CHICAGO JOURNAL OF THEORETICAL COMPUTER SCIENCE 2017, Article 1, pages 1–25
http://cjtcs.cs.uchicago.edu/

Quantum Algorithms for Matrix Products over Semirings

François Le Gall   Harumichi Nishimura

Received July 24, 2015; Revised May 15, 2016; Published May 11, 2017

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, by Duan and Pettie (SODA'09), has complexity $O(n^{2.687})$.
- 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, the best known classical algorithm for the same problem, by Vassilevska and Williams (STOC'06) and Yuster (SODA'09), has complexity $O(2^{\ell}n^{2.69})$.

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.

Key words and phrases: quantum algorithms, matrix multiplication, semirings

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., [6, 19, 21, 22, 23, 25]). One example is Boolean

DOI: 10.4086/cjtcs.2017.001

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 $k \in \{1,\dots,n\}$ 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 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 $(i,j) \in \{1,\dots,n\} \times \{1,\dots,n\}$. 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., [21]), 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 [9] or for $\oplus = \min$ and $\oplus = \max$ using quantum algorithms for minimum finding [8]) and each operation $\odot$ can be implemented in $\mathrm{polylog}(n)$ 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 [6], 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 (without introducing assumptions on the matrices).

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 the following result for matrix multiplication over the $(\max,\min)$ semiring.

Theorem 1.1. *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 [6], 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 [7] 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., [6, 19, 23]) 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 1.1, we thus obtain a $O(n^{2.473})$-time quantum algorithm computing the all-pairs bottleneck

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

paths of a graph of n vertices, while classically the best upper bound for this task is $O(n^{2.687})$, again from [6].

In order to prove Theorem 1.1, we construct a quantum algorithm that computes the product of two $n \times n$ matrices over the existence dominance semiring (defined in the next section) in time $\tilde{O}(n^{(5+\omega)/3}) \leq O(n^{2.458})$. The dominance product has applications in computational geometry [18] and graph algorithms [22] and, in comparison, the best known classical algorithm for this product [25] 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 develop both quantum and classical algorithms that compute efficiently this product. This is the most technical part of this paper.

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 [22] had 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 [25]. 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, 11, 13, 14], 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 evaluate the performance of quantum algorithms in this scenario. Our result (Theorem 5.1) shows how several standard combinatorial ideas for sparse Boolean matrix multiplication can be adapted in the quantum setting, and used to construct quantum algorithms faster than the best known classical algorithms. In particular, we obtain the following result.

Theorem 1.2 (simplified version – the complete version appears as Theorem 5.2 in Section 5). *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*

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

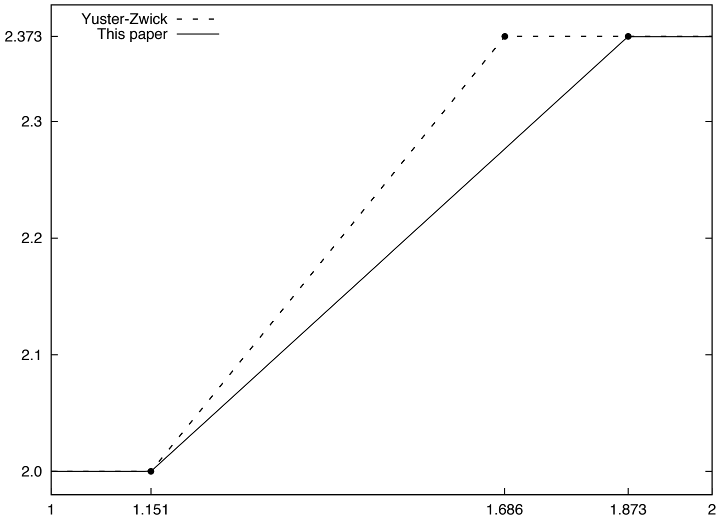

Figure 1: The upper bounds of Theorem 1.2 (in solid lines). The horizontal axis represents the logarithm of m with respect to basis n (i.e., the value $\log_n(m)$). The vertical axis represents the logarithm of the complexity with respect to basis n. The dashed lines represent the upper bounds obtained by [26].

The complexity of the algorithm of Theorem 1.2 is the piece-linear function of $\log_n(m)$ represented in Figure 1. In comparison, the best known classical algorithm, by Yuster and Zwick [26], has complexity $\tilde{O}(n^2)$ if $m \le n^{1.151}$, $\tilde{O}(m^{0.697}n^{1.199})$ if $n^{1.151} \le m \le n^{(1+\omega)/2}$, and $\tilde{O}(n^{\omega})$ if $n^{(1+\omega)/2} \le m \le n^2$. Our algorithm performs better when $n^{1.151} < m < n^{\omega-1/2}$. For instance, if $m = O(n^{(1+\omega)/2}) = O(n^{1.686\ldots})$, then our algorithm has complexity $O(n^{2.277})$, while the algorithm by [26] has complexity $\tilde{O}(n^{\omega})$.

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, 6, 23, 26] 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. More detailed technical overviews are given at the beginning of Sections 3, 4 and 5.

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 [5, 15, 20, 24]. 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 [12]. The following facts are known, and will be used in this paper. We refer to [4, 10] for details.

Fact 2.1. *$\omega(1,k,1) = 2$ for $k \leq \alpha$ and $\omega(1,k,1) \leq 2 + \beta(k - \alpha)$ for $\alpha \leq k \leq 1$.*

Fact 2.2. *The following relations hold for any values $k_1, k_2, k_3 > 0$:*

(i) $\omega(kk_1, kk_2, kk_3) = k\omega(k_1, k_2, k_3)$ *for any $k > 0$;*

(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 [6, 22, 23].

Definition 2.3. 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.

As mentioned for instance in [6, 23], 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 [21], 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, 11, 13, 14] for Boolean matrix multiplication. More precisely, let A and B be two $n \times n$ matrices, for any positive integer n (the model presented below can be generalized to deal with rectangular matrices in a straightforward way). We suppose that these matrices can be accessed directly by a quantum algorithm: We have an oracle O_A that, for any $i, j \in \{1, \ldots, n\}$, and any $z \in \{0,1\}^*$, maps the state $|i\rangle|j\rangle|0\rangle|z\rangle$ to $|i\rangle|j\rangle|A[i,j]\rangle|z\rangle$. We have a similar oracle O_B for B. Since we are interested in time complexity, we will count all the computational steps of the algorithm and assign a cost of one for each call to O_A or O_B, which corresponds to the cases where quantum access to the inputs A and B can be done at unit cost, for example in a random access model working in quantum superposition. We say that a quantum algorithm for matrix multiplication computes the product of A and B with high probability if, when given access to oracles O_A and O_B corresponding to A and B, the algorithm outputs with probability at least $2/3$ all the entries of the product of A and B. The complexity of several algorithms in this paper will be stated using an upper bound λ on the number of non-zero or non-infinite entries in the product of A and B. The same complexity, up to a logarithmic factor, can actually be obtained even if no nontrivial upper bound is known a priori, see [13, 14].

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 [17]) 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 $\tilde{O}\big(\sqrt{N/t}+\sqrt{N/(t-1)}+\cdots+\sqrt{N/1}\big) = \tilde{O}(\sqrt{N(t+1)})$ 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 [6]: 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 [6] 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 3.1. 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:

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

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 3.2. *Let $A^{(1)},\dots,A^{(u)}$ be u matrices of size $n \times n$ with entries in $\mathbb{Z} \cup \{\infty\}$, and $B^{(1)},\dots,B^{(v)}$ be v matrices of size $n \times n$ with entries in $\mathbb{Z} \cup \{-\infty\}$. Let $m_1 \in \{1,\dots,n^2u\}$ denote the total number of finite entries in the matrices $A^{(1)},\dots,A^{(u)}$, and $m_2 \in \{1,\dots,n^2v\}$ denote the total number of finite entries in the matrices $B^{(1)},\dots,B^{(v)}$. For any parameter $t \in \{1,\dots,m_1\}$, 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 L be the list of all finite entries in $A^{(1)},\dots,A^{(u)}$ sorted in increasing order (if the finite entries of these matrices are not distinct then some elements will appear more than once in the list). Decompose L into t successive parts $L_1,\dots,L_t$, each containing at most $\lceil m_1/t \rceil$ entries. For each $x \in \{1,\dots,u\}$ and each $r \in \{1,\dots,t\}$ we construct two $n \times n$ matrices $A_r^{(x)}, \bar{A}_r^{(x)}$ as follows: for all $(i,j) \in \{1,\dots,n\} \times \{1,\dots,n\}$,

$$A_r^{(x)}[i,j] = \begin{cases} A^{(x)}[i,j] & \text{if } A^{(x)}[i,j] \in L_r, \\ \infty & \text{otherwise,} \end{cases}$$

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

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

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

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

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

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

$$A^{(x)} * B^{(y)} = \sum_{r=1}^{t} \left(\bar{A}_r^{(x)} \cdot \bar{B}_r^{(y)}\right) + \sum_{r=1}^{t} \left(A_r^{(x)} * B_r^{(y)}\right). \tag{3.1}$$

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 $(i,j) \in \{1,\dots,n\} \times \{1,\dots,n\}$,

$$C_1[i,j] = \max\left\{\{(0,0)\} \cup \{(x,y) \in S \mid \sum_{r=1}^{t} \bar{A}_r^{(x)} \cdot \bar{B}_r^{(y)}[i,j] = 1\}\right\}, \tag{3.2}$$

$$C_2[i,j] = \max\left\{\{(0,0)\} \cup \{(x,y) \in S \mid \sum_{r=1}^{t} A_r^{(x)} * B_r^{(y)}[i,j] = 1\}\right\}. \tag{3.3}$$

From Equation (3.1), 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 $O(n^2uv)$ 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 $\tilde{O}(n^2t(u+v))$).

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

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 2.2, we conclude that the matrix C_1 can be computed in time

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

We now explain how to compute the matrix C_2. Intuitively, the main difficulty is that Equation (3.3) cannot be used directly since we do not know how to compute the dominance product $*$ efficiently. Lemma 3.3 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 (3.3) with Equation (3.4) 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 (3.3) by the term $\{D[i,j]\}$ in Equation (3.4), 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, for readability its proof is postponed until Section 6.

Lemma 3.3. *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

$$C_2[i,j] = \max\left\{\{D[i,j]\} \cup \{(x,y) \in S \mid \sum_{r=1}^{t} \hat{A}_r^{(x)} \cdot \hat{B}_r^{(y)}[i,j] = 1\}\right\} \tag{3.4}$$

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

$$\tilde{O}\left(n^2t(u+v) + \sqrt{\frac{m_1m_2n}{t}} + \sqrt{\frac{m_1m_2uv}{tn}}\right).$$

After applying the quantum algorithm of Lemma 3.3, 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 2.2. □

We can give a classical version of this result, whose proof can be found in Section 6, that will be used to prove Theorem 4.3 in Section 4.2.

Proposition 3.4. *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 3.2, we obtain the following theorem.

Theorem 3.5. *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 3.2 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 2.2 to obtain the upper bound $\omega(1, 1+\mu, 1) \leq \omega + \mu$, and optimize the complexity of the algorithm by taking $t = \lceil (m_1 m_2)^{1/3} n^{(1-2\omega)/3} \rceil$, 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 3.5 shows that the complexity of the algorithm is $\tilde{O}(n^{(5+\omega)/3}) \leq O(n^{2.458})$.

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

In this section we show how to apply the results of Section 3 to construct quantum algorithms for the $(\max, \min)$-product and the 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 1.1. Our algorithm first exploits the methodology by Vassilevska et al. [23] 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 3.2 to obtain precisely only those entries.

Theorem 4.1. *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 $R^i_1,\ldots,R^i_s$ 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 $A[i,j]\in R^i_r$ and $A_r[i,j]=\infty$ otherwise. The cost of this (classical) preprocessing is $O(n^2s)$ 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 3.2 with $u=s$, $v=1$, $A^{(r)}=A_r$ for each $r\in\{1,\ldots,s\}$ and $B^{(1)}=B$. Note that $m_1\le s\times(ng)=O(n^2)$ and $m_2\le n^2$. 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

$$\omega(2-\delta,1+\gamma,1)=(5-\gamma)/2, \tag{4.1}$$

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

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

and $C[i,j]$ can be computed in time $\tilde{O}(\sqrt{g})$ using the quantum algorithm for maximum finding [8], since $|R^i_{r_{ij}}|\le g$. 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 2.2. This expression is minimized for δ and γ satisfying $\delta + \gamma = 1$. Injecting this constraint into Equation (4.1), 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 Fact 2.1 and items (i) and (ii) of Fact 2.2, 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 $\tilde{O}\left(n^{(12-6\alpha\beta+\beta)/(5-2\alpha\beta)}\right) \leq O(n^{2.473})$, as claimed. □

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 [22] 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 [18], 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, obtained by reducing the computation of the ℓ most significant bits to computing a generalized existence dominance product.

Theorem 4.2. *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.*

Proof. Note that the trivial $\tilde{O}(n^{5/2})$-time quantum algorithm can be used to compute all the bits of each entry of the distance product C of A and B. Therefore, we will assume, without loss of generality, that ℓ satisfies the inequality $2^{0.640\ell} n^{(5+\omega)/3} \leq n^{5/2}$, which implies in particular that $2^\ell \leq n^2$.

Assume first that all the entries of C are finite and non-negative. What we want to do is to compute, for each $(i,j) \in \{1,\ldots,n\} \times \{1,\ldots,n\}$, the integer $d \in \{0, 1 \ldots, 2^\ell - 1\}$ such that $C[i,j]$ is in the interval $[dW/2^\ell, (d+1)W/2^\ell)$.

For any integer x, define the matrices A'_x and B'_x as follows: for all $(i,j) \in \{1,\dots,n\} \times \{1,\dots,n\}$,

$$A'_x[i,j] = A[i,j] - \frac{xW}{2^{\ell/2}}, \qquad\qquad B'_x[i,j] = -B[i,j] + \frac{xW}{2^{\ell}}.$$

Assume for simplicity that ℓ is even (a similar argument works for ℓ odd). For each $d \in \{0,1\dots,2^{\ell}-1\}$, let $d_1,d_2 \in \{0,1\dots,2^{\ell/2}-1\}$ denote the values such that $d = d_1 2^{\ell/2} + d_2$. For each $d \in \{0,1\dots,2^{\ell}-1\}$, define the Boolean matrix $D_d = A'_{d_1} * B'_{d_2}$, where $*$ means the strict[2] existence dominance product. Note that $\frac{d_1 W}{2^{\ell/2}} + \frac{d_2 W}{2^{\ell}} = \frac{dW}{2^{\ell}}$. Observe that, for each $(i,j) \in \{1,\dots,n\} \times \{1,\dots,n\}$, we have

$$D_d[i,j] = 0 \Longleftrightarrow \min_k (A[i,k] + B[k,j]) \geq \frac{dW}{2^{\ell}}.$$

For each $(i,j) \in \{1,\dots,n\} \times \{1,\dots,n\}$, the integer $d \in \{1,\dots,2^{\ell}\}$ such that $C[i,j]$ is in the interval $[(d-1)W/2^{\ell}, dW/2^{\ell})$ can thus be found by computing the smallest $d \in \{0,1\dots,2^{\ell}-1\}$ such that $D_d[i,j] = 1$. We can thus use[3] the quantum algorithm of Proposition 3.2, with $u = v = 2^{\ell/2}$, $A^{(x)} = A'_{x-1}$ and $B^{(y)} = B'_{y-1}$ for each $x,y \in \{1,\dots,2^{\ell/2}\}$. Since $m_1 \leq 2^{\ell/2}n^2$, $m_2 \leq 2^{\ell/2}n^2$ and from the inequality $2^{\ell} \leq n^2$ on ℓ, the complexity is

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

for any parameter $t \in \{1,\dots,2^{\ell/2}n^2\}$. Let us write $\mu = \log_n(2^{\ell})$ and $t = n^{\gamma}$. The complexity is minimized for the value γ such that $5+\mu-\gamma = 2\omega(1+\mu/2,1+\gamma,1+\mu/2)$, for which the complexity is $\tilde{O}\left(n^{(5+\mu-\gamma)/2}\right)$. Using Fact 2.1 and items (i) and (iii) of Fact 2.2, we obtain

$$\begin{aligned}\omega(1+\mu/2,1+\gamma,1+\mu/2) &\leq \gamma + (1+\mu/2)\,\omega\left(1,\frac{1}{1+\mu/2},1\right)\\ &\leq \gamma + (1+\mu/2)\left(2+\beta\left(\frac{2}{2+\mu}-\alpha\right)\right).\end{aligned}$$

This gives $5+\mu-\gamma \leq 2\gamma + (2-\alpha\beta)\mu + (4+2\beta-2\alpha\beta)$ and thus

$$\gamma \geq \frac{(\alpha\beta-1)\mu + (1-2\beta+2\alpha\beta)}{3}.$$

The complexity is thus

$$\tilde{O}\left(n^{\frac{5}{2}+\frac{2\beta-2\alpha\beta-1}{6}+\frac{(4-\alpha\beta)}{6}\mu}\right) = \tilde{O}\left(n^{\frac{5+\omega}{3}+0.640\mu}\right) = O\left(2^{0.640\ell}n^{2.458}\right).$$

[2] The strict existence dominance product is obtained by replacing $\leq$ by $<$ in the definition of the existence dominance product (Definition 3.1). Note that all our results on the existence dominance product also hold for the strict existence dominance product and their proofs are essentially the same, just replacing inequalities by strict inequalities.

[3] Actually, we need to modify the order $\prec$ in Definition 3.1 so that the algorithm of Proposition 3.2 finds the smallest d such that $D_d[i,j] = 1$ instead of the largest d. This is done simply by choosing $\prec$ as the decreasing lexicographic order instead of the usual lexicographic order. Proposition 3.2 and its proof are unchanged, since the proof only uses the fact that $\prec$ is a strict total order.

Finally, we discuss the general case where the entries of C can be negative or infinite. Observe that the above algorithm detects which entries of C are larger than $(2^\ell-1)W/2^\ell$: these are the entries such that the algorithm finds no d such that $D_d[i,j]=1$. We can find which of these entries are larger than W (and thus infinite) by computing the dominance product $A'_0 * B'_{2^\ell}$. Note that the algorithm also finds which entries of C are negative: these are the entries for which the smallest d such that $D_d[i,j]=1$ is $d=0$. □

Similarly, we can obtain a better classical algorithm as shown in the following theorem.

Theorem 4.3. *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 $\tilde{O}(2^{0.960\ell}n^{(3+\omega)/2})\leq O(2^{0.960\ell}n^{2.687})$.*

Proof. The proof is similar to the proof of Theorem 4.2, but we use Proposition 3.4 instead of Proposition 3.2. The complexity becomes

$$\tilde{O}\left(\frac{2^\ell n^3}{t}+n^{\omega(1+\log_n(2^{\ell/2}),1+\log_n t,1+\log_n(2^{\ell/2}))}\right)$$

for any parameter $t\in\{1,\ldots,2^{\ell/2}n^2\}$. Let us write $\mu=\log_n(2^\ell)$ and $t=n^\gamma$. This expression is then

$$O\left(n^{3+\mu-\gamma}+n^{\omega(1+\mu/2,1+\gamma,1+\mu/2)}\right).$$

This expression is minimized for the value γ such that

$$3+\mu-\gamma=\omega(1+\mu/2,1+\gamma,1+\mu/2),$$

for which the complexity is $\tilde{O}\left(n^{3+\mu-\gamma}\right)$.

Using Fact 2.1 and items (i) and (iii) of Fact 2.2, we obtain

$$\begin{aligned}\omega(1+\mu/2,1+\gamma,1+\mu/2) &\leq \gamma+(1+\mu/2)\omega\left(1,\frac{1}{1+\mu/2},1\right)\\ &\leq \gamma+(1+\mu/2)\left(2+\beta\left(\frac{2}{2+\mu}-\alpha\right)\right).\end{aligned}$$

This gives the inequality

$$3+\mu-\gamma\leq\gamma+(1-\frac{\alpha\beta}{2})\mu+(2+\beta-\alpha\beta),$$

from which we obtain

$$\gamma\geq\frac{\alpha\beta\mu/2+(1-\beta+\alpha\beta)}{2}.$$

The complexity is thus

$$\tilde{O}\left(n^{\frac{(5+\beta-\alpha\beta)}{2}+(1-\frac{\alpha\beta}{4})\mu}\right)=\tilde{O}\left(n^{\frac{3+\omega}{2}+0.960\mu}\right)=O\left(2^{0.960\ell}n^{2.687}\right),$$

as claimed. □

Note that the dependency on n of the $\tilde{O}\left(2^{\ell}n^{2.687}\right)$-time algorithm by Vassilevska and Williams [22] can be slightly improved using the recent $O(n^{2.684})$-time algorithm for dominance product by Yuster [25] based on rectangular matrix multiplication. We can similarly obtain an improved bound $O(2^{c\ell}n^{2.684})$, for some $c < 1$, with the same approach as in the proof of Theorem 4.3. However, it is complicated to express the value of c in a closed form, so we omit the statement of this slight improvement.

5 Sparse Boolean Matrix Multiplication

In this section we describe quantum versions of several known combinatorial techniques for handling sparse Boolean matrix products. The main result is the following theorem, which shows how to compute the Boolean product of two matrices A and B by reducing it to four products, each easier to compute than the original one when A and B are sparse enough. Note that similar ideas have been used in [1, 26] to analyze applications of those combinatorial techniques in the classical setting. Here we show how to implement these ideas using quantum enumeration and analyze the complexity of the resulting algorithm.

Theorem 5.1. *Assume that there exists an algorithm that, in time $M(n_1,n_2,n_3,L)$, computes the product of any $n_1 \times n_2$ Boolean matrix and any $n_2 \times n_3$ Boolean matrix such that their product contains at most L non-zero entries. Let A and B be two $n \times n$ Boolean matrices with at most m_1 and m_2 non-zero entries in A and B, respectively. For any values of the three parameters $\ell_1 \in \{1,\ldots,m_1\}$ and $\ell_2,\ell_3 \in \{1,\ldots,m_2\}$, there exists a quantum algorithm that computes, with high probability, the Boolean product $A \cdot B$ and has time complexity*

$$\tilde{O}\left(M(\ell_1',\ell_2',\ell_3',\lambda)+\sqrt{\frac{m_1m_2\cdot\min(\lambda,m_1m_2/\ell_2)}{\ell_2}}+\lambda\sqrt{\frac{m_1}{\ell_1}}+\lambda\sqrt{\frac{m_2}{\ell_3}}+n^2\right),$$

where λ denotes the number of non-zero entries in $A \cdot B$, and $\ell_i' = \min(\ell_i,n)$ for each $i \in \{1,2,3\}$.

Proof. For any $k \in \{1,\ldots,n\}$, let a_k^R (resp. b_k^R) be the number of non-zero entries in the k-th row of A (resp. B) and a_k^C (resp. b_k^C) be the number of non-zero entries in the k-th column of A (resp. B). We define the following six sets of indexes, and compute them classically in time $O(n^2)$.

$$\begin{aligned} S &= \{k \in \{1,\ldots,n\} \mid b_k^R \geq m_2/\ell_2\} & S' &= \{k \in \{1,\ldots,n\} \mid b_k^R < m_2/\ell_2\} \\ T &= \{k \in \{1,\ldots,n\} \mid a_k^R \geq m_1/\ell_1\} & T' &= \{k \in \{1,\ldots,n\} \mid a_k^R < m_1/\ell_1\} \\ U &= \{k \in \{1,\ldots,n\} \mid b_k^C \geq m_2/\ell_3\} & U' &= \{k \in \{1,\ldots,n\} \mid b_k^C < m_2/\ell_3\} \end{aligned}$$

Given two sets $R,C \subseteq \{1,\ldots,n\}$ and an $n \times n$ Boolean matrix M, the notation M_R^C will represent the $n \times n$ Boolean matrix such that $M_R^C[i,j] = 1$ if and only if $M[i,j] = 1$ and $(i,j) \in R \times C$. For convenience, M_R will represent the matrix M_R^C for $C = \{1,\ldots,n\}$, and M^C the matrix M_R^C for $R = \{1,\ldots,n\}$.

It is easy to check that

$$A \cdot B = A_T^S \cdot B_S^U + A^{S'} \cdot B_{S'} + A_{T'} \cdot B + A \cdot B^{U'},$$

where $+$ represents the entry-wise OR operation. We will individually compute the four terms of this sum.

The computation of $A^S_T \cdot B^U_S$ consists in the computation of a $|T| \times |S|$ matrix by a $|S| \times |U|$ matrix. We implement this part using the algorithm whose existence is assumed in the statement of the theorem. Note that, from the sparsity of B, we have $m_2 \geq \sum_{k=1}^n b^R_k \geq \sum_{k\in S} b^R_k \geq |S| m_2/\ell_2$, and thus $|S| \leq \ell_2$. Similarly we have $|T| \leq \ell_1$ and $|U| \leq \ell_3$. Additionally, we know that S, T and U have size at most n. Thus this part can be implemented in $M(\ell_1', \ell_2', \ell_3', \lambda)$ time.

In order to compute $A^{S'} \cdot B_{S'}$ we do the following. Let m_1' denote the number of non-zero entries of $A^{S'}$. First, we list all these non-zero entries, classically in time $O(n^2)$, and record them into two arrays M_1 and M_2 of size m_1': for each $p \in \{1, \ldots, m_1'\}$ the value $\mathsf{M}_1[p]$ records the row index of the p-th element of the list, while $\mathsf{M}_2[p]$ records its column index. Then, for each $k \in S'$, we compute the set of indexes $j \in \{1, \ldots, n\}$ such that $B[k, j] = 1$ and record them into an array N_k. Note that N_k has length b^R_k, and that $b^R_k < m_2/\ell_2$ from the definition of S'. The computation of all the N_k's can be done classically in $O(n^2)$ time. Finally, take $N = \sum_{c=1}^{m_1'} b^R_{\mathsf{M}_2[c]}$ and define the function $g\colon \{1, \ldots, N\} \to \{1, \ldots, n\} \times \{1, \ldots, n\}$ as follows: for any $p \in \{1, \ldots, m_1'\}$ and any $q \in \{1, \ldots, b^R_{\mathsf{M}_2[p]}\}$,

$$g\left(q + \sum_{c=1}^{p-1} b^R_{\mathsf{M}_2[c]}\right) = (\mathsf{M}_1[p], \mathsf{N}_{\mathsf{M}_2[p]}[q]),$$

where $\mathsf{N}_{\mathsf{M}_2[p]}[q]$ denotes the q-th element of the array $\mathsf{N}_{\mathsf{M}_2[p]}$. It is easy to check that

$$g(\{1, \ldots, N\}) = \Big\{ (i, j) \in \{1, \ldots, n\} \times \{1, \ldots, n\} \,|\, \text{there exists } k \in S' \text{ such that } A[i, k] = B[k, j] = 1 \Big\},$$

i.e., $g(\{1, \ldots, N\})$ is precisely the set of non-zero entries of $A^{S'} \cdot B_{S'}$ that we want to find. A crucial point here is that the function g can be evaluated in $\mathrm{poly}(\log n)$ time using the data structures M_1, M_2 and N_k. For any subset Σ of $\{1, \ldots, n\} \times \{1, \ldots, n\}$, let $f_\Sigma\colon \{1, \ldots, N\} \to \{0, 1\}$ be the function such that $f_\Sigma(x) = 1$ if and only if $g(x) \notin \Sigma$. The quantum procedure starts with Σ being empty, performs successive quantum searches over $\{1, \ldots, N\}$, each time searching for an element x such that $f_\Sigma(x) = 1$ and adding $g(x)$ to Σ as soon as such an x is found, and stops when no new element x is found. From the discussion of Section 2, with high probability all searches succeed, in which case at the end of the procedure $\Sigma = g(\{1, \ldots, N\})$. Let λ' denote the number of non-zero entries in $A^{S'} \cdot B_{S'}$ and observe that $\lambda' \leq \min(\lambda, m_1 m_2/\ell_2)$, since $N < m_1' m_2/\ell_2 \leq m_1 m_2/\ell_2$. The overall complexity of this quantum procedure is

$$\tilde{O}\left(n^2 + \sqrt{N \times (\lambda' + 1)}\right) = \tilde{O}\left(n^2 + \sqrt{\frac{m_1 m_2 \cdot \min(\lambda, m_1 m_2/\ell_2)}{\ell_2}}\right).$$

The computation of $A_{T'} \cdot B$ is done as follows. For each $k \in \{1, \ldots, n\}$, let a_k' denote the number of non-zero entries in the k-th column of $A_{T'}$. We first perform a $O(n^2)$-time classical preprocessing step: for each $k \in \{1, \ldots, n\}$, we construct the set E_k of the row indexes of all non-zero entries in the k-th column of $A_{T'}$, and construct the set F_k of the column indexes of all non-zero entries in the k-th row of B. Note that $|E_k| = a_k'$ and $|F_k| = b^R_k$. The quantum procedure computing $A_{T'} \cdot B$ uses a set $\Sigma \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$, initially empty. For each $k \in \{1, \ldots, n\}$, all the $(i, j) \in E_k \times F_k$ such that

$A_{T'}[i,k] = B[k,j] = 1$ and $(i,j) \notin \Sigma$ are computed by performing a quantum enumeration, as above, over the set $E_k \times F_k$, adding (i,j) to Σ as soon as such a (i,j) is found, and stopping when no new element (i,j) is found. The overall time complexity is $\tilde{O}\left(n^2 + \sum_{k=1}^{n} \sqrt{a'_k b^R_k (\lambda_k + 1)}\right)$, where λ_k is the number of elements found when processing k. Note that the inequality $\sum_k a'_k b^R_k < \lambda \min(m_1/\ell_1, n)$ holds, since $\sum_k a'_k b^R_k$ also represents the total number of witnesses of $A_{T'} \cdot B$, i.e., the number of triples (i,j,k) such that $A_{T'}[i,k] = B[k,j] = 1$ (observe that there are at most λ pairs (i,j) satisfying this condition, all such that $i \in T'$). Since $\sum_k \lambda_k \leq \lambda \leq n^2$, this complexity is upper bounded by

$$\begin{aligned}\tilde{O}\left(n^2 + \sqrt{\lambda + n} \times \sqrt{\sum_{k=1}^{n} a'_k b^R_k}\right) &= \tilde{O}\left(n^2 + \sqrt{\lambda + n} \times \sqrt{\lambda \min\left(\frac{m_1}{\ell_1}, n\right)}\right) \\ &= \tilde{O}\left(n^2 + \lambda \times \sqrt{\frac{m_1}{\ell_1}}\right).\end{aligned}$$

Computing $A \cdot B^{U'}$ is done similarly to the computation of the product $A_{T'} \cdot B$ with complexity $\tilde{O}\left(n^2 + \lambda\sqrt{m_2/\ell_3}\right)$. □

We now compare the results of Theorem 5.1 to previous works. For the case $m_1 = m_2 \approx n^2$, the bounds obtained in Theorem 5.1 are not better than the best known output-sensitive algorithms for Boolean matrix multiplication [11, 14, 16]. Interestingly, we nevertheless recover the same complexity $O(\lambda\sqrt{n})$ as in [14] for the region $n^{3/2} \leq \lambda \leq n^2$, but using different methods (this is done by taking $\ell_1 = m_1/(n+1)$, which gives $T' = \{1, \ldots, n\}$ and reduces the computation of $A \cdot B$ to the computation of only $A_{T'} \cdot B$). Consider now sparse input matrices and, for concreteness, focus on the case $m_1 = m_2$ (we denote this value simply by m). The complexity of the algorithm by Amossen and Pagh [1], while not stated in this form, can be written as

$$\tilde{O}\left(M(\ell'_1, \ell'_2, \ell'_1, \lambda) + m^2/\ell_2 + \lambda m/\ell_1 + n^2\right)$$

using the notations of Theorem 5.1. In comparison, Theorem 5.1 gives (by choosing $\ell_1 = \ell_3$) the upper bound

$$\tilde{O}\left(M(\ell'_1, \ell'_2, \ell'_1, \lambda) + \min(m\sqrt{\lambda/\ell_2}, m^2/\ell_2) + \lambda\sqrt{m/\ell_1} + n^2\right).$$

We see that the second and third terms in our complexity are never worse. In order to evaluate quantitatively the speedup obtained in the quantum setting, let us consider the case when only the input matrices are sparse (i.e., $\lambda \approx n^2$). In this case, the algorithm by Amossen and Pagh has the same complexity as the algorithm by Yuster and Zwick [26] described in the introduction. In comparison, Theorem 5.1 gives the following result, which shows that our quantum algorithm is better than their classical algorithm, as discussed in the introduction.

Theorem 5.2 (complete version of Theorem 1.2)**.** *Let A and B be two $n \times n$ Boolean matrices with at most m_1 and m_2 non-zero entries in A and B, respectively. There exists a quantum algorithm that computes,* *with high probability, the Boolean matrix product $A\cdot B$ and has time complexity*

$$\begin{cases} \tilde{O}(n\times\min(m_1,m_2)) & \text{if } 1\le\sqrt{m_1m_2}\le n,\\ \tilde{O}(n^2) & \text{if } n\le\sqrt{m_1m_2}\le n^{1+\alpha/2},\\ \tilde{O}\left((m_1m_2)^{\frac{\beta}{1+2\beta}}n^{\frac{2+2\beta-\alpha\beta}{1+2\beta}}\right) & \text{if } n^{1+\alpha/2}\le\sqrt{m_1m_2}\le n^{\omega-1/2},\\ \tilde{O}(n^{\omega}) & \text{if } n^{\omega-1/2}\le\sqrt{m_1m_2}\le n^2. \end{cases}$$

Proof. First consider the case $\sqrt{m_1m_2}\le n$. Assume for now that $m_1\le m_2$. We use the following strategy: we first use quantum enumeration to find all the non-zero entries of A and, then, for each such entry $A[i,k]$, we output all the j's such that $B[k,j]=1$. The complexity of this strategy is $\tilde{O}(\sqrt{(m_1+1)n^2}+m_1n)=\tilde{O}(m_1n)$. The same argument gives the upper bound $\tilde{O}(m_2n)$ when $m_2\le m_1$.

If $n\le\sqrt{m_1m_2}\le n^{1+\alpha/2}$, then we use the quantum algorithm of Theorem 5.1 with parameters $\ell_1=m_1$, $\ell_2=m_1m_2/n^2$, $\ell_3=m_2$, and applying the algorithm for rectangular matrix multiplication over a field described in Section 2 for the part $M(\ell_1',\ell_2',\ell_3',n^2)$. This gives overall complexity $\tilde{O}(n^2)$ time.

If $n^{1+\alpha/2}\le\sqrt{m_1m_2}\le n^{\omega-1/2}$, then we use the quantum algorithm of Theorem 5.1 with parameters $\ell_1=m_1$, $\ell_3=m_2$ and

$$\ell_2=(m_1m_2)^{\frac{1}{1+2\beta}}n^{\frac{2(\alpha\beta-1)}{1+2\beta}},$$

giving overall complexity

$$\tilde{O}\left((m_1m_2)^{\frac{\beta}{1+2\beta}}n^{\frac{2+2\beta-\alpha\beta}{1+2\beta}}\right).$$

Finally, if $\sqrt{m_1m_2}\ge n^{\omega-1/2}$, then we simply use the best existing classical algorithm for dense matrix multiplication. □

6 Proofs of Lemma 3.3 and Proposition 3.4

In this section we give the proofs of Lemma 3.3 and Proposition 3.4.

6.1 Proof of Lemma 3.3

We will use the notation $\mathrm{col}(M,k)$ to denote the number of finite entries in the k-th row of M, for any $n_1\times n_2$ matrix M with entries in $\mathbb{Z}\cup\{\pm\infty\}$ and any $k\in\{1,\dots,n_2\}$.

Our algorithm proceeds in several steps.

Preprocessing: column balancing

For each $r\in\{1,\dots,t\}$, we do the following. Consider the u matrices $A_r^{(1)},\dots,A_r^{(u)}$. Each matrix has size $n\times n$ and we know that the total number of finite entries in these u matrices is at most $\lceil m_1/t\rceil$:

$$\sum_{x=1}^{u}\sum_{k=1}^{n}\mathrm{col}(A_r^{(x)},k)\le\lceil m_1/t\rceil\,. \tag{6.1}$$

We will construct u matrices $\tilde{A}_r^{(1)},\dots,\tilde{A}_r^{(u)}$, each of size $n\times 2n$. Each $\tilde{A}_r^{(x)}$ will contain all the finite entries in $A_r^{(x)}$, but these u matrices will satisfy the following sparsity condition on each column:

$$\sum_{x=1}^{u}\mathrm{col}(\tilde{A}_r^{(x)},k')\leq\lceil m_1/(nt)\rceil\quad\text{for all } k'\in\{1,\dots,2n\}. \tag{6.2}$$

These matrices are related to the concept of column balancing developed in [6].

Let us describe how to construct these matrices $\tilde{A}_r^{(1)},\dots,\tilde{A}_r^{(u)}$. For each index $k\in\{1,\dots,n\}$, we first collect together all the finite entries in the k-th column of $A_r^{(1)},\dots,A_r^{(u)}$ and sort them in increasing order. This gives, for each k, a sorted list of at most nu numbers, with possible repetitions. We then divide this list into successive parts $T_{r,k}^1,T_{r,k}^2,\dots,T_{r,k}^{a_{r,k}}$, for some $a_{r,k}\geq 1$, such that

$$\begin{cases} |T_{r,k}^q|=\lceil m_1/(nt)\rceil & \text{for } q\in\{1,\dots,a_{r,k}-1\},\\ |T_{r,k}^q|\leq\lceil m_1/(nt)\rceil & \text{for } q=a_{r,k}.\end{cases}$$

Define $p_r=\sum_{k=1}^n a_{r,k}$ and notice that $p_r\leq 2n$: there are at most n parts of size exactly $\lceil m_1/(nt)\rceil$ due to Equation (6.1), and at most n parts of size strictly less than $\lceil m_1/(nt)\rceil$ (these parts are among the n parts with $q=a_{r,k}$). To each pair (k,q) with $k\in\{1,\dots,n\}$ and $q\in\{1,\dots,a_{r,k}\}$, we assign an arbitrary index in $\{1,\dots,p_r\}$, denoted $\rho_r(k,q)$, in a bijective way. Finally, for each $x\in\{1,\dots,u\}$, we construct the $n\times 2n$ matrix $\tilde{A}_r^{(x)}$ as follows: for all $i\in\{1,\dots,n\}$ and all $k'\in\{1,\dots,2n\}$,

$$\tilde{A}_r^{(x)}[i,k']=\begin{cases} A_r^{(x)}[i,k] & \text{if } k'\in\{1,\dots,p_r\} \text{ and } A_r^{(x)}[i,k]\in T_{r,k}^q, \text{ where } (k,q)=\rho_r^{-1}(k'),\\ \infty & \text{otherwise.}\end{cases}$$

This means that each finite entry of $A_r^{(x)}$ appears in $\tilde{A}_r^{(x)}$, in the same row but generally in a different column. By construction, Equation (6.2) holds. The overall cost of this (classical) preprocessing step is $\tilde{O}(n^2tu)$ time.

Preprocessing: recording relevant information about the input matrices

Since the complexity of the quantum procedure described in the last part of the proof will depend crucially on the way information about matrices $\tilde{A}_r^{(x)}$ and $B_r^{(y)}$ is stored, we introduce adequate data structures to record this information.

For each $x\in\{1,\dots,u\}$, we do the following. For all $r\in\{1,\dots,t\}$ we list the finite entries in each column of $\tilde{A}_r^{(x)}$, classically in time $\tilde{O}(n^2t)$, and create a 3-dimensional array $\mathsf{U}^{(x)}$ such that $\mathsf{U}^{(x)}[r,k',b]$ records the index of the row of the b-th finite entry in the k'-th column of $\tilde{A}_r^{(x)}$, for each $r\in\{1,\dots,t\}$, each $k'\in\{1,\dots,2n\}$, and each $b\in\{1,\dots,\mathrm{col}(\tilde{A}_r^{(x)},k')\}$.

For each $y\in\{1,\dots,v\}$, we do the following. We construct, classically in time $O(n^2t)$, a list containing all the finite entries of $B_1^{(y)},\dots,B_t^{(y)}$. Let us denote the total number of these finite entries by $m^{(y)}$, and remember that we have $\sum_{y=1}^{v}m^{(y)}\leq m_2$. We then create an array $\mathsf{V}^{(y)}$ of size $m^{(y)}$: for each $a\in\{1,\dots,m^{(y)}\}$, if the a-th element of the list is $B_r^{(y)}[k,j]$, then $\mathsf{V}^{(y)}[a]$ is set to the 3-tuple (r,k,j).

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

Construction of the matrices $\hat{A}_r^{(x)}$ and $\hat{B}_r^{(y)}$

For each $r \in \{1,\dots,t\}$ and each $x \in \{1,\dots,u\}$, we construct an $n \times 2n$ Boolean matrix $\hat{A}_r^{(x)}$ as follows: for all $i \in \{1,\dots,n\}$ and all $k' \in \{1,\dots,2n\}$,

$$\hat{A}_r^{(x)}[i,k'] = 1 \text{ iff } \tilde{A}_r^{(x)}[i,k'] \neq \infty.$$

For each $r \in \{1,\dots,t\}$ and each $y \in \{1,\dots,v\}$, we construct a $2n \times n$ Boolean matrix $\hat{B}_r^{(y)}$ as follows: for all $k' \in \{1,\dots,2n\}$ and all $j \in \{1,\dots,n\}$,

$$\hat{B}_r^{(y)}[k',j] = 1 \text{ iff } k' \in \{1,\dots,p_r\} \text{ and } B_r^{(y)}[k,j] \geq \max T_{r,k}^q, \text{ where } (k,q) = \rho_r^{-1}(k').$$

These are the matrices mentioned in the statement of the lemma. The overall cost of this (classical) construction step is $\tilde{O}(n^2 t(u+v))$ time.

Relation with the matrix C_2

For each $r \in \{1,\dots,t\}$, each $x \in \{1,\dots,u\}$ and each $y \in \{1,\dots,v\}$, consider the Boolean product $\hat{A}_r^{(x)} \cdot \hat{B}_r^{(y)}$. This product gives us some of the non-zero entries of $A_r^{(x)} * B_r^{(y)}$, but not all. Indeed, by definition, $\hat{A}_r^{(x)}[i,k'] = 1$ if and only if $A_r^{(x)}[i,k] \in T_{r,k}^q$, where $(k,q) = \rho_r^{-1}(k')$. The indexes of the non-zero entries of $\hat{A}_r^{(x)} \cdot \hat{B}_r^{(y)}$ are thus precisely all the $(i,j) \in \{1,\dots,n\} \times \{1,\dots,n\}$ for which there exists some $k \in \{1,\dots,n\}$ satisfying

$$A_r^{(x)}[i,k] \in T_{r,k}^q \text{ for some } q \in \{1,\dots,a_{r,k}\} \text{ and } B_r^{(y)}[k,j] \geq \max T_{r,k}^q.$$

Let us now consider the remaining non-zero entries of $A_r^{(x)} * B_r^{(y)}$: the $(i,j) \in \{1,\dots,n\} \times \{1,\dots,n\}$ for which there exists some $k \in \{1,\dots,n\}$ satisfying

$$A_r^{(x)}[i,k] \in T_{r,k}^q \text{ for some } q \in \{1,\dots,a_{r,k}\} \text{ and } A_r^{(x)}[i,k] \leq B_r^{(y)}[k,j] < \max T_{r,k}^q. \tag{6.3}$$

Define the $n \times n$ matrix D with entries in $S \cup \{(0,0)\}$ as follows. For any $(i,j) \in \{1,\dots,n\} \times \{1,\dots,n\}$, the entry $D[i,j]$ is the largest element $(x,y) \in S$ such that Equation (6.3) holds for some $r \in \{1,\dots,t\}$ and some $k \in \{1,\dots,n\}$, if at least one such (x,y) exists, and $D[i,j] = (0,0)$ otherwise.

We then have

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

for all $(i,j) \in \{1,\dots,n\} \times \{1,\dots,n\}$, as claimed in the statement of the lemma.

Construction of the matrix D

We finally show how to compute the matrix D. The idea is to find, for all $(x,y) \in S$ in decreasing order, all the pairs of indexes $(i,j) \in \{1,\dots,n\} \times \{1,\dots,n\}$ such that Equation (6.3) holds for some $r \in \{1,\dots,t\}$ and some $k \in \{1,\dots,n\}$, and strike out those pairs as soon as they are found.

The procedure for computing D is described in Figure 2. The set R, initially empty, records all pairs (i,j) for which $D[i,j]$ has already been computed. During the loop of Steps 8-13 the procedure enumerates all the $(i,j) \in (\{1,\dots,n\} \times \{1,\dots,n\}) \setminus R$ such that Equation (6.3) holds for some $r \in \{1,\dots,t\}$ and some

```
1   $R \leftarrow \emptyset$;
2   for all $(i,j) \in \{1,\ldots,n\} \times \{1,\ldots,n\}$ do $D[i,j] \leftarrow (0,0)$; enddo
3   for all $y \in \{1,\ldots,v\}$, all $r \in \{1,\ldots t\}$ and all $(j,k) \in \{1,\ldots,n\} \times \{1,\ldots,n\}$ do
4       compute the smallest $q \in \{1,\ldots,a_{r,k}\}$ satisfying $B_r^{(y)}[k,j] < \max T_{r,k}^{q}$ and denote it by $q_{rkj}^{(y)}$;
5   enddo
6   for all $(x,y) \in S$ in decreasing order do
7       test-full $\leftarrow$ false;
8       while test-full = false do
9           find $(r,i,j,k) \in \{1,\ldots,t\} \times \{1,\ldots,n\}^3$ such that $(i,j) \notin R$ and $\tilde{A}_r^{(x)}[i,\rho_r(k,q_{rkj}^{(y)})] \leq B_r^{(y)}[k,j]$;
 #          comment: the search of Step 9 is actually done over $\Gamma^{(x,y)} \subset \{1,\ldots,t\} \times \{1,\ldots,n\}^3$
10          if a solution $(r,i,j,k)$ is found
11              then $D[i,j] \leftarrow (x,y)$; $R \leftarrow R \cup \{(i,j)\}$;
12              else test-full $\leftarrow$ true;
13      enddo
14  enddo
```

Figure 2: Procedure computing the matrix D.

$k \in \{1,\ldots,n\}$. Note that only the non-$(-\infty)$ entries of $B_r^{(y)}$ need to be considered and, from Equation (6.3), for each such non-$(-\infty)$ entry $B_r^{(y)}[k,j]$ only the non-(∞) entries $A_r^{(x)}[i,k]$ of $A_r^{(x)}$ such that

$$A_r^{(x)}[i,k] \in T_{r,k}^{q_{rkj}^{(y)}}$$

need to be considered, where $q_{rkj}^{(y)}$ is the smallest integer in $\{1,\ldots,a_{r,k}\}$ such that

$$B_r^{(y)}[k,j] < \max T_{r,k}^{q_{rkj}^{(y)}}.$$

By construction, these non-(∞) entries of $A_r^{(x)}$ are in the $\rho_r(k,q_{rkj}^{(y)})$-th column of $\tilde{A}_r^{(x)}$. The loop of Steps 8-13 thus performs successive quantum searches over the set

$$\Gamma^{(x,y)} = \Big\{(r,i,j,k) \in \{1,\ldots,t\} \times \{1,\ldots,n\}^3 \,\Big|\, B_r^{(y)}[k,j] \neq -\infty \text{ and } \tilde{A}_r^{(x)}[i,\rho_r(k,q_{rkj}^{(y)})] \neq \infty\Big\},$$

looking for elements $(r,i,j,k) \in \Gamma^{(x,y)}$ such that

$$(i,j) \notin R \text{ and } \tilde{A}_r^{(x)}[i,\rho_r(k,q_{rkj}^{(y)})] \leq B_r^{(y)}[k,j].$$

The procedure of Figure 2 correctly computes the matrix D whenever the quantum enumeration does not err, that is, with probability at least $1-1/\mathrm{poly}(n)$ if safe Grover search is used, as discussed in

Section 2. Let us consider its time complexity. The cost of Step 2 is $\tilde{O}(n^2)$, and the cost of the loop of Steps 3-5 is $\tilde{O}(n^2tv)$ since each $q^{(y)}_{rkj}$ can be found in $\mathrm{poly}(\log n)$ time using binary search.

In order to evaluate the cost of the loop of Steps 8-13, we need to discuss in more details how to perform the quantum search over the set $\Gamma^{(x,y)}$ since subtle issues arise when considering how to access time-efficiently the relevant entries of the matrices and how to check if an element is a solution in $\mathrm{poly}(n)$ time. Note that

$$|\Gamma^{(x,y)}| = \sum_{r=1}^{t} \sum_{\substack{j,k \text{ such that} \\ B_r^{(y)}[k,j]\neq-\infty}} \mathrm{col}(\tilde{A}_r^{(x)}, \rho_r(k, q^{(y)}_{rkj})).$$

We define a bijection g from the set $\{1,\ldots,|\Gamma^{(x,y)}|\}$ to the set $\Gamma^{(x,y)}$ as follows. Remember that the data structures $\mathsf{U}^{(x)}$ and $\mathsf{V}^{(y)}$ are available, recording information about the $\tilde{A}_r^{(x)}$'s and the $B_r^{(y)}$'s, respectively. For notational convenience for each $a \in \{1,\ldots,m^{(y)}\}$ with corresponding value $\mathsf{V}^{(y)}[a] = (r,k,j)$, we will write $\mathsf{V}_1[a] = r$, $\mathsf{V}_2[a] = k$, $\mathsf{V}_3[a] = j$ and $\mathsf{W}[a] = \rho_r(k, q^{(y)}_{rkj})$. Note that these four values can be immediately obtained from $\mathsf{V}^{(y)}[a]$. We define the function g as follows: for all $a \in \{1,\ldots,m^{(y)}\}$ and all $b \in \{1,\ldots,\mathrm{col}(\tilde{A}^{(x)}_{\mathsf{V}_1[a]}, \mathsf{W}[a])\}$,

$$g\left(b + \sum_{c=1}^{a-1} \mathrm{col}(\tilde{A}^{(x)}_{\mathsf{V}_1[c]}, \mathsf{W}[c])\right) = (\mathsf{V}_1[a], \mathsf{U}^{(x)}\Big[\mathsf{V}_1[a], \mathsf{W}[a], b\Big], \mathsf{V}_3[a], \mathsf{V}_2[a]).$$

It is easy to check that g is a bijection from $\{1,\ldots,|\Gamma^{(x,y)}|\}$ to $\Gamma^{(x,y)}$. The crucial point here is that the function g can be evaluated in $\mathrm{poly}(\log n)$ time since $\mathsf{U}^{(x)}$ and $\mathsf{V}^{(y)}$ are available (in particular, given any $z \in \{1,\ldots,|\Gamma^{(x,y)}|\}$ one can find the values a and b such that $z = b + \sum_{c=1}^{a-1} \mathrm{col}(\tilde{A}^{(x)}_{\mathsf{V}_1[c]}, \mathsf{W}[c])$ efficiently, using binary search for instance). We can then implement Step 9 by performing quantum searches over the set $\{1,\ldots,|\Gamma^{(x,y)}|\}$. From the discussion in Section 2, the time complexity of the loop of Steps 8-13, for fixed (x,y), is thus

$$\begin{aligned}
&\tilde{O}\left(\sqrt{|\Gamma^{(x,y)}| \times (\lambda^{(x,y)}+1)}\right) \\
&= \tilde{O}\left(\sqrt{\left(\sum_{r=1}^{t} \sum_{\substack{j,k \text{ such that} \\ B_r^{(y)}[k,j]\neq-\infty}} \mathrm{col}(\tilde{A}_r^{(x)}, \rho_r(k, q^{(y)}_{rkj}))\right) \times (\lambda^{(x,y)}+1)}\right),
\end{aligned}$$

where $\lambda^{(x,y)}$ denotes the number of elements found during the execution of the loop (i.e., the number of new entries of D computed).

The total cost of the procedure of Figure 2 is then

$$\tilde{O}\left(n^2tv + \sum_{x=1}^{u}\sum_{y=1}^{v} \sqrt{\left(\sum_{r=1}^{t} \sum_{\substack{j,k \text{ such that} \\ B_r^{(y)}[k,j]\neq-\infty}} \mathrm{col}(\tilde{A}_r^{(x)}, \rho_r(k, q^{(y)}_{rkj}))\right) \times (\lambda^{(x,y)}+1)}\right).$$

Using Equation (6.2), the inequality $\sum_{x=1}^{u}\sum_{y=1}^{v}\lambda^{(x,y)} \leq n^2$, and the Cauchy-Schwarz inequality, we can rewrite this expression as

$$\begin{aligned}
&\tilde{O}\left(n^2tv + \sum_{y=1}^{v}\sqrt{\left(\sum_{r=1}^{t}\sum_{\substack{j,k \text{ such that}\\ B_r^{(y)}[k,j]\neq-\infty}}\frac{m_1}{nt}\right)\times\left(u+\sum_{x=1}^{u}\lambda^{(x,y)}\right)}\right)\\
&=\tilde{O}\left(n^2tv + \sqrt{\frac{m_1m_2}{nt}\times\left(uv+\sum_{x=1}^{u}\sum_{y=1}^{v}\lambda^{(x,y)}\right)}\right)\\
&=\tilde{O}\left(n^2tv + \sqrt{\frac{m_1m_2n}{t}}+\sqrt{\frac{m_1m_2uv}{tn}}\right).
\end{aligned}$$

This concludes the description of how to construct the matrix D.

Since the preprocessing has cost $\tilde{O}(n^2t(u+v))$, the overall complexity of the algorithm is

$$\tilde{O}\left(n^2t(u+v) + \sqrt{\frac{m_1m_2n}{t}}+\sqrt{\frac{m_1m_2uv}{tn}}\right).$$

This concludes the proof of Lemma 3.3.

6.2 Proof of Proposition 3.4

The algorithm is essentially the same as in the proof of Proposition 3.2. The only change is that a classical algorithm, which we describe below, is used instead of the quantum algorithm in Lemma 3.3. In the proof of Lemma 3.3 we use classical enumeration (i.e., exhaustive search) instead of quantum enumeration in the procedure of Figure 2. The complexity of the loop of Steps 8-13 of the procedure of Figure 2 is thus $|\Gamma^{(x,y)}|$, and the total complexity of the procedure becomes

$$\tilde{O}\left(n^2tv + \sum_{x=1}^{u}\sum_{y=1}^{v}\sum_{r=1}^{t}\sum_{\substack{j,k \text{ such that}\\ B_r^{(y)}[k,j]\neq-\infty}}\mathrm{col}(\tilde{A}_r^{(x)},\rho_r(k,q_{rkj}^{(y)}))\right)=\tilde{O}\left(n^2tv+\frac{m_1m_2}{tn}\right).$$

The overall complexity of the classical version of Lemma 3.3 is thus

$$\tilde{O}\left(n^2t(u+v)+n^2tv+\frac{m_1m_2}{tn}\right)=\tilde{O}\left(n^2t(u+v)+\frac{m_1m_2}{tn}\right),$$

which gives the upper bound claimed.

Acknowledgements

This work is supported by the Grant-in-Aid for Young Scientists (B) No. 24700005, the Grant-in-Aid for Young Scientists (A) No. 16H05853 and the Grant-in-Aids for Scientific Research (A) Nos. 24240001 and 16H01705 of the Japan Society for the Promotion of Science, and the Grant-in-Aid for Scientific Research on Innovative Areas No. 24106009 of the Ministry of Education, Culture, Sports, Science and Technology in Japan.

References

[1] RASMUS RESEN AMOSSEN AND RASMUS PAGH: Faster join-projects and sparse matrix multiplications. In *Proceedings of the 12th International Conference on Database Theory*, pp. 121–126, 2009. 4, 14, 16

[2] MICHEL BOYER, GILLES BRASSARD, PETER HØYER, AND ALAIN TAPP: Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4-5):493–505, 1998. 6

[3] HARRY BUHRMAN AND ROBERT ŠPALEK: Quantum verification of matrix products. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 880–889, 2006. 3, 5

[4] PETER BÜRGISSER, MICHAEL CLAUSEN, AND MOHAMMAD AMIN SHOKROLLAHI: *Algebraic complexity theory*. Springer, 1997. 5

[5] ALEXANDER M. DAVIE AND ANDREW J. STOTHERS: Improved bound for complexity of matrix multiplication. *Proceedings of the Royal Society of Edinburgh*, 143A:351–370, 2013. 5

[6] RAN DUAN AND SETH PETTIE: Fast algorithms for (max, min)-matrix multiplication and bottleneck shortest paths. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 384–391, 2009. 1, 2, 3, 4, 5, 6, 18

[7] DIDIER DUBOIS AND HENRI PRADE: *Fuzzy sets and systems: Theory and applications*. Academic Press, 1980. 2

[8] CHRISTOPH DÜRR AND PETER HØYER: A quantum algorithm for finding the minimum. arXiv:quant-ph/9607014, 1996. 2, 10

[9] LOV K. GROVER: A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, pp. 212–219, 1996. 2

[10] XIAOHAN HUANG AND VICTOR Y. PAN: Fast rectangular matrix multiplication and applications. *Journal of Complexity*, 14(2):257–299, 1998. 5

[11] STACEY JEFFERY, ROBIN KOTHARI, AND FRÉDÉRIC MAGNIEZ: Improving quantum query complexity of Boolean matrix multiplication using graph collision. In *Proceedings of the 39th International Colloqium on Automata, Languages, and Programming, Part I, Lecture Notes in Computer Science Vol. 7391*, pp. 522–532, 2012. 3, 5, 16

[12] FRANÇOIS LE GALL: Faster algorithms for rectangular matrix multiplication. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 514–523, 2012. 5

[13] FRANÇOIS LE GALL: Improved output-sensitive quantum algorithms for Boolean matrix multiplication. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1464–1476, 2012. 3, 5, 6

[14] FRANÇOIS LE GALL: A time-efficient output-sensitive quantum algorithm for Boolean matrix multiplication. In *Proceedings of the 23rd International Symposium on Algorithms and Computation, Lecture Notes in Computer Science Vol. 7676*, pp. 639–648, 2012. 3, 5, 6, 16

[15] FRANÇOIS LE GALL: Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, pp. 296–303, 2014. 5

[16] ANDRZEJ LINGAS: A fast output-sensitive algorithm for Boolean matrix multiplication. *Algorithmica*, 61(1):36–50, 2011. 16

[17] FRÉDÉRIC MAGNIEZ, MIKLOS SANTHA, AND MARIO SZEGEDY: Quantum algorithms for the triangle problem. *SIAM Journal on Computing*, 37(2):413–424, 2007. 6

[18] JIRÍ MATOUŠEK: Computing dominances in E^n. *Information Processing Letters*, 38(5):277–278, 1991. 3, 11

[19] ASAF SHAPIRA, RAPHAEL YUSTER, AND URI ZWICK: All-pairs bottleneck paths in vertex weighted graphs. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 978–985, 2007. 1, 2

[20] ANDREW STOTHERS: *On the Complexity of Matrix Multiplication*. Ph. D. thesis, University of Edinburgh, 2010. 5

[21] VIRGINIA VASSILEVSKA: *Efficient Algorithms for Path Problems in Weighted Graphs*. Ph. D. thesis, Carnegie Mellon University, 2008. 1, 2, 5

[22] VIRGINIA VASSILEVSKA AND RYAN WILLIAMS: Finding a maximum weight triangle in $n^{3-\delta}$ time, with applications. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pp. 225–231, 2006. 1, 3, 5, 11, 14

[23] VIRGINIA VASSILEVSKA, RYAN WILLIAMS, AND RAPHAEL YUSTER: All pairs bottleneck paths and max-min matrix products in truly subcubic time. *Theory of Computing*, 5(1):173–189, 2009. 1, 2, 4, 5, 10

[24] VIRGINIA VASSILEVSKA WILLIAMS: Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference*, pp. 887–898, 2012. 5

[25] RAPHAEL YUSTER: Efficient algorithms on sets of permutations, dominance, and real-weighted APSP. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 950–957, 2009. 1, 3, 14

[26] RAPHAEL YUSTER AND URI ZWICK: Fast sparse matrix multiplication. *ACM Transactions on Algorithms*, 1(1):2–13, 2005. 4, 14, 16

AUTHORS

François Le Gall
Kyoto University
legall@i.kyoto-u.ac.jp
http://www.francoislegall.com/

Harumichi Nishimura
Nagoya University
hnishimura@math.cm.is.nagoya-u.ac.jp
http://www.math.cm.is.nagoya-u.ac.jp/~hnishimura/e-index.html