Quantum and approximation algorithms for maximum witnesses of Boolean matrix products

Miroslaw Kowaluk\textsuperscript{1} and Andrzej Lingas\textsuperscript{2}

\textsuperscript{1} Institute of Informatics, University of Warsaw, Warsaw, Poland. kowaluk@mimuw.edu.pl
\textsuperscript{2} Department of Computer Science, Lund University, 22100 Lund, Sweden. Andrzej.Lingas@cs.lth.se

Abstract. The problem of finding maximum (or minimum) witnesses of the Boolean product of two Boolean matrices (MW for short) has a number of important applications, in particular the all-pairs lowest common ancestor (LCA) problem in directed acyclic graphs (dags). The best known upper time-bound on the MW problem for $n \times n$ Boolean matrices of the form $O(n^{2.575})$ has not been substantially improved since 2006. In order to obtain faster algorithms for this problem, we study quantum algorithms for MW and approximation algorithms for MW (in the standard computational model). Some of our quantum algorithms are input or output sensitive. Our fastest quantum algorithm for the MW problem, and consequently for the related problems, runs in time $\tilde{O}(n^{2+\lambda/2}) / 2^{\lambda}$, where $\lambda$ satisfies the equation $\omega(1, \lambda, 1) = 1 + 1/\lambda$ and $\omega(1, \lambda, 1)$ is the exponent of the multiplication of an $n \times n^\lambda$ matrix by an $n^\lambda \times n$ matrix. Next, we consider a relaxed version of the MW problem (in the standard model) asking for reporting a witness of bounded rank (the maximum witness has rank 1) for each non-zero entry of the matrix product. First, by adapting the fastest known algorithm for maximum witnesses, we obtain an algorithm for the relaxed problem that reports for each non-zero entry of the product matrix a witness of rank at most $\ell$ in time $\tilde{O}((n/\ell)n^{\omega(1, \log n, 1)} \ell^{11})$. Then, by reducing the relaxed problem to the so called $k$-witness problem, we provide an algorithm that reports for each non-zero entry $C[i, j]$ of the product matrix $C$ a witness of rank $O(WC(i, j)/k)$, where $WC(i, j)$ is the number of witnesses for $C[i, j]$, with high probability. The algorithm runs in $O(n^{\omega k^{0.4653}} + n^{2+o(1)}k)$ time, where $\omega = \omega(1, 1, 1)$.

1 Introduction

If $A$ and $B$ are two $n \times n$ Boolean matrices and $C$ is their Boolean matrix product then for any entry $C[i, j] = 1$ of $C$, a witness is an index $k$ such that $A[i, k] \land B[k, j] = 1$. The largest (or, smallest) possible witness for an entry is called the maximum witness (or minimum witness, respectively) for the entry.

The problem of finding “witnesses” of Boolean matrix product has been studied for decades mostly because of its applications to shortest-path problems \cite{23}. The problem of finding maximum witnesses of Boolean matrix product (MW for short) has been studied first in \cite{7} in order to obtain faster algorithms for all-pairs lowest common ancestor (LCA) problem in directed acyclic
graphs (dags) \([10]\). It has found many other applications since then including the all-pairs bottleneck weight path problem \([20]\) and finding for a set of edges in a vertex-weighted graph heaviest triangles including an edge from the set \([21]\). The fastest known algorithm for the MW problem and the aforementioned problems runs in \(O(n^{2+\lambda})\) time \([7]\), where \(\lambda\) satisfies the equation \(\omega(1, \lambda, 1) = 1 + 2\lambda\) and \(\omega(1, \lambda, 1)\) is the exponent of the multiplication of an \(n \times n^\lambda\) matrix by an \(n^\lambda \times n\) matrix. The currently best bounds on \(\omega(1, \lambda, 1)\) follow from a fact in \([5,12]\) (see Fact 4 in Preliminaries) combined with the recent improved estimations on the parameters \(\omega = \omega(1, 1, 1)\) and \(\alpha\) (see Preliminaries) \([14,17]\). They yield an \(O(n^{2.509})\) upper bound on the running time of the algorithm (originally, \(O(n^{2.575})\) \([7]\)). For faster algorithms in sparse cases, see \([6]\).

In this paper, we study two different approaches to deriving faster algorithms for the problem of computing maximum (or minimum) witnesses of the Boolean product of two \(n \times n\) Boolean matrices (MW for short). The first approach is to consider the MW problem in the more powerful model of quantum computation. The other approach is to relax the MW problem (in the standard model) by allowing its approximation.

In the first part of our paper, we present quantum algorithms for the MW problem assuming a Quantum Random Access Machine (QRAM) model \([19]\). First, we consider a straightforward algorithm for MW that uses the quantum minimum search due to Dürr and Høyer \([8]\) for each entry of the product matrix separately in order to find its maximum witness\(^3\). It runs in \(\tilde{O}(n^{2.5})\) time. By adding as a preprocessing a known output-sensitive quantum algorithm for Boolean matrix product, we obtain an output-sensitive quantum algorithm for MW running in \(\tilde{O}(n\sqrt{s} + s\sqrt{n})\) time, where \(s\) is the number of non-zero entries in the product matrix. By refining the straightforward algorithm in a different way, we obtain also an input-sensitive quantum algorithm for MW running in \(\tilde{O}(n^{2+1.5m^{0.5}})\) time, where \(m\) is the number of non-zero entries in the sparsest among the two input matrices. Then, we combine the idea of multiplication of rectangular submatrices of the input Boolean matrices with that of using the quantum minimum search of Dürr and Høyer in order to obtain our fastest quantum algorithm for MW running in \(\tilde{O}(n^{2+\lambda/2})\) time, where \(\lambda\) satisfies the equation \(\omega(1, \lambda, 1) = 1 + 1.5\lambda\). By the currently best bounds on \(\omega(1, \lambda, 1)\), the

---

\(^3\) For somewhat related applications of the quantum minimum search of Dürr and Høyer to shortest path problems see \([18]\).
running time of our algorithm is \( O(n^{2.434}) \) \(^4\) We obtain the same asymptotic upper time-bounds for the aforementioned problems related to MW.

In the second part of our paper, we consider a relaxed version of the MW problem (in the standard model) asking for reporting a witness of bounded rank (the maximum witness has rank 1) for each non-zero entry of the matrix product. First, by adapting the fastest known algorithm for maximum witnesses, we obtain an algorithm for the relaxed problem that reports for each non-zero entry of the product matrix a witness of rank at most \( \ell \) in time \( \tilde{O}((n/\ell)n^{\omega(1, \log_n \ell, 1)}) \).

Then, by reducing the relaxed problem to the so called \( k \)-witness problem, we provide an algorithm that reports for each non-zero entry \( C[i,j] \) of the product matrix \( C \) a witness of rank \( O([W_C(i,j)/k]) \) with high probability, where \( W_C(i,j) \) is the number of witnesses for \( C[i,j] \). The algorithm runs in \( \tilde{O}(n^{\omega(k^{0.4653} + n^{2+o(1)}})k) \) time, where \( \omega \) is the exponent of fast \( n \times n \) matrix multiplication.

1.1 Organization

In Preliminaries, we provide some basic notions and/or facts on matrix multiplication and quantum computation. In Section 3, we present our basic procedure for searching an interval of indices for the maximum witness, the straightforward quantum algorithm for MW implied by the procedure, and the output-sensitive and input-sensitive refinements of the algorithm. In Section 4, we present and analyze our fastest in the general case quantum algorithm for MW. In Section 5, we provide applications of our quantum algorithms to the problems related to MW. In Section 6, we present our approximation algorithms for MW in the standard computational model. We conclude with final remarks.

2 Preliminaries

For a positive integer \( r \), we shall denote the set of positive integers not greater than \( r \) by \([r]\).

For a matrix \( D \), \( D^t \) denotes its transpose.

A witness for a non-zero entry \( C[i,j] \) of the Boolean matrix product \( C \) of a Boolean \( p \times q \) matrix \( A \) and a Boolean \( q \times r \) matrix \( B \) is any index \( k \in [q] \) such that \( A[i,k] \) and \( B[k,j] \) are equal to 1. The number of witnesses for \( C[i,j] \) is denoted by \( W_C(i,j) \). A witness \( k \) for \( C[i,j] \) is of rank \( h \) if there are exactly \( h-1 \)

\(^4\) In the upper bound on the running time of our fastest quantum algorithm for MW, we could replace \( \omega(\ ) \) by its generalization to include the model of quantum computation. However, since no quantum algorithms for Boolean matrix product faster than those algebraic ones in the general case are known so far, this would not yield an improvement at present.
witnesses for this entry greater than $k$. The witness of rank 1 is the maximum witness for $C[i,j]$. The witness problem is to report a witness for each non-zero entry of the Boolean matrix product of the two input matrices. The maximum witness problem (MW) is to report the maximum witness for each non-zero entry of the Boolean matrix product of the two input matrices.

Recall that for natural numbers $p$, $q$, $r$, and $\omega(p, q, r)$ denotes the exponent of fast matrix multiplication for rectangular matrices $n^p \times n^q$ and $n^q \times n^r$, respectively. For convenience, $\omega = \omega(1, 1, 1)$. The following recent upper bound on $\omega$ is due Alman and Vassilevska Williams [1].

**Fact 1** The fast matrix multiplication algorithm for $n \times n$ matrices runs in $O(n^{\omega})$ time, where $\omega$ is not greater than $2.37286$ [11] (cf. [14] [22]).

Alon and Naor provided almost equally fast solution to the witness problem for square Boolean matrices in [3]. It can be easily generalized to include the Boolean product of two rectangular Boolean matrices of sizes $n \times n^q$ and $n^q \times n$, respectively. The asymptotic matrix multiplication time $n^\omega$ is replaced by $n^{\omega(1,q,1)}$ in the generalization.

**Fact 2** For $q \in (0, 1]$, the witness problem for the Boolean matrix product of an $n \times n^q$ Boolean matrix with an $n^q \times n$ Boolean matrix can be solved (deterministically) in $\tilde{O}(n^{\omega(1,q,1)})$ time.

Let $\alpha$ stand for $\sup\{0 \leq r \leq 1 : \omega(1,r,1) = 2 + o(1)\}$. The following recent lower bound on $\alpha$ is due to Le Gall and Urrutia [17].

**Fact 3** The inequality $\alpha > 0.31389$ holds [17].

Coppersmith [5] and Huang and Pan [12] proved the following fact.

**Fact 4** The inequality $\omega(1,r,1) \leq \beta(r)$ holds, where $\beta(r) = 2 + o(1)$ for $r \in [0, \alpha]$ and $\beta(r) = 2 + \frac{\omega - 2}{1 - \alpha}(r - \alpha) + o(1)$ for $r \in [\alpha, 1]$ [5][12].

It will be the most convenient to formulate our quantum algorithms in the model of Quantum Random Access Machine (QRAM) saving the reader a lot of technical details of alternative formulations in the quantum circuit model [19]. Thus, our quantum algorithm can access any entry of any input matrix $A$ in an access random manner (cf. [15][13]). More precisely, following [18], we assume that there is an oracle $O_A$ which for $i$, $j \in [n]$ and $z \in \{0, 1\}^*$, maps the state $|i\rangle|j\rangle|0\rangle|z\rangle$ into the state $|i\rangle|j\rangle|A[i,j]\rangle|z\rangle$. When a whole table $T$ is stored in the random access memory of QRAM such an oracle $o_T$ corresponding to $T$ is implicit. We shall estimate the time complexity of our quantum algorithms in the unit cost model, in particular we shall assign unit cost to each call to an
oracle. In case the time complexity of our quantum algorithm exceeds the size of the input matrices, we may assume w.l.o.g. that the input matrices are just read into the QRAM memory.

Following Le Gall [15], we can generalize the definition of a quantum algorithm for Boolean matrix product to include the MW problem.

**Definition 1.** A quantum algorithm for witnesses of Boolean matrix product (or the MW problem) is a quantum algorithm that when given access to oracles $O_A$ and $O_B$ corresponding to Boolean matrices $A$ and $B$, computes with probability at least $2/3$ all the non-zero entries of the product $A \times B$ along with one witness (the maximum witness, respectively) for each non-zero entry.

Note the probability of at least $2/3$ can be enhanced to at least $1 - n^{-\gamma}$, for $\gamma \geq 1$, by iterating the algorithm $O(\log n)$ times. When the size of the input is bounded by $\text{poly}(n)$, one uses the term *almost certainly* for the latter probability.

In fact, all our quantum algorithms for MW but the output sensitive one report also “No” for each zero entry of the product matrix.

### 3 Quantum search for the maximum witness

One can find the maximum witness for a given entry of the Boolean product of two Boolean $n \times n$ matrices in $\tilde{O}(\sqrt{n})$ time with high probability by recursively using Grover’s quantum search [11] interleaved with a binary search. However, the most convenient is to use a specialized variant of Grover’s search due to Dürr and Høyer [4,8] for finding an entry of the minimum value in a table.

**Fact 5 (Dürr and Høyer [8])** Let $T[k]$, $1 \leq k \leq n$ be an unsorted table where all values are distinct. Given an oracle for $T$, the index $k$ for which $T[k]$ is minimum can be found by a quantum algorithm with probability at least $1/2$ in $O(\sqrt{n})$ time.

Using this fact, we can design the following procedure $\text{MaxWit}(A, B, i, j)$ returning the maximum witness of the entry $C[i, j]$ (if any) of the product $C$ of two Boolean $n \times n$ matrices $A$ and $B$.

**procedure MaxWit(A, B, i, j)**

*Input:* oracles corresponding to a Boolean $p \times q$ matrix $A$ and a Boolean $q \times r$ matrix $B$, and indices $i \in [p]$, $j \in [r]$.

*Output:* if the $C[i, j]$ entry of the Boolean product $C$ of $A$ and $B$ has a witness then its maximum witness in $[q]$ otherwise “No”.

1. $n \leftarrow \max\{p, q, r\}$
2. Define an oracle for a virtual, one-dimensional integer table $T[k]$, $k \in [q]$
by $T[k] = 2^n - A[i, k]B[k, j]n - k$.
3. Iterate $O(\log n)$ times the algorithm of Dürr and Høyer for $T$ and set $k'$
to the index minimizing $T$.
4. If $T[k'] < n$ then return $k'$ as the maximum witness otherwise return “No”.

Lemma 1. Let $\beta$ be a positive integer. By repetitively using the algorithm of Dürr and Høyer, $\text{MaxWit}(A, B, i, j)$ can be implemented in $\tilde{O}(\beta \sqrt{n})$ time such that it returns a correct answer with probability at least $1 - n^{-\beta}$.

Proof. To begin with observe that for $k, k' \in [q]$, if $k \neq k'$ then $T[k] \neq T[k']$. This obviously holds for $k, k' \in [q]$ if $A[i, k]B[k, j] = A[i, k']B[k', j]$ as well when $A[i, k]B[k, j] \neq A[i, k']B[k', j]$. Furthermore, the value of $T[k]$ can be computed with the help of the oracles for $A, B$ in constant time in the QRAM model. Next, suppose that the minimum value of $T$ is achieved for the index $k'$. It is easily seen that if $T[k'] < n$ then $k'$ is the maximum witness of $C[i, j]$ and otherwise $C[i, j]$ does not have any witness. By running the minimum search algorithm of Dürr and Høyer $O(\beta \log n)$ times, we can identify the maximum witness of $C[i, j]$ with probability at least $1 - n^{-\beta}$ in $\tilde{O}(\beta \sqrt{n})$ time.

3.1 A straightforward quantum algorithm for MW

By Lemma 1, a straightforward $\tilde{O}(n^{2.5})$-time method for MW is just to run the procedure $\text{MaxWit}(A, B, i, j)$ with appropriately large constant $\beta$ for each entry $C[i, j]$ of the product matrix $C$. See Algorithm 1 for a pseudo-code of this method.

Algorithm 1

Input: oracles corresponding to Boolean $n \times n$ matrices $A$, $B$.
Output: maximum witnesses for all non-zero entries of the Boolean product of $A$ and $B$, and “No” for all zero entries of the product.

for all $i, j \in [n]$ do
$\text{MaxWit}(A, B, i, j)$

Note that Algorithm 1 returns also “No” for zero entries of $C$.

By Lemma 1 with sufficiently large $\beta$, we obtain immediately the following theorem.

Theorem 1. Algorithm 1 solves the MW problem in $\tilde{O}(n^{2.5})$ time.

3.2 An output-sensitive quantum algorithm for MW

By adding as a preprocessing a known output-sensitive quantum algorithm for the Boolean product of the matrices $A$ and $B$, Algorithm 1 can be transformed into an output-sensitive one.
Algorithm 2
Input: oracles corresponding to Boolean $n \times n$ matrices $A$, $B$.
Output: maximum witnesses for all non-zero entries of the Boolean product of $A$ and $B$.

1. Run an output-sensitive quantum algorithm for the Boolean product $C$ of $A$ and $B$.
2. for all non-zero entries $C[i,j]$ do
   $\text{MaxWit}(A, B, i, j)$

Theorem 2. The MW problem can be solved by a quantum algorithm in $\tilde{O}(n \sqrt{s} + s \sqrt{n})$ time, where $s$ is the number of non-zero entries in the product.

Proof. Consider Algorithm 2. Due to Step 1, the procedure $\text{MaxWit}$ is called only for non-zero entries of $C$. Hence, the total time taken by Step 2 is $\tilde{O}(s \sqrt{n})$ by Lemma 1 with any fixed $\beta$. It is sufficient now to plug in the output-sensitive quantum algorithm for Boolean matrix product due to Le Gall [15] running in $\tilde{O}(n \sqrt{s} + s \sqrt{n})$ time to implement Step 1. In order to obtain enough large probability of the correctness of the whole output, we can iterate the plug in algorithm a logarithmic number of times and pick enough large $\beta$ in Lemma 1. We obtain the output-sensitive upper bound claimed in the theorem. \[\square\]

Interestingly enough, the asymptotic time complexity of our output-sensitive quantum algorithm for MW coincides with that of the output-sensitive quantum algorithm for Boolean matrix product due Le Gall [15,16].

3.3 An input-sensitive quantum algorithm for MW

We can also refine the straightforward quantum algorithm for MW in order to obtain an input-sensitive quantum algorithm for MW.

Algorithm 3
Input: Boolean $n \times n$ matrices $A$, $B$.
Output: maximum witnesses for all non-zero entries of the Boolean product of $A$ and $B$ and “No” for all zero entries of the product.

1. For each column $j$ of the matrix $B$ compute the sequence $K_j$ of indices $k \in [n]$ in decreasing order such that $B[k,j] = 1$ by using the oracle for the matrix $B$. Construct a one dimensional integer table $S_j$ of length $|K_j|$ such that for $s \in [|K_j|]$, $S_j[s]$ is the $s$-th largest element in $K_j$.
2. for all $i$, $j \in [n]$ do
(a) Define an oracle for a virtual, one-dimensional integer table $T_{i,j}$ of length $|K_j|$ such that for $s \in [|K_j|]$, $T_{i,j}[s] = 2n - A[i, S_j[s]]n - S_j[s]$. (The value of $T_{i,j}[s]$ can be retrieved in constant time by using the oracle for the matrix $A$ and the table $S_j$.)

(b) Iterate $O(\log n)$ times the algorithm of Dürr and Høyer for $T_{i,j}$ and set $s'$ to the index minimizing $T_{i,j}$.

(c) If $T_{i,j}[s'] < n$ then return $S_j[s']$ (i.e., $n - T_{i,j}[s']$) as the maximum witness for $C[i,j]$ otherwise return “No” for $C[i,j]$.

An analysis of Algorithm 1 yields the following theorem.

**Theorem 3.** The MW problem for the Boolean product of two Boolean $n \times n$ matrices, with $m_1$ and $m_2$ non-zero entries respectively, admits a quantum algorithm running in $\tilde{O}(n^2 + n^{1.5} \sqrt{\min\{m_1, m_2\}})$ time.

**Proof.** Consider Algorithm 3. Its correctness follows from the definition of the tables $T_{i,j}$, in particular the fact that each of them has distinct values. Let us estimate the time complexity of Algorithm 3. We may assume w.l.o.g. that the number of non-zero entries in the matrix $B$ is $m_2$. Steps 1, 2(a) and 2(c) can be easily done in $\tilde{O}(n^2)$ total time. In Step 2(b), computing the maximum witnesses for the entries in the $i$-th row of the product matrix takes $\tilde{O}(\sum_{j=1}^{n} \sqrt{|K_j|})$ time by Fact[3]. Since $\sum_{j=1}^{n} |K_j| \leq m_2$ and the arithmetic mean does not exceed the quadratic one, we obtain $\sum_{j=1}^{n} \sqrt{|K_j|} \leq n \sqrt{\frac{m_2}{n}}$. Consequently, Algorithm 3 runs in $\tilde{O}(n^2 + n^{2} \sqrt{\frac{m_2}{n}})$ time.

As in case of Algorithms 1 and 2, we can pick enough large constant at $\log n$ in the upper bound on the number of iterations of the algorithm of Dürr and Høyer in order to guarantee that the whole output of Algorithm 3 is correct with probability at least $\frac{2}{3}$. Hence, by the time analysis of Algorithm 3 and $A \times B = (B^t \times A^t)^t$, we obtain the theorem. \qed

**4 The fastest method: combining rectangular Boolean matrix multiplication with quantum search**

The best known algorithm for MW from [7] relies on the multiplication of rectangular submatrices of the input matrices. We can combine this idea with that of our procedure $MaxWit$ based on the quantum search for the minimum in order to obtain our fastest quantum algorithm for MW.
Algorithm 4

Input: oracles corresponding to Boolean $n \times n$ matrices $A$, $B$, and a parameter $\ell \in [n]$.

Output: maximum witnesses for all non-zero entries of the Boolean product of $A$ and $B$, and “No” for all zero entries of the product.

1. Divide $A$ into $\lceil n/\ell \rceil$ vertical strip submatrices $A_1, \ldots, A_{\lceil n/\ell \rceil}$ of width $\ell$ with the exception of the last one that can have width $\leq \ell$.
2. Divide $B$ into $\lceil n/\ell \rceil$ horizontal strip submatrices $B_1, \ldots, B_{\lceil n/\ell \rceil}$ of width $\ell$ with the exception of the last one that can have width $\leq \ell$.
3. for $p \in \lceil n/\ell \rceil$ compute the Boolean product $C_p$ of $A_p$ and $B_p$
4. for all $i, j \in [n]$ do
   (a) Find the largest $p$ such that $C_p[i, j] = 1$ or set $p = 0$ if it does not exist.
   (b) if $p > 0$ then return $\ell(p - 1) + \text{MaxWit}(A_p, B_p, i, j)$ else return “No”

---

**Fig. 1.** The input matrices $A$ and $B$ are divided into vertical and horizontal strip submatrices $A_p$ and $B_p$, respectively, in Algorithm 4.

**Fig. 2.** An example illustrating Algorithm 4.
Lemma 2. Algorithm 4 runs in time $\tilde{O}((n/\ell)n^{\omega(1, \log_n \ell, 1)} + n^{3/\ell} + n^2 \sqrt{\ell})$.

**Proof.** Steps 1, 2, take $O(n^2)$ time. Step 3 requires $O((n/\ell)n^{\omega(1, \log_n \ell, 1)})$ time. Step 4(a) takes $O(n^2 \times n/\ell)$ time. Finally, Step 4(b) requires $\tilde{O}(n^2 \sqrt{\ell})$ time totally by Lemma 1. □

By Lemma 1 with sufficiently large $\beta$ and the time analysis in Lemma 2, we obtain the following trade-offs between preprocessing time and answering a maximum witness query time depending on $\ell$.

**Theorem 4.** Let $C$ denote the Boolean product of two Boolean $n \times n$ matrices, and let $i, j$ be any integers in $[n]$. Without any preprocessing, a maximum witness query for $C[i, j]$ can be answered in $\tilde{O}(\sqrt{n})$ time. Let $\ell$ be a parameter in $[n]$. After an $O((n/\ell)n^{\omega(1, \log_n \ell, 1)})$ time preprocessing (Steps 1, 2, 3 in Algorithm 4), a maximum witness query for $C[i, j]$ can be answered in $\tilde{O}(n/\ell + \sqrt{\ell})$ time.

After an $O((n/\ell)n^{\omega(1, \log_n \ell, 1)} + n^3/\ell)$ time preprocessing (Steps 1, 2, 3, 4(a) in Algorithm 4), a maximum witness query for $C[i, j]$ can be answered in $\tilde{O}(\sqrt{\ell})$ time. Finally, after running the whole Algorithm 4 in time $\tilde{O}((n/\ell)n^{\omega(1, \log_n \ell, 1)} + n^3/\ell + n^2 \sqrt{\ell})$, a maximum witness query for $C[i, j]$ can be answered in $O(1)$ time.

### 4.1 Finding $\ell$ minimizing the total time.

Recall that $\omega(1, r, 1)$ denotes the exponent of the multiplication of an $n \times n^r$ matrix by an $n^r \times n$ matrix. By Lemma 2, the total time taken by Algorithm 4 for maximum witnesses is

$$\tilde{O}((n/\ell) \cdot n^{\omega(1, \log_n \ell, 1)} + n^3/\ell + n^2 \sqrt{\ell}).$$

By setting $r$ to $\log_n \ell$ our upper bound transforms to $\tilde{O}(n^{1-r+\omega(1, r, 1)} + n^{3-r} + n^{2+r/2})$. Note that by assuming $r \geq \frac{2}{3}$, we can get rid of the additive $n^{3-r}$ term. Hence, by solving the equation $1 - \lambda + \omega(1, \lambda, 1) = 2 + \lambda/2$ implying $\lambda \geq \frac{2}{3}$ by $\omega(1, \lambda, 1) \geq 2$ and setting sufficiently large $\beta$ in Lemma 1, we obtain our main result.

**Theorem 5.** Let $\lambda$ be such that $\omega(1, \lambda, 1) = 1 + 1.5 \lambda$. The maximum witnesses for all non-zero entries of the Boolean product of two $n \times n$ Boolean matrices can be computed almost certainly by a quantum algorithm in $\tilde{O}(n^{2+\lambda/2})$ time.

Note that by Fact 1, the solution $\lambda$ of the equation $\omega(1, \lambda, 1) = 1 + 1.5 \lambda$ is satisfied by $\lambda = \frac{1+o(1)}{15(1-o)(\omega-2)} + o(1)$. Note also that $\lambda$ is increasing in $\omega$ and decreasing in $\alpha$. Hence, the inequality $\lambda < 0.8671$ holds by Fact 1 and Fact 3. We obtain the following concrete corollary.
Corollary 1. The maximum witnesses for all non-zero entries of the Boolean product of two $n \times n$ Boolean matrices can be computed almost certainly by a quantum algorithm in $\tilde{O}(n^{2.4335})$ time.

5 Applications of quantum algorithms for MW

The problem of finding a lowest common ancestor (LCA) in a tree, or more generally, in a directed acyclic graph (dag) is a basic problem in algorithmic graph theory. A LCA of vertices $u$ and $v$ in a dag is an ancestor of both $u$ and $v$ that has no descendant which is an ancestor of $u$ and $v$, see Fig. 3. We consider the problem of preprocessing a dag such that LCA queries can be answered quickly for any pair of vertices. The all-pairs LCA problem is to compute LCA for all pairs of vertices in the input dag. In the proof of Theorem 11 in [7], on the basis of an input $n$-vertex dag, a Boolean $n \times n$ matrix $A$ is constructed in $O(n^\omega)$ time such that the maximum witness for $C[i, j]$, where $C = A \times A^t$ yields an LCA for vertices $i, j$ in the dag. Combining this with Corollary 1, we obtain also the following theorem.

Theorem 6. The all-pairs LCA problem can be solved by a quantum algorithm in $O(n^{2.4335})$ time.

Very recently, Grandoni et al. have presented an $\tilde{O}(n^{2.447})$-time algorithm for the LCA problem in the standard computational model [10].

Shapiro et al. considered the following all-pairs bottleneck weight path problem in directed, vertex weighted graphs in [20]. Let $G = (V, E)$ be a directed, vertex-weighted graph. The bottleneck weight of a directed path in $G$ is the minimum weight of a vertex on the path. For two vertices $u, v$ of $G$, the bottleneck

Fig. 3. An example of a dag. Note that the vertices 5 and 6 are LCA for the vertices 2, 3, and the vertex 6 is also an LCA for the vertices 1, 3 but it is not an LCA for the vertices 1, 2.
weight from \( u \) to \( v \) is the maximum bottleneck weight of a directed path from \( u \) to \( v \) in \( G \). The all-pairs bottleneck paths problem (APBP) is to find bottleneck weights for all ordered pairs of vertices in \( G \). The authors of [20] considered two variants of APBP, an open variant where the weights of the start and end vertices are not counted and a closed variant where the weights of the start and end vertices are counted. In particular, in Theorem 2 in [20], they show that both variants of APBP, MW, and the problem of computing maximum weight of two-edge paths between all pairs of vertices in vertex weighted graphs are computationally equivalent (up to constant factors). Hence, by Corollary [1] we obtain the following theorem.

**Theorem 7.** The following problems admit an \( \tilde{O}(n^{2.434}) \)-time quantum algorithm: Open APBP, Closed APBP, the all-pairs maximum weight two-edge paths in vertex weighted graphs.

As a corollary, we obtain a faster quantum algorithm for the problem considered in [21].

**Corollary 2.** Let \( G \) be an undirected vertex-weighted graph on \( n \) vertices. The problem of finding for each edge \( \{u, v\} \) of \( G \), a heaviest (or, lightest) triangle \( \{u, v, w\} \) in \( G \) admits a quantum algorithm running in \( O(n^{2.434}) \) time.

### 6 Approximation algorithms

In this section, we present two approximation approaches to MW in a standard computational model. The first approach follows the idea of the fastest known algorithm for MW [7] but instead of searching the final index intervals where the respective maximum witnesses are localized some witnesses from the intervals are reported. The second approach relies on the repetitive applying the deterministic algorithm for multiple witnesses from [9] and the goodness of its approximation for a matrix product entry depends on the number of witnesses for the entry.

#### 6.1 The method based on rectangular matrix multiplication

By slightly modifying the algorithm for MW [7] (or, the quantum Algorithm 4) based on fast rectangular multiplication, we can obtain a faster approximation algorithm. For a given \( \ell \), it reports for each non-zero entry of the Boolean matrix product a witness of rank not exceeding \( \ell \) instead of the maximum witness. In the time analysis of the approximation algorithm, we rely on the fact that witnesses for non-zero entries of the Boolean product of two Boolean matrices
can be reported in time proportional to the time taken by fast Boolean matrix
multiplication up to polylogarithmic factors (see Fact 2).

**Algorithm 5**

*Input:* Boolean $n \times n$ matrices $A$, $B$, and a parameter $\ell \in [n]$.

*Output:* witnesses for all non-zero entries of the Boolean product of $A$ and $B$
having rank not exceeding $\ell$ and “No” for all zero entries of the product.

1. Divide $A$ into $\lceil n/\ell \rceil$ vertical strip submatrices $A_1, ..., A_{\lceil n/\ell \rceil}$ of width $\ell$
with the exception of the last one that can have width $\leq \ell$.

2. Divide $B$ into $\lceil n/\ell \rceil$ horizontal strip submatrices $B_1, ..., B_{\lceil n/\ell \rceil}$ of width $\ell$
with the exception of the last one that can have width $\leq \ell$.

3. for $p \in \lceil [n/\ell] \rceil$ do
   Compute the Boolean product $C_p$ of $A_p$ and $B_p$ along with single witnesses
   for all positive entries of the product

4. for all $i, j \in [n]$ do
   (a) Find the largest $p$ such that $C_p[i, j] = 1$ or set $p = 0$ if it does not exist.
   (b) if $p > 0$ then return the found witness of $C_p[i, j]$ else return “No”

**Lemma 3.** Algorithm 5 runs in time $\tilde{O}((n/\ell)n^{\omega(1, \log n, \ell, 1)})$.

**Proof.** Steps 1, 2, take $O(n^2)$ time. Step 3 requires $\tilde{O}((n/\ell)n^{\omega(1, \log n, \ell, 1)})$ time
by a straightforward generalization of the $\tilde{O}(n^\omega)$-time algorithmic solution to
the witness problem for square Boolean matrices given in Fact 2 to include
rectangular Boolean matrices. Step 4(a) takes $O(n^2 \times n/\ell)$ time totally. Fi-
nally, Step 4(b) requires $O(n^2)$ time totally. It remains to observe that the term
$\tilde{O}((n/\ell)n^{\omega(1, \log n, \ell, 1)})$ dominates the asymptotic time complexity of the algo-

**Theorem 8.** For all non-zero entries of the Boolean matrix product of two Boolean
$n \times n$ matrices, witnesses of rank not exceeding $\ell$ can be reported in time
$\tilde{O}((n/\ell)n^{\omega(1, \log n, \ell, 1)})$.

**6.2 The method based on multi-witnesses**

A straightforward method to obtain single witnesses of rank $O(\lceil W_{C}(i, j)/k \rceil)$
for the nonzero entries $C[i, j]$ of the Boolean product $C$ of two Boolean $n \times n$
matrices is to iterate a randomized algorithm for single witnesses for the entries
of $C$ [3]. After $O(k \log n)$ iterations such witnesses can be reported with high
probability. This straightforward method takes $\tilde{O}(n^{\omega}k)$ time. We provide a more
efficient algorithm for this problem based on the algorithm for the so called $k$-

**Lemma 3.** Algorithm 5 runs in time $\tilde{O}((n/\ell)n^{\omega(1, \log n, \ell, 1)})$.

**Proof.** Steps 1, 2, take $O(n^2)$ time. Step 3 requires $\tilde{O}((n/\ell)n^{\omega(1, \log n, \ell, 1)})$ time
by a straightforward generalization of the $\tilde{O}(n^\omega)$-time algorithmic solution to
the witness problem for square Boolean matrices given in Fact 2 to include
rectangular Boolean matrices. Step 4(a) takes $O(n^2 \times n/\ell)$ time totally. Fi-
nally, Step 4(b) requires $O(n^2)$ time totally. It remains to observe that the term
$\tilde{O}((n/\ell)n^{\omega(1, \log n, \ell, 1)})$ dominates the asymptotic time complexity of the algo-

**Theorem 8.** For all non-zero entries of the Boolean matrix product of two Boolean
$n \times n$ matrices, witnesses of rank not exceeding $\ell$ can be reported in time
$\tilde{O}((n/\ell)n^{\omega(1, \log n, \ell, 1)})$.

**6.2 The method based on multi-witnesses**

A straightforward method to obtain single witnesses of rank $O(\lceil W_{C}(i, j)/k \rceil)$
for the nonzero entries $C[i, j]$ of the Boolean product $C$ of two Boolean $n \times n$
matrices is to iterate a randomized algorithm for single witnesses for the entries
of $C$ [3]. After $O(k \log n)$ iterations such witnesses can be reported with high
probability. This straightforward method takes $\tilde{O}(n^{\omega}k)$ time. We provide a more
efficient algorithm for this problem based on the algorithm for the so called $k$-

**Lemma 3.** Algorithm 5 runs in time $\tilde{O}((n/\ell)n^{\omega(1, \log n, \ell, 1)})$.

**Proof.** Steps 1, 2, take $O(n^2)$ time. Step 3 requires $\tilde{O}((n/\ell)n^{\omega(1, \log n, \ell, 1)})$ time
by a straightforward generalization of the $\tilde{O}(n^\omega)$-time algorithmic solution to
the witness problem for square Boolean matrices given in Fact 2 to include
rectangular Boolean matrices. Step 4(a) takes $O(n^2 \times n/\ell)$ time totally. Fi-
nally, Step 4(b) requires $O(n^2)$ time totally. It remains to observe that the term
$\tilde{O}((n/\ell)n^{\omega(1, \log n, \ell, 1)})$ dominates the asymptotic time complexity of the algo-

**Theorem 8.** For all non-zero entries of the Boolean matrix product of two Boolean
$n \times n$ matrices, witnesses of rank not exceeding $\ell$ can be reported in time
$\tilde{O}((n/\ell)n^{\omega(1, \log n, \ell, 1)})$.
The \textit{k-witness problem} for the Boolean matrix product of two $n \times n$ Boolean matrices is to produce a list of $r$ witnesses for each positive entry of the product, where $r$ is the minimum of $k$ and the total number of witnesses for this entry.

In the following fact from \cite{9}, the upper bounds have been updated by incorporating the more recent results on the parameters $\omega$ (Fact 1) and $\alpha$ [17].

\textbf{Fact 6} \cite{9} There is a randomized algorithm solving the \textit{k-witness problem} almost certainly in time $\tilde{O}(n^{2+o(1)}k + n^{\omega_k(3-\omega-\alpha)/(1-\alpha)})$, where $\alpha \approx 0.31389$ (see Fact 3). One can rewrite the upper time bound as $\tilde{O}(n^\omega k^\mu + n^{2+o(1)}k)$, where $\mu \approx 0.46530$.

\textbf{Algorithm 6}

\textit{Input:} Boolean $n \times n$ matrices $A$, $B$, and a parameter $k \in [n]$ not less than 4.

\textit{Output:} single witnesses $\text{Wit}[i,j]$ for all non-zero entries $C[i,j]$ of the Boolean product $C$ of $A$ and $B$ such that $\text{rank}(\text{Wit}[i,j]) \leq 4\lceil W_C(i,j)/k \rceil$ with probability at least $\frac{1}{2} - e^{-1}$.

1. $D \leftarrow B$
2. Initialize $n \times n$ integer matrix $\text{Wit}$ by setting all its entries to 0.
3. \textbf{for} $q = 1, \ldots, O(\log n)$ \textbf{do}
   (a) Run an algorithm for the \textit{k-witness problem} for the product $F$ of the matrices $A$ and $D$.
   (b) For all $1 \leq i, j \leq n$, set $\text{Wit}[i,j]$ to the maximum of $\text{Wit}[i,j]$ and the maximum among the reported witnesses for $F[i,j]$.
   (c) Uniformly at random set each entry of $D$ to zero with probability $\frac{1}{2}$.

$TW(n, k)$ will stand for the running time of the \textit{k-witness algorithm} for the Boolean product of the two input Boolean matrices of size $n \times n$ used in Algorithm 6.

\textbf{Lemma 4}. Algorithm 6 runs in $\tilde{O}(TW(n, k) + n^2k)$ time.

\textit{Proof.} The block of the while loop can be implemented in $O(TW(n, k) + n^2k)$ time. It is sufficient to observe that the block is iterated $O(\log n)$ times. \hfill $\Box$

\textbf{Lemma 5}. For $1 \leq i, j \leq n$ and $k \geq 4$, the final value of $\text{Wit}[i,j]$ in Algorithm 5 is a witness of $C[i,j]$ with rank at most $4\lceil W_C(i,j)/k \rceil$ with probability not less than $\frac{1}{2} - e^{-1}$.

\textit{Proof.} We may assume without loss of generality that $W_C(i,j)/k > 1$ since otherwise the maximum witness for $C[i,j]$ is found already in the first iteration of the block of the while loop. Let $\ell = \lceil \log_2 W_C(i,j)/k \rceil$. A witness of the entry $C[i,j]$ survives $\ell + 1$ iterations of the block of the while loop.
with probability $2^{-\ell-1}$. Hence, after $\ell + 1$ iterations of the block of the while loop the expected number of witnesses of the entry $C[i, j]$ that survive does not exceed $k/2$. Consequently, the number of witnesses of $C[i, j]$ that survive does not exceed $k$ with probability at least $1/2$. They are reported as witnesses of $F[i, j]$ in the $\ell + 2$ iteration. On the other hand, the probability that none of witnesses not greater than $4W_C(i, j)/k$ survives the $\ell + 1$ iterations is at most $(1 - \frac{1}{2e^2})^{4W_C(i, j)/k} \leq e^{-1}$ by $k \geq 4$.

Observe that for events $A$ and $B$, \(\text{Prob}(A \cap B) \geq 1 - \text{Prob}(\bar{A} \cup \bar{B}) \geq 1 - \text{Prob}(\bar{A}) - \text{Prob}(\bar{B})\). Hence, at least one witness of rank at most $4W_C(i, j)/k$ survives $\ell + 1$ iterations and it is reported in the $\ell + 2$ iteration with probability at least $1 - \frac{1}{2} - e^{-1} \geq \frac{1}{2} - e^{-1}$.  

\[\text{Theorem 9. Let } C \text{ be the Boolean product of two Boolean } n \times n \text{ matrices and let } k \text{ be an integer not less than 4. One can compute for all non-zero entries } C[i, j] \text{ single witnesses of rank } O(\lceil W_C(i, j)/k \rceil) \text{ in } \tilde{O}(n^{\omega(k, 0.4653 + n^2 + o(1)k)}) \text{ time almost certainly.}\]

\[\text{Proof. By Lemma 5 it is sufficient to iterate Algorithm 5 } O(\log n) \text{ times to achieve the probability of at least } 1 - n^{-\beta}, \beta \geq 1. \text{ The time complexity bound follows from Lemma 4 by the upper bound on } TW(n, k) \text{ from Fact 6}.\]

By plugging the randomized upper bound on $CW(n, k)$ from Fact 2 into Theorem 9 and assuming the notation from the theorem, we obtain the following corollary.

\[\text{Corollary 3. There is a randomized algorithm that for } 4 \leq k \leq n^{0.4212} \text{ computes for all non-zero entries } C[i, j] \text{ single witnesses of rank } O(\lceil W_C(i, j)/k \rceil) \text{ almost certainly in time substantially subsuming the best known upper time bound for computing maximum witnesses for all non-zero entries of } C. \text{ In particular, if the number of witnesses for each entry of } C \text{ is upper bounded by } w \leq n^{0.4212} \text{ then by setting } k = w, \text{ we obtain for all non-zero entries of } C \text{ a witness of rank } O(1) \text{ almost certainly, substantially faster than maximum witnesses for these entries.}\]

7 Final remarks

Due to the quantum search for the minimum, the MW problem is relatively easier in the quantum computation model. Already the straightforward quantum algorithm (Algorithm 1) running in $\tilde{O}(n^{2.5})$ time, is substantially faster than the best known algorithm for MW in the standard model running in $O(n^{2.569})$ time (originally, $O(n^{2.575})$ time [7]). Also, the gap between our fastest algorithm for MW (Algorithm 4) running in $O(n^{2.434})$ time and the fastest algorithm
for Boolean matrix product in the quantum computation model is substantially smaller than the corresponding gap in the standard model. An additional reason here is that no quantum algorithm for Boolean matrix product in general case faster than the algebraic ones in the standard model is known so far.

Our input-sensitive quantum algorithm for MW (Algorithm 3) is faster than our $O(n^{2.434})$-time algorithm for MW (Algorithm 4) when one of the input $n \times n$ matrices has a number of non-zero entries substantially smaller than $O(n^{1.868})$. Similarly, our output-sensitive quantum algorithm for MW (Algorithm 2) is faster than the $O(n^{2.434})$-time algorithm for MW when the number $s$ of non-zero entries in the product matrix is substantially smaller than $O(n^{1.934})$.

Our approximation algorithm for MW could be used for example to find triangles passing through specified edges approximating heaviest ones in vertex weighted graph (cf. [21]).

Acknowledgments

The authors thank Francois Le Gall for a useful clarification of the current status of quantum algorithms for Boolean matrix product. The research has been supported in part by Swedish Research Council grant 621-2017-03750.

References

1. Alman, J., Vassilevska Williams, V.: A Refined Laser Method and Faster Matrix Multiplication. Proc. SODA, pp. 522–539, 2021.
2. Alon, N., Galil, Z., Margalit, O., Naor, M.: Witnesses for Boolean matrix multiplication and for shortest paths. In: Proc. 33rd Symposium on Foundations of Computer Science (FOCS), pp. 417–426 (1992)
3. Alon, N., Naor, M.: Derandomization, witnesses for Boolean matrix multiplication and construction of perfect hash functions. Algorithmica 16, 434–449 (1996)
4. Ambainis, A.: Quantum search algorithms. SIGACT News, 35 (2), pp. 22–35, 2004.
5. Coppersmith, D.: Rectangular matrix multiplication revisited. Journal of Symbolic Computation, 1, pp. 42–49, 1997.
6. Cohen, K. Yuster, R.: On minimum witnesses for Boolean matrix multiplication. Algorithmica 69(2), 431–442 (2014)
7. Czumaj, A., Kowaluk, M., Lingas, A.: Faster algorithms for finding lowest common ancestors in directed acyclic graphs. Theoretical Computer Science, 380(1-2), 37–46 (2007)
8. Dür, C., and Høyer, P.: A quantum algorithm for finding the minimum. In: arXiv: 9607.014, 1996/99.
9. Gąsieniec, L., Kowaluk, M., Lingas, A.: Faster multi-witnesses for Boolean matrix product. Information Processing Letters 109, pp. 242–247 (2009)
10. Grandoni, F., Italiano, G.F., Łukasiewicz, A, Parotsidis, N., Uznański, P.: All-Pairs LCA in DAGs: Breaking through the $O(n^{2.5})$ barrier. To appear in proc. SODA 2021, CoRR abs/2007.08914 (2020)
11. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proc. Annual ACM Symposium on Theory of Computing (STOC), pp. 212-219 (1996)
12. Huang, X., and Pan, V.Y.: Fast rectangular matrix multiplications and applications. Journal of Complexity, 14, pp. 257–299, 1998.
13. Kowaluk, M. and Lingas, A.: Quantum and approximation algorithms for maximum witnesses of Boolean matrix products. CoRR abs/2004.14064 (2020)
14. Le Gall, F.: Powers of Tensors and Fast Matrix Multiplication. In: Proc. 39th International Symposium on Symbolic and Algebraic Computation, pp. 296–303 (2014)
15. Le Gall, F.: A Time-Efficient Output-Sensitive Quantum Algorithm for Boolean Matrix Multiplication. In: Proc. ISAAC 2012, pp. 639–648 (2012)
16. Le Gall, F.: Improved output-sensitive quantum algorithms for Boolean matrix multiplication. In: Proc. SODA 2012, pp. 1464–1476 (2012)
17. Le Gall, F. and Urrutia, F.: Improved Rectangular Matrix Multiplication using Powers of the Coppersmith-Winograd Tensor. In: Proc. SODA 2018, pp. 1029–1046 (2018)
18. Navebi, A. and Vassilevska Williams, V.: Quantum algorithms for shortest path problems in structured instances. In: arXiv:1410.6220 (2014).
19. Nielsen, M., and Chuang, I.: Quantum Computation and Quantum Information. Cambridge University Press, 2000.
20. Shapira, A., Yuster, R., Zwick, U.: All-Pairs Bottleneck Paths in Vertex Weighted Graphs. Algorithmica 59, 621–633 (2011)
21. Vassilevska, V., Williams, R., Yuster, R.: Finding heaviest H-subgraphs in real weighted graphs, with applications. ACM Transactions on Algorithms 6(3), 44:1–44:23 (2010)
22. Vassilevska Williams, V.: Multiplying matrices faster than Coppersmith-Winograd. In: Proc. 44th Annual ACM Symposium on Theory of Computing (STOC), pp. 887–898 (2012)