hat{A}_1^{(u)} & \cdots & \hat{A}_t^{(u)} \end{bmatrix} \cdot \begin{bmatrix} \hat{B}_1^{(1)} & \cdots\cdots & \hat{B}_1^{(v)} \\ \vdots & & \vdots \\ \hat{B}_t^{(1)} & \cdots\cdots & \hat{B}_t^{(v)} \end{bmatrix}$$

The cost of this matrix multiplication is $\tilde{O}\left(n^{\omega(1+\log_n u, 1+\log_n t, 1+\log_n v)}\right)$. The total cost of computing the matrix C_2 is thus

$$\tilde{O}\left(n^2 t(u+v) + \sqrt{\frac{m_1 m_2 n}{t}} + \sqrt{\frac{m_1 m_2 uv}{tn}} + n^{\omega(1+\log_n u, 1+\log_n t, 1+\log_n v)}\right),$$

which is the desired bound since the term $n^2 t(u+v)$ is negligible here by item (iv) of Fact 2. □

We can also give a classical version of the algorithm of Proposition 1, as stated in the following proposition (see [15] for a proof).

Proposition 2. *There exists a classical algorithm that computes the generalized existence dominance product in time* $\tilde{O}\left(\frac{m_1 m_2}{tn} + n^{\omega(1+\log_n u, 1+\log_n t, 1+\log_n v)}\right)$, *for any parameter* $t \in \{1, \ldots, m_1\}$.

We now consider the case $u = v = 1$ corresponding to the standard existence dominance product. By optimizing the choice of the parameter t in Proposition 1, we obtain the following theorem.

Theorem 3. *Let A be an $n \times n$ matrix with entries in $\mathbb{Z} \cup \{\infty\}$ containing at most m_1 non-(∞) entries, and B be an $n \times n$ matrix with entries in $\mathbb{Z} \cup \{-\infty\}$ containing at most m_2 non-$(-\infty)$ entries. There exists a quantum algorithm that computes, with high probability, the existence dominance product of A and B in time $\tilde{O}(\sqrt{m_1 m_2 n^{1-\mu}})$, where μ is the solution of the equation $\mu + 2\omega(1, 1+\mu, 1) = 1 + \log_n(m_1 m_2)$. In particular, this time complexity is upper bounded by $\tilde{O}\left((m_1 m_2)^{1/3} n^{(\omega+1)/3}\right)$.*

Proof. The complexity of the algorithm of Proposition 1 is minimized for $t = n^{\mu}$, where μ is the solution of the equation $\mu + 2\omega(1, 1+\mu, 1) = 1 + \log_n(m_1 m_2)$. We can use items (ii) and (iii) of Fact 2 to obtain the upper bound $\omega(1, 1+\mu, 1) \leq \omega + \mu$, and optimize the complexity of the algorithm by taking $t = \left\lceil (m_1 m_2)^{1/3} n^{(1-2\omega)/3} \right\rceil$, which gives the upper bound claimed in the second part of the theorem. □

In the case of completely dense input matrices (i.e., $m_1 \approx n^2$ and $m_2 \approx n^2$), the second part of Theorem 3 shows that the complexity of the algorithm is $\tilde{O}(n^{(5+\omega)/3}) \leq O(n^{2.458})$.

4 Applications: (max, min)-Product, Distance Product

4.1 Quantum Algorithm for the (max, min)-Product

In this subsection we present a quantum algorithm for the matrix product $\lhd$, which immediately gives a quantum algorithm with the same complexity for the (max, min)-product as explained in Section 2, and then gives Theorem 1. Our algorithm first exploits the methodology by Vassilevska et al. [21] to reduce the computation of the product $\lhd$ to the computation of several sparse dominance products. The main technical difficulty to overcome is that, unlike in the classical case, computing all the sparse dominance products successively becomes too costly (i.e., the cost exceeds the complexity of all the other parts of the quantum algorithm). Instead, we show that it is sufficient to obtain a small fraction of the entries in each dominance product and that this task reduces to the computation of a generalized existence dominance product, and then use the quantum techniques of Proposition 1 to obtain precisely only those entries.

Theorem 4. *There exists a quantum algorithm that computes, for any two $n \times n$ matrices A and B with entries respectively in $\mathbb{Z} \cup \{\infty\}$ and $\mathbb{Z} \cup \{-\infty\}$, the product $A \lhd B$ with high probability in time $\tilde{O}(n^{(5-\gamma)/2})$, where γ is the solution of the equation $\gamma + 2\omega(1+\gamma, 1+\gamma, 1) = 5$. In particular, this complexity is upper bounded by $O(n^{2.473})$.*

Proof. Let $g \in \{1, \ldots, n\}$ be a parameter to be chosen later. For each $i \in \{1, \ldots, n\}$, we sort the entries in the i-th row of A in increasing order and divide the list into $s = \lceil n/g \rceil$ successive parts $R_1^i, \ldots, R_s^i$ with at most g entries in each part. For each $r \in \{1, \ldots, s\}$, define the $n \times n$ matrix A_r as follows: $A_r[i,j] = A[i,j]$ if $A[i,j] \in R_r^i$ and $A_r[i,j] = \infty$ otherwise. The cost of this (classical) preprocessing is $O(n^2 s)$ time.

We describe below the quantum algorithm that computes $C = A \lhd B$.

Step 1. For each $(i,j) \in \{1, \ldots, n\} \times \{1, \ldots, n\}$, we compute the largest $r \in \{1, \ldots, s\}$ such that $(A_r * B)[i,j] = 1$, if such an r exists. This is done by using the quantum algorithm of Proposition 1 with $u = s$, $v = 1$, $A^{(r)} = A_r$ for each $r \in \{1, \ldots, s\}$ and $B^{(1)} = B$. Note that $m_1 \leq s \times (ng) = O(n^2)$ and $m_2 \leq n^2$. The complexity of this step is thus

$$\tilde{O}\left(\frac{n^{5/2}}{\sqrt{t}} + n^{\omega(1+\log_n s, 1+\log_n t, 1)}\right)$$

for any parameter $t \in \{1, \ldots, n^2\}$. We want to minimize this expression. Let us write $t = n^\gamma$ and $g = n^\delta$. For a fixed δ, the first term is a decreasing function of γ, while the second term is an increasing function of γ. The expression is thus minimized for the value of γ solution of the equation

$$\omega(2-\delta, 1+\gamma, 1) = (5-\gamma)/2, \tag{5}$$

in which case the expression becomes $\tilde{O}(n^{(5-\gamma)/2})$.

Step 2. Note that at Step 1 we also obtain all $(i,j) \in \{1, \ldots, n\} \times \{1, \ldots, n\}$ such that no r satisfying $(A_r * B)[i,j] = 1$ exists. For all those (i,j), we set $C[i,j] = -\infty$. For all other (i,j), we will denote by r_{ij} the value found at Step 1. We now know that

$$C[i,j] = \max_{k:\, A[i,k] \in R^i_{r_{ij}}} \{A_{r_{ij}}[i,k] \mid A_{r_{ij}}[i,k] \leq B[k,j]\},$$

and $C[i,j]$ can be computed in time $\tilde{O}(\sqrt{g})$ using the quantum algorithm for maximum finding [7], since $|R^i_{r_{ij}}| \leq g$. The complexity of Step 2 is thus $\tilde{O}(n^2\sqrt{g})$.

This algorithm computes, with high probability, all the entries of $C = A \lhd B$. Its complexity is

$$\tilde{O}\left(n^2 s + n^{(5-\gamma)/2} + n^2\sqrt{g}\right) = \tilde{O}\left(n^{(5-\gamma)/2} + n^{2+\delta/2}\right),$$

since the term $n^2 s = n^{3-\delta}$ is negligible with respect to $n^{(5-\gamma)/2} = n^{\omega(2-\delta,1+\gamma,1)}$ by item (iv) of Fact 2. This expression is minimized for δ and γ satisfying $\delta + \gamma = 1$. Injecting this constraint into Equation (5), we find that the optimal value of γ is the solution of the equation $\gamma + 2\omega(1+\gamma, 1+\gamma, 1) = 5$, as claimed. Using items (i) and (ii) of Fact 2 and Fact 1, we obtain

$$\begin{aligned} 5 = \gamma + 2(1+\gamma)\omega\left(1, 1, \frac{1}{1+\gamma}\right) &\leq \gamma + 2(1+\gamma)\left(2 + \beta\left(\frac{1}{1+\gamma} - \alpha\right)\right) \\ &= (4 + 2\beta - 2\alpha\beta) + (5 - 2\alpha\beta)\gamma \end{aligned}$$

and then $\gamma \geq \frac{1+2\alpha\beta-2\beta}{5-2\alpha\beta}$. The complexity is thus $\tilde{O}\left(n^{(12-6\alpha\beta+\beta)/(5-2\alpha\beta)}\right) \leq O(n^{2.473})$. $\square$

4.2 Quantum Algorithm for the Distance Product

In this subsection we present a quantum algorithm that computes the most significant bits of the distance product of two matrices, as defined below.

Let A and B be two $n \times n$ matrices with entries in $\mathbb{Z} \cup \{\infty\}$. Let W be a power of two such that the value of each finite entry of their distance product C is upper bounded by W. For instance, one can take the smallest power of two larger than $\max_{i,j}\{A[i,j]\} + \max_{i,j}\{B[i,j]\}$, where the maxima are over the finite entries of the matrices. Each non-negative finite entry of C can then be expressed using $\log_2(W)$ bits: the entry $C[i,j]$ can be expressed as $C[i,j] = \sum_{k=1}^{\log_2(W)} C[i,j]_k \frac{W}{2^k}$ for bits $C[i,j]_1, \ldots, C[i,j]_{\log_2(W)}$. For any $\ell \in \{1, \ldots, \log_2(W)\}$, we say that an algorithm computes the ℓ most significant bits of each entry if, for all $(i,j) \in \{1, \ldots, n\} \times \{1, \ldots, n\}$ such that $C[i,j]$ is finite and non-negative, the algorithm outputs all the bits $C[i,j]_1, C[i,j]_2, \cdots, C[i,j]_\ell$. Vassilevska and Williams [20] have studied this problem, and shown how to reduce the computation of the ℓ most significant bits to the computation of $O(2^\ell)$ existence dominance matrix products of $n \times n$ matrices. By combining this with the $\tilde{O}(n^{(3+\omega)/2})$-time algorithm for dominance product from [17], they obtained a classical algorithm that computes the ℓ most significant bits of each entry of the distance product of A and B in time $\tilde{O}\left(2^\ell n^{(3+\omega)/2}\right) \leq \tilde{O}\left(2^\ell n^{2.687}\right)$.

Here is the main result of this subsection, whose proof is given in [15], obtained by reducing the computation of the ℓ most significant bits to computing a generalized existence dominance product.

Theorem 5. *There exists a quantum algorithm that computes, for any two $n \times n$ matrices A and B with entries in $\mathbb{Z} \cup \{\infty\}$, the ℓ most significant bits of each entry of the distance product of A and B in time $\tilde{O}\left(2^{0.640\ell} n^{(5+\omega)/3}\right) \leq O(2^{0.640\ell} n^{2.458})$ with high probability.*

Similarly, we can obtain a better classical algorithm as shown in the following theorem. We refer to [15] for details.

Theorem 6. *There exists a classical algorithm that computes, for any two $n \times n$ matrices A and B with entries in $\mathbb{Z} \cup \{\infty\}$, the ℓ most significant bits of each entry of the distance product of A and B in time $\tilde{O}\big(2^{0.960\ell} n^{(3+\omega)/2}\big) \leq O(2^{0.960\ell} n^{2.687})$.*

References

1. Amossen, R.R., Pagh, R.: Faster join-projects and sparse matrix multiplications. In: Proceedings of ICDT, pp. 121–126 (2009)
2. Boyer, M., Brassard, G., Høyer, P., Tapp, A.: Tight bounds on quantum searching. Fortschritte der Physik 46(4-5), 493–505 (1998)
3. Buhrman, H., Špalek, R.: Quantum verification of matrix products. In: Proceedings of SODA, pp. 880–889 (2006)
4. Bürgisser, P., Clausen, M., Shokrollahi, M.A.: Algebraic complexity theory. Springer (1997)
5. Duan, R., Pettie, S.: Fast algorithms for (max, min)-matrix multiplication and bottleneck shortest paths. In: Proceedings of SODA, pp. 384–391 (2009)
6. Dubois, D., Prade, H.: Fuzzy sets and systems: Theory and applications. Academic Press (1980)
7. Dürr, C., Høyer, P.: A quantum algorithm for finding the minimum. arXiv:quant-ph/9607014 (1996)
8. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of STOC, pp. 212–219 (1996)
9. Huang, X., Pan, V.Y.: Fast rectangular matrix multiplication and applications. Journal of Complexity 14(2), 257–299 (1998)
10. Jeffery, S., Kothari, R., Magniez, F.: Improving quantum query complexity of Boolean matrix multiplication using graph collision. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part I. LNCS, vol. 7391, pp. 522–532. Springer, Heidelberg (2012)
11. Le Gall, F.: Faster algorithms for rectangular matrix multiplication. In: Proceedings of FOCS, pp. 514–523 (2012)
12. Le Gall, F.: Improved output-sensitive quantum algorithms for Boolean matrix multiplication. In: Proceedings of SODA, pp. 1464–1476 (2012)
13. Le Gall, F.: A time-efficient output-sensitive quantum algorithm for Boolean matrix multiplication. In: Chao, K.-M., Hsu, T.-S., Lee, D.-T. (eds.) ISAAC 2012. LNCS, vol. 7676, pp. 639–648. Springer, Heidelberg (2012)
14. Le Gall, F.: Powers of tensors and fast matrix multiplication. In: Proceedings of ISSAC (to appear, 2014)
15. Le Gall, F., Nishimura, H.: Quantum algorithms for matrix products over semirings. Full version of the present paper, available as arXiv:1310.3898
16. Magniez, F., Santha, M., Szegedy, M.: Quantum algorithms for the triangle problem. SIAM Journal on Computing 37(2), 413–424 (2007)
17. Matoušek, J.: Computing dominances in E^n. Information Processing Letters 38(5), 277–278 (1991)
18. Shapira, A., Yuster, R., Zwick, U.: All-pairs bottleneck paths in vertex weighted graphs. In: Proceedings of SODA, pp. 978–985 (2007)
19. Vassilevska, V.: Efficient Algorithms for Path Problems in Weighted Graphs. PhD thesis, Carnegie Mellon University (2008)
20. Vassilevska, V., Williams, R.: Finding a maximum weight triangle in $n^{3-\delta}$ time, with applications. In: Proceedings of STOC, pp. 225–231 (2006)

21. Vassilevska, V., Williams, R., Yuster, R.: All pairs bottleneck paths and max-min matrix products in truly subcubic time. Theory of Computing 5(1), 173–189 (2009)
22. Vassilevska Williams, V.: Multiplying matrices faster than Coppersmith-Winograd. In: Proceedings of STOC, pp. 887–898 (2012)
23. Yuster, R.: Efficient algorithms on sets of permutations, dominance, and real-weighted APSP. In: Proceedings of SODA, pp. 950–957 (2009)
24. Yuster, R., Zwick, U.: Fast sparse matrix multiplication. ACM Transactions on Algorithms 1(1), 2–13 (2005)