A Fast Output-Sensitive Algorithm for Boolean Matrix Multiplication*

Andrzej Lingas

Department of Computer Science, Lund University, 22100 Lund, Sweden
Andrzej.Lingas@cs.lth.se

Abstract. We use randomness to exploit the potential sparsity of the Boolean matrix product in order to speed up the computation of the product. Our new fast output-sensitive algorithm for Boolean matrix product and its witnesses is randomized and provides the Boolean product and its witnesses almost certainly. Its worst-case time performance is expressed in terms of the input size and the number of non-zero entries of the product matrix. It runs in time $\widetilde{O}(n^2 s^{\omega/2-1})$, where the input matrices have size $n \times n$, the number of non-zero entries in the product matrix is at most s, ω is the exponent of the fast matrix multiplication and $\widetilde{O}(f(n))$ denotes $O(f(n)\log^d n)$ for some constant d. By the currently best bound on ω, its running time can be also expressed as $\widetilde{O}(n^2 s^{0.188})$. Our algorithm is substantially faster than the output-sensitive column-row method for Boolean matrix product for s larger than $n^{1.232}$ and it is never slower than the fast $\widetilde{O}(n^\omega)$-time algorithm for this problem.

We also present a partial derandomization of our algorithm as well as its generalization to include the Boolean product of rectangular Boolean matrices. Finally, we show several applications of our output-sensitive algorithms.

1 Introduction

Boolean matrix multiplication is a basic tool in the design of efficient algorithms, especially graph algorithms (see, e.g., [5,17,18]). Just squaring a Boolean matrix is equivalent to the following fundamental problem: for n subsets of $\{1, 2, ..., n\}$, determine all pairs of the subsets that have a non-empty intersection. This problem has numerous applications, among other things, in databases and data mining [2].

By the standard definition of Boolean matrix multiplication, it is sufficient to use $O(n^3)$ conjunctions and disjunctions to compute the Boolean product of two Boolean $n \times n$ matrices. Slightly less known is another simple combinatorial way of computing the Boolean product termed as a *column-row* method [21].

Consider two Boolean $n \times n$ matrices A and B, and their Boolean product C. If $C[i, j] = 1$ then any index k such that $A[i, k] = 1$ and $B[k, j] = 1$ is called a *witness* for $C[i, j]$. In the column-row approach, for each k, one considers the set of indices of rows in A that have 1 in the k-th column of A as well as the set of indices of columns in B that have 1 in the k-th row of B. Note that for any i in the former set and any j in the latter one, k is a witness for $C[i, j]$. It follows that the column-row method can

* Research supported in part by the VR grant 621-2005-4085.

A. Fiat and P. Sanders (Eds.): ESA 2009, LNCS 5757, pp. 408–419, 2009.

be implemented in time $O(W + n^2)$ where W is the total number of witnesses for all non-zero entries of C [21].

The column-row approach is output sensitive in terms of the number s of non-zero entries of the product matrix. Simply, each non-zero entry can have at most n witnesses and therefore the total number of witnesses cannot exceed ns. In this way, we obtain the output sensitive upper time-bound of $O(ns + n^2)$ for Boolean square matrix multiplication.

This upper bound is still attractive in comparison with the running time of the fastest known combinatorial algorithms for Boolean matrix multiplication, which is $O(n^3/\log^2 n)$ (see [3,5,18]). However, it seems less attractive when compared with the running time of the fastest known algorithm for Boolean matrix multiplication. The latter is simply obtained by treating the Boolean 0 and 1 as arithmetic ones and using the fast arithmetic matrix multiplication. Thus, its running time is $O(n^\omega)$, where ω is the smallest constant for which the product of two $n \times n$ arithmetic matrices can be computed in time $O(n^\omega)$. The best currently known asymptotic upper bound on ω is 2.376, due to Coppersmith and Winograd [9]. Thus, the fast algorithm for Boolean matrix multiplication subsumes the output-sensitive column-row method whenever s is substantially larger than $n^{1.376}$.

In the mathematical programming literature, the assumption of *no cancellation*, i.e., that the inner product of two vectors with overlapping non-zero entries never cancels to zero is very common [7]. The justification of the assumption comes from the unlikeness of cancellations and the aggregation of rounding errors. For this reason, the important problem of the structure prediction of the product of two arithmetic matrices reduces to that of the Boolean product of the two corresponding Boolean matrices, where the Boolean 1 entries correspond to the non-zero ones.

The prediction of the structure of the product of two matrices is important both in efficient memory block allocation as well as in determining an optimal order of chains of matrix products [7,12]. Cohen provided an efficient procedure for close estimation of the size of rows and columns in the Boolean product [7]. The problem of more precise efficient prediction of the structure of arithmetic matrix product has remained open.

Of course one can simply compute the Boolean product of the corresponding Boolean matrices. Unfortunately, the fast method for Boolean matrix multiplication takes no fewer operations than the multiplication of the original arithmetic matrices itself, i.e., $O(n^\omega)$. What about the situation when the Boolean matrix product is moderately dense or sparse, i.e., it has a substantially sub-quadratic and substantially super-linear in n number of non-zero entries?

Very recently, Amossen and Pagh [2], extending the input sensitive algorithm of Yuster and Zwick [22], proposed an input- and output-sensitive algorithm for Boolean matrix product running in time $O(t^{0.86}s^{0.41} + (ts)^{2/3})$, where t stands for the number of non-zero entries in the input matrices and s for the number of non-zero entries in the output matrix. While their upper time-bound is very interesting, it can easily exceed n^ω, when the input matrices are quite dense and the ouput one is moderately dense.

Summarizing, a natural question arises if there is a fast output-sensitive algorithm for Boolean matrix multiplication, whose running time is substantially better than that

of the output-sensitive column-row method, and than $O(n^\omega)$ when the product matrix is moderately dense/sparse?

We answer this question in the affirmative by providing a randomized algorithm for Boolean matrix product and its witnesses running in time $\widetilde{O}(n^2 s^{\omega/2-1})$.

Thus, using the upper bound of Coppersmith-Winograd on ω, our algorithm runs in time $\widetilde{O}(n^2 s^{0.188})$. Since $n^2 s^{\omega/2-1} \leq n^\omega$ for all $s \leq n^2$, our algorithm is never slower than that of Coppersmith-Winograd and it is faster than the column-row method for $s > n^{1/(2-w/2)} \approx n^{1.232}$.

It is not an easy task to compare our algorithm with the algorithm of Amossen and Pagh which is also input sensitive. For instance, for moderately dense/sparse random input Boolean matrices, the Boolean product might be expected to be already dense. Then the input sensitive algorithms of Yuster and Zwick, and of Amossen and Pagh will be superior. *However, the sparsity of the product matrix does not imply the sparsity of any of the input matrices generally.* For instance, if only the first halves of rows in the first input matrix are filled with ones and similarly only the second halves of columns in the other input matrix are filled with ones, then we have $t = n^2$ and $s = 0$. Assuming a worst-case scenario for input sensitivity, i.e., $t = \Omega(n^2)$, our algorithm is faster than that of Amossen and Pagh for $s > n^{1.244}$.

The use of randomness to exploit the potential sparsity of the Boolean product is crucial in the design of our algorithm. However, if upper bounds on the number of non-zero entries in the respective rows and columns of the product matrix summing to at most $2s$ are given *a priori* then we can partially derandomize our algorithm so it uses only $O(\log^2 n)$ random bits and runs in the same asymptotic time. In both cases, our algorithms provide the Boolean matrix products and their witnesses almost certainly. We also present generalizations of our algorithms to include the Boolean product of rectangular Boolean matrices.

Finally, we show applications of our algorithms and its generalizations to the derivation of output-sensitive upper time-bounds for the following problems: composition of binary relations, maximum witnesses of Boolean matrix product, lowest common ancestors in directed acyclic graphs and prediction of the structure of arithmetic matrix product.

1.1 Other Related Results

The column-row approach to (non-necessarily Boolean) matrix multiplication has been known for a long time. In the arithmetic case, it can be expressed as the fact that the matrix product of two $n \times n$ matrices A and B is equal to the sum of the outer products of the i-th column of A with the i-th row of B, over $i = 1, ..., n$ [7,13]. Thus, the column-row approach takes $O(\sum_{k=1}^{n} a_k b_k)$ multiplications and additions, where a_k is the number of non-zero elements in the k-th column of A and b_k is the number of elements in the k-th row of B. Additionally, an $O(n^2)$ time is sufficient to compute the list of non-zero elements for each column of A and each row of B. Hence, under the no cancellation assumption the $O(ns+n^2)$ time-bound, where s is the number of non-zero entries in the product, easily follows in the arithmetic case.

Sparse arithmetic matrices, and hence also, sparse Boolean matrices, are well known to admit more efficient multiplication algorithms than those aforementioned for the

general case, see [7,22]. Note that if A or B has at most m non-zero entries then by $\sum_{k=1}^{n} a_k b_k \leq \max\{(\sum_{k=1}^{n} a_k)n, (\sum_{k=1}^{n} b_k)n\} \leq mn$, one obtains an $O(mn + n^2)$ upper bound for sparse matrix multiplication by the column-row approach [22]. For the case where both A and B have at most m non-zero entries, Yuster and Zwick provide an algorithm using $O(m^{0.7}n^{1.2} + n^{2+o(1)})$ multiplications, additions and subtractions over R [22]. Their algorithm consists in a nice and simple reduction to the sum of two rectangular matrix multiplications, the first fast one corresponding to the indices that are witnesses for relatively many entries and the other column-row one corresponding to the indices that are witnesses for relatively few entries.

One of the drawbacks of the fast Boolean matrix multiplication is that it does not yield explicitly witnesses for the non-zero entries of the product matrix in contrast with the aforementioned combinatorial methods. However, at the beginning of the 90s, efficient randomized and deterministic methods for Boolean product witnesses have been developed [4,11,20]. They typically provide a single witness per non-zero entry and run in time $\widetilde{O}(n^{\omega})$.

1.2 Organization

In the next section, we present our fast output-sensitive randomized algorithm for the Boolean product of two square Boolean matrices and its analysis. In Section 3, we discuss its partial derandomization. In section 4, we outline a generalization of our algorithm and its analysis to include rectangular input Boolean matrices. In section 5, we show a number of applications of our output-sensitive algorithms for Boolean matrix product.

2 The Output-Sensitive Algorithm for Boolean Product

The following definition and two lemmata lie behind the idea of our algorithm.

Definition 1. *Let C be an $n \times n$ Boolean matrix. A row of C is* r-sparse *if the number of non-zero entries in it is at most r, and otherwise, the row is* r-dense. *Similarly, a column of C is r-sparse if the number of non-zero entries in it is at most r, and otherwise, the column is* r-dense. *An entry of C is r-sparse if both its row and column are r-sparse, otherwise the entry is* r-dense.

Lemma 1. *If an $n \times n$ Boolean matrix C has at most r^2 non-zero entries then the number of r-dense rows and r-dense columns in C is $O(r)$.*

Proof. Otherwise, there would be more than r^2 non-zero entries in C. □

Lemma 2. *Let C be an $n \times n$ Boolean matrix with at most r^2 non-zero entries. Let C' be the matrix resulting from permuting uniformly at random the rows and the columns of C. For a given constant $c > 1$, divide C' into $cr \times cr$ sub-arrays of size $\frac{n}{cr} \times \frac{n}{cr}$. For any given r-sparse non-zero entry $C[i, j]$ of C the probability that another non-zero entry of C is in the same sub-array of C' is $O(\frac{1}{c})$.*

Proof. The probability that another non-zero entry $C[i', j']$ of C, where $i \neq i'$ and $j \neq j'$, is in the same sub-array of C' is at most

$$(r^2 - 1)\frac{(\frac{n}{cr} - 1)}{n-1}\frac{(\frac{n}{cr} - 1)}{n-1} = O(\frac{1}{c^2})$$

The probability that another non-zero entry $C[i', j']$ of C, where either $i = i'$ or $j = j'$, is in the same sub-array of C' is at most

$$2(r-1)\frac{(\frac{n}{cr} - 1)}{n-1} = O(\frac{1}{c})$$

□

Before presenting our algorithm, we recall the concept of witnesses in Boolean matrix multiplication.

Definition 2. *If an entry $C[i, j]$ of the Boolean product of two Boolean matrices A and B is equal to 1 then any index l such that $A[i, l]$ and $B[l, j]$ are equal to 1 is a* ***witness*** *for $C[i, j]$. The problem of* **computing witnesses** *for the product matrix C consists in determining for each non-zero entry of C a single witness.*

Our algorithm consists of two main stages. The first stage takes care of the r-sparse non-zero entries. It consists of iterations of Algorithm 1 which is an efficient algorithmic counterpart of Lemma 2, where the matrix C is the Boolean product of two Boolean matrices A and B. In Algorithm 1, the sub-arrays of C' in Lemma 2 correspond to single entries of a smaller product matrix resulting from gluing batches of $\Omega(\frac{n}{r})$ rows in the row-permuted matrix A and batches of $\Omega(\frac{n}{r})$ columns in the column-permuted matrix B.

Algorithm 1

Input: Two $n \times n$ Boolean matrices A and B whose Boolean product has at most r^2 non-zero entries.

Output: An $n \times n$ Boolean matrix C whose non-zero entries are a subset of the non-zero entries of the Boolean product of A and B, and witnesses for the non-zero entries of C.

1. Permute uniformly at random the rows of A and the columns of B.
2. Contract each block of $\frac{n}{r}$ consecutive rows of row-permuted A into a single super-row by taking the "or" along the $\frac{n}{r}$-long column fragments. Similarly, contract each block of $\frac{n}{r}$ consecutive columns of column permuted B into a single super-column by taking the "or" along the $\frac{n}{r}$-long row fragments.
3. Form the $r \times n$ matrix A^* composed of the super-rows, and similarly form the $n \times r$ matrix B^* composed of the super-columns.
4. Compute the witnesses of the Boolean product C^* of A^* with B^* (one for each non-zero entry of C^*).
5. For each witness k of an entry $C^*[i^*, j^*]$ computed in the previous step, find the first row i' in the block of the row-permuted A corresponding to the i^* (super)row of A^* satisfying $A[i', k] = 1$, as well as the first column j' in the block of column-permuted B corresponding to the j^* (super)column of B^* satisfying $B[k, j'] = 1$.

Let i be the row number of i' in the original matrix A, and similarly, let j be the column number of j' in the original matrix B. Set $C[i,j]$ to 1 and the witness of $C[i,j]$ to k.

Lemma 3. *Algorithm 1 is partially correct, i.e., it sets to* 1 *only non-zero entries of the product matrix and it provides true witnesses (one for each non-zero entry) for them.*

Proof. It is sufficient to prove the correctness of the settings in Step 5. The i^*-th row of A^* is the result of column-wise "or" of the rows $(i^*-1)\frac{n}{r}+1$ through $i^*\frac{n}{r}$. Similarly, the j^*-th column of B^* is the result of row-wise "or" of the columns $(j^*-1)\frac{n}{r}+1$ through $j^*\frac{n}{r}$. Consequently, a witness k for the entry $C^*[i^*,j^*]$ is an index k which satisfies $\bigvee_{i''=(i^*-1)\frac{n}{r}+1}^{i^*\frac{n}{r}} A[i'',k] = 1$ and $\bigvee_{j''=(j^*-1)\frac{n}{r}+1}^{j^*\frac{n}{r}} B[k,j''] = 1$. Hence, the indices i' and j' are well defined in Step 5 and k is also a witness of $C[i,j]$. □

Throughout this section, we shall assume that there is an available method for square $(m \times m)$ Boolean matrix product also producing a witness for each non-zero entry of the product, running in time $f(m)$. Clearly, we have $f(m) \geq m^2$, and we may assume w.l.o.g that $f(qm) \leq q^3 f(m)$ for any positive integer q.

For instance, $f(m)$ could be $O(m^3)$ in case of the straightforward Boolean product method following from the definition, or $\widetilde{O}(n^\omega)$ in case of the method based on the fast arithmetic matrix multiplication and the following fact due to Alon and Naor [4] (see also [20] for an earlier randomized version of this fact).

Fact 1. For all non-zero entries of the Boolean product of two $n \times n$ Boolean matrices representatives of their witnesses can be computed deterministically in total time $\widetilde{O}(n^\omega)$.

Lemma 4. *Algorithm 1 can be implemented in time* $O(\frac{n}{r}f(r)+n^2)$.

Proof. Steps 1,2,3 can be easily implemented in time linear in the input size, i.e., $O(n^2)$. To implement Step 4, we divide vertically A^* into square $r \times r$ sub-arrays A^*_p, $p = 1, ..., \frac{n}{r}$, as well as B^* horizontally into $r \times r$ sub-arrays B^*_p, $p = 1, ..., \frac{n}{r}$. In this way, we reduce the computation of the witnesses for the product C^* of A^* and B^* to the computation of witnesses for the products $A^*_p \times B^*_p$, $p = 1, ..., \frac{n}{r}$, and taking their union. This all takes time $\frac{n}{r} \times O(f(r)+r^2) = O(\frac{n}{r}f(r)+nr)$. Finally, Step 5 takes time $O(\frac{n}{r}r^2) = O(nr)$. □

Lemma 5. *Suppose that the Boolean product C of A and B has at most r^2 non-zero entries. For sufficiently large constant c, after $O(\log n)$ iterations of Algorithm 1, the non-zero r-sparse entries of the Boolean product C will be set to* 1 *and their witnesses will be provided, almost certainly, i.e., with probability not less than* $1 - \frac{1}{n^\alpha}$ *for some* $\alpha > 1$.

Proof. Set c to the smallest positive integer such that the probability bound $O(\frac{1}{c})$ in Lemma 2 is at most $\frac{1}{2}$. It follows then from the specification of Algorithm 1 and Lemma 2 that for a given non-zero r-sparse entry of C the probability that it is not set to 1 (and its witness is not provided) after $d \log n$ iterations is at most $2^{-d \log n}$. Hence, for a given $\alpha > 1$, there is a sufficiently large constant d so the probability that at least one non-zero r-sparse entry is not set to 1 after $d \log n$ iterations is at most $n^{-\alpha}$. □

The second stage of our algorithm takes care of, i.e., provides witnesses for, the non-zero r-dense entries of the Boolean product of A and B. It relies on Lemma 1 and the following fact due to Cohen (see pp. 312-315 in [7]).

Fact 2. *Let A and B be two $n \times n$ Boolean matrices. For any fixed $\epsilon > 0$, there is an $O(n^2 \log n)$-time randomized algorithm that estimates the number of non-zero entries in the columns and rows of the Boolean product of A and B with relative error $< \epsilon$ almost certainly.*

Lemma 6. *The witnesses for a superset of the non-zero r-dense entries of the Boolean product of A and B can be computed in time $O((\frac{n}{r})^2 f(r) + n^2 \log n)$, almost certainly.*

Proof. We can detect a superset of the rows of the matrix A corresponding to the r-dense rows of the product of A and B, as well as the columns of the matrix B corresponding to the r-dense columns of the product matrix as follows. We run the algorithm of Cohen [7] with ϵ set, say to $\frac{1}{2}$, to estimate the number of non-zero entries in each row and each column of the product matrix in time $O(n^2 \log n)$. If the estimation for a row or a column of the product matrix exceeds $\frac{r}{2}$ then we mark the corresponding row of A or the corresponding column of B.

Note that in particular all r-dense rows and columns of the product matrix are marked in this way. On the other hand, each of the marked rows or columns is $\frac{r}{4}$ dense.

Let A^* be the matrix composed of the marked rows of the matrix A. Similarly, let B^* be the matrix composed of the marked rows of B. Now, it is sufficient to determine witnesses for the Boolean products of A^* with B and A with B^*. The union of the two resulting sets of witnesses has to include witnesses for all r-dense non-zero entries of the product of A with B. Since the rows of A^* correspond to some $\frac{r}{4}$ dense rows of the product matrix, A^* has at most $O(r)$ rows by Lemma 1. Analogously, B^* has at most $O(r)$ columns. Hence, by dividing A^* vertically into $O(\frac{n}{r})$ sub-arrays of size $O(r) \times O(r)$ and B both horizontally and vertically into $O((\frac{n}{r})^2)$ sub-arrays of size $O(r) \times O(r)$, we can compute the witnesses of the product of A^* with B by computing the witnesses of $O((\frac{n}{r})^2)$ products of $O(r) \times O(r)$ sub-arrays. This takes time $O((\frac{n}{r})^2 f(r))$. Symmetrically, we can compute the witnesses for the product of A with B^* in time $O((\frac{n}{r})^2 f(r))$. □

We can also use the application of Fact 2 in Lemma 6 to estimate the number r^2 of non-zero entries in the Boolean product matrix for the sake of the iterations of Algorithm 1. Note that if we set $s = r^2$ then the upper bound $O((\frac{n}{r})^2 f(r) + n^2 \log n)$ given in Lemma 6 can be rewritten as $O(\frac{n^2}{s} f(\sqrt{s}) + n^2 \log n) = \widetilde{O}(\frac{n^2}{s} f(\sqrt{s}))$. Similarly the time taken by $O(\log n)$ iterations of Algorithm 1 can be expressed as $\widetilde{O}(\frac{n}{\sqrt{s}} f(\sqrt{s}) + n^2)$ by Lemma 4. Hence, by combining Lemmata 3, 4, 5, 6, we obtain our main theorem.

Theorem 1. *Let A and B be two $n \times n$ Boolean matrices whose Boolean product has at most s non-zero entries. The Boolean product of A and B, and the witnesses for the product can be computed by a randomized algorithm in time $\widetilde{O}(\frac{n^2}{s} f(\sqrt{s}))$, almost certainly. In particular, if we apply the fastest algorithm for the Boolean product and its witnesses, i.e., $f(m) = \widetilde{O}(m^\omega)$, the running time is $\widetilde{O}(n^2 s^{\omega/2-1})$.*

Note that if we apply the straightforward cubic-time algorithm for the Boolean product and its witnesses, i.e., $f(m) = O(m^3)$, we obtain another output-sensitive algorithm for this problem running in time $\widetilde{O}(n^2\sqrt{s})$.

3 Partial Derandomization

Let S_n be the set of all permutations of $0, ..., n-1$. A family of permutations $F \subseteq S_n$ is *pairwise independent* (cf. [6]) if for any π chosen at random in F and any $\{i_1, i_2, k_1, k_2\} \subseteq \{0, ..., n-1\}$ where $i_1 \neq i_2$ and $k_1 \neq k_2$,

$$Pr(\pi(i_1) = k_1 \ \& \ \pi(i_2) = k_2) = \frac{1}{n(n-1)}$$

Assuming that n is a prime number, the family of linear transformations of the form $\pi(i) = ai + b \mod n$, where $a \neq 0$, is known to be pairwise independent (e.g., see Exercise in [16]).

Under the assumption that n is a prime number, we can replace uniform at random permutations in step 1 in Algorithm 1 with such randomly chosen linear transformations. Due to the property of pairwise independence, Lemma 5 can be proved analogously after the replacement. Then, if additionally we use the deterministic version of the algorithm due to Alon and Naor for witnesses of Boolean matrix product [4], one iteration of the so modified Algorithm 1 will require only $O(\log n)$ random bits. Thus, the total number of random bits used by our method, but for the application of Cohen's algorithm to estimate the number of non-zero entries in the rows and columns of the product, can be decreased to $O(\log^2 n)$ at the cost of increasing its time complexity by a poly-logarithmic factor. The increase is caused by patching the input matrices to the size $n' \times n'$, where n' is the smallest prime number not larger than n, and the use of the deterministic algorithm for witnesses [4]. (To find such a prime number n', one can use the AKS primality testing or its improved versions running in time polynomial in the length of bit representation [1].)

Hence, we obtain the following variant of Theorem 1.

Theorem 2. *Let A and B be two $n \times n$ Boolean matrices. Suppose that there are given* a priori *upper bounds on the number of non-zero entries in the respective rows and columns of the Boolean product of A and B summing to at most $2s$. The Boolean product of A and B, and the witnesses for the product can be computed almost certainly by a randomized algorithm, using $O(\log^2 n)$ random bits and $\widetilde{O}(n^2 s^{\omega/2-1})$ time.*

4 Rectangular Boolean Matrix Multiplication

Analogous output sensitive algorithms can be derived for rectangular Boolean matrix multiplication.

Denote by $\omega(p, q, t)$ the exponent of the fast multiplication of an $n^p \times n^q$ matrix by an $n^q \times n^t$ matrix.

We obtain the following partial generalization of Theorems 1, 2.

Theorem 3. *Let A and B be two Boolean matrices of size $n^p \times n^q$ and $n^q \times n^t$, respectively, such that their product has at most $n^{p(1-\delta_1)+t(1-\delta_2)}$ non-zero entries. The Boolean product of A and B and its witnesses can be computed by a randomized algorithm running in time $\widetilde{O}(n^{\omega(|p-\delta_1|,q,t)}+n^{\omega(p,q,|t-\delta_2|)}+n^{p+q}+n^{q+t})$, almost certainly. Furthermore, if upper bounds on the number of non-zero entries in the respective rows and columns of the Boolean product of A and B summing to at most $n^{q(1-\delta_1)+t(1-\delta_2)}$ are known a priori then the algorithm requires only $O(\log^2 n)$ random bits.*

Proof. sketch. Let C denote the $n^p \times n^t$ Boolean product of A and B. Consider $n^{t(1-\delta_2)}$-sparse rows of C, i.e., the rows of C with at most $n^{t(1-\delta_2)}$ non-zero entries as well as $n^{p(1-\delta_1)}$-sparse columns of C, i.e., the columns of C with at most $n^{p(1-\delta_1)}$ non-zero entries. Next, define $(n^{p(1-\delta_1)}, n^{t(1-\delta_2)})$-sparse non-zero entries as those non-zero entries which lie within a $n^{t(1-\delta_2)}$-sparse row and a $n^{p(1-\delta_1)}$-sparse column of C. The remaining entries are $(n^{p(1-\delta_1)}, n^{t(1-\delta_2)})$-dense.

The consecutive steps of the proof are straightforward generalizations of those in the proof of Theorems 1, 2, resulting from the use of the generalized concepts of sparse (and, dense) rows, columns and entries of C. For instance, the generalization of Lemma 1 states that if C has at most $n^{p(1-\delta_1)+t(1-\delta_2)}$ non-zero entries then the number of $n^{t(1-\delta_2)}$-dense rows is $O(n^{p(1-\delta_1)})$ while the number of $n^{p(1-\delta_1)}$-dense columns is $O(n^{t(1-\delta_2)})$. Next, in the generalization of Lemma 2, C' is divided into $cn^{p(1-\delta_1)} \times cn^{t(1-\delta_2)}$ sub-arrays of size $\frac{n^{p\delta_1}}{c} \times \frac{n^{t\delta_2}}{c}$, and so on. The further analogous details in the generalizations of consecutive lemmata as well as in Algorithm 1 are left to the full version.

The terms $n^{\omega(|p-\delta_1|,q,t)}$ and $n^{\omega(p,q,|t-\delta_2|)}$ in the claimed upper time-bound come from the final step (the generalization of Lemma 6), taking care of the non-zero $(n^{p(1-\delta_1)}, n^{t(1-\delta_2)})$-dense entries. They overshadow the term $n^{\omega(|p-\delta_1|,q,|t-\delta_2|)}$ coming from the iterations of a generalization of Algorithm 1. □

Note that since the sizes of A and B are not necessarily symmetric, forcing $\delta_1 = \delta_2$ might weaken the upper time-bound in Theorem 3.

5 Applications

Theorems 1, 2 yield corresponding output-sensitive upper time-bounds for several applications of Boolean matrix product and its witnesses, e.g., composition of binary relations, maximum witnesses of Boolean product, lowest common ancestors in directed acyclic graphs, prediction of the structure of the arithmetic matrix product, etc.

A *binary relation* is a set of ordered pairs over some universe. The composition of two binary relations R_1 and R_2 over a common universe U is a binary relation R_3 over U defined by $R_3 = \{(v,w)|\exists_{u \in U}(v,u) \in R_1 \wedge (u,w) \in R_2\}$.

Corollary 1. *If the composition of two binary relations over an n-element universe has at most s elements then it can be computed from these two relations by a randomized algorithm in time $\widetilde{O}(n^2 s^{\omega/2-1})$, almost certainly.*

A *maximum witness* for a non-zero entry $C[i,j]$ of the Boolean product of two $n \times n$ Boolean matrices A, B is the largest witness for $C[i,j]$, i.e., the largest index k such

that $A[i,k] = 1$ and $B[k,j] = 1$. The next corollary is concerned with computing maximum witnesses (one per each non-zero entry) in a sparse Boolean matrix product.

Corollary 2. *Let A and B be two $n \times n$ Boolean matrices whose product has at most $n^{2-2\delta}$ non-zero entries, and let $\lambda(\delta)$ be such that $\omega(1, \lambda(\delta), |1-\delta|) = 1 + \lambda(\delta)$. The maximum witnesses of the product of A and B can be computed by a randomized algorithm running in time $\widetilde{O}(n^{2+\lambda(\delta)})$, almost certainly. By augmenting this upper time bound with the additive term $O(n^{\omega})$, we can decrease the number of required random bits to $O(\log^2 n)$.*

Proof. The proof requires solely a slight modification of the proof of Theorem 16 in [10]. The key observation is that in the aforementioned proof the matrices C_p, $p = 1, ..., n/l$, have also at most $n^{2-2\delta}$ non-zero entries like the Boolean product C of A and B. Hence, since such a matrix C_p is the Boolean product of an $n \times n^r$ Boolean matrix with an $n^r \times n$ Boolean matrix (see [10]), where $r = \log_n l$, it can be computed in time $\widetilde{O}(n^{\omega(1,r,|1-\delta|)} + n^{\omega(|1-\delta|,r,1)} + n^{1+r})$ by Theorem 3. Therefore, by the proof of Theorem 16 in [10] and the equality $\omega(1, r, |1-\delta|) = \omega(|1-\delta|, r, 1)$, the total cost of computing the maximum witnesses is $\widetilde{O}(n^{1-r+\omega(1,r,|1-\delta|)} + n^{3-r} + n^{2+r})$. Analogously as in [10], we get rid of the additive term n^{3-r} assuming $r \geq \frac{1}{2}$, and solve the equation $1 - \lambda(\delta) + \omega(1, \lambda(\delta), |1-\delta|) = 2 + \lambda(\delta)$, implying $\lambda(\delta) \geq \frac{1}{2}$ by $\omega(1, \lambda(\delta), |1-\delta|) \geq 2$, in order to balance the two remaining exponent terms. This yields the first claimed upper time-bound. To estimate the number of non-zero entries in the rows and columns of the product matrix we can simply compute the product in time $O(n^{\omega})$ instead of using the algorithm of Cohen [7]. □

By [8,14], the best known upper bound on $\lambda(0)$ is 0.575 (see, e.g., [10]). Hence, the partially derandomized version of Corollary 2 should be interesting at least for small δ.

In turn, the problem of maximum witnesses has many applications [19]. Here, we just consider its original application to computing lowest common ancestors in directed acyclic graphs [10,15]. By combining Corollary 2 with Theorem 11 in [10], we obtain the corollary

Corollary 3. *Let G be a directed acyclic graph on n vertices such that at most $n^{2-2\delta}$ pairs of vertices in G have a common ancestor. Next, let $\lambda(\delta)$ be such that $\omega(1, \lambda, |1-\delta|) = 1 + \lambda(\delta)$. There is a randomized algorithm almost certainly reporting for all pairs of vertices in G with a common ancestor their lowest common ancestor, running in total time $\widetilde{O}(n^{2+\lambda(\delta)})$. By augmenting this upper time bound with the additive term $O(n^{\omega})$, we can decrease the number of required random bits to $O(\log^2 n)$.*

The methods of Theorems 1 and 2 applied to Boolean counterparts of two $n \times n$ arithmetic matrices yield the complete information on the location of non-zero entries in the product of these two arithmetic matrices under the assumption of no cancellation.

Corollary 4. *Let A and B be two $n \times n$ arithmetic matrices whose product has at most s non-zero entries. Under the assumption of no cancellation, the exact non-zero structure of the product of A and B can be determined by a randomized algorithm running in time $\widetilde{O}(n^2 s^{\omega/2-1})$, almost certainly. Furthermore, if the upper bounds on*

the number of non-zero entries in the respective rows and columns of the product matrix summing to at most $2s$ *are known* a priori *then the algorithm requires only* $O(\log^2 n)$ *random bits.*

6 Conclusions and Extensions

Our main new idea is the use of randomness to compute sparse Boolean product. We have applied the idea to the fast algorithms for Boolean product and its witnesses, treating them as black boxes.

The next natural step would be to derive general upper time bounds sensitive both to the sparsity of the input matrices as well as to that of the product matrix by combining the ideas from this paper with those from [2,22].

Acknowledgments

The author is grateful to all those who provided useful comments, in particular Mirosław Kowaluk, Christos Levcopoulos, Rasmus Pagh and several anonymous referees.

References

1. Agrawal, M., Kayal, N., Saxena, N.: PRIMES is in P. Annals of Mathematics 160(2), 781–793 (2004)
2. Amossen, R.R., Pagh, R.: Faster Join-Projects and Sparse Matrix Multiplication. To appear in proc. ACM International Conference on Database Theory (ICDT 2009), St. Petersburg (March 2009)
3. Arlazow, V.L., Dinic, E.A., Kronrod, M.A., Faradzev, I.A.: On economical construction of the transitive closure of an oriented graph. Soviet Math. Dokl. 11, 1209–1210 (1970)
4. Alon, N., Naor, M.: Derandomization, Witnesses for Boolean Matrix Multiplication and Construction of Perfect hash functions. Algorithmica 16, 434–449 (1996)
5. Bash, J., Khanna, S., Motwani, R.: On Diameter Verification and Boolean Matrix Multiplication. Technical Report, Stanford University CS department (1995)
6. Broder, A.Z., Charikar, M., Frieze, A.M., Mitzenmacher, M.: Min-Wise Independent Permutations. Journal of Computer and System Sciences 60, 630–659 (2000)
7. Cohen, E.: Structure Prediction and Computation of Sparse Matrix Products. Journal of Combinatorial Optimization 2, 307–332 (1999)
8. Coppersmith, D.: Rectangular matrix multiplication revisited. Journal of Symbolic Computation 13, 42–49 (1997)
9. Coppersmith, D., Winograd, S.: Matrix Multiplication via Arithmetic Progressions. J. of Symbolic Computation 9, 251–280 (1990)
10. Czumaj, A., Kowaluk, M., Lingas, A.: Faster algorithms for finding lowest common ancestors in directed acyclic graphs. The special ICALP 2005 issue of Theoretical Computer Science 380(1-2), 37–46 (2005)
11. Galil, Z., Margalit, O.: Witnesses for Boolean Matrix Multiplication and Shortest Paths. Journal of Complexity, 417–426 (1993)
12. George, A., Gilbert, J., Liu, J.W.H. (eds.): Graph Theory and Sparse Matrix Computation. The IMA Volumes in Mathematics and its Applications, vol. 56. Springer, Heidelberg (1993)

13. Golub, G., Van Loan, C. (eds.): Matrix Computations. The Johns Hopkins U. Press, Baltimore (1989)
14. Huang, X., Pan, V.Y.: Fast rectangular matrix multiplications and applications. Journal of Complexity 14, 257–299 (1998)
15. Kowaluk, M., Lingas, A.: LCA queries in directed acyclic graphs. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 241–248. Springer, Heidelberg (2005)
16. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, Cambridge (1995)
17. Munro, J.I.: Efficient determination of the transitive closure of a directed graph. Information Processing Letters 1(2), 56–58 (1971)
18. Rytter, W.: Fast recognition of pushdown automaton and context-free languages. Information and Control 67(1-3), 12–22 (1985)
19. Shapira, A., Yuster, R., Zwick, U.: All-pairs bottleneck paths in vertex weighted graphs. In: Proc. SODA 2007, pp. 978–985 (2007)
20. Seidel, R.: On the All-Pairs-Shortest-Path Problem. In: Proc. 24th annual ACM Symposium on Theory of Computing, pp. 745–749 (1992)
21. Schnorr, C.P., Subramanian, C.R.: Almost Optimal (on the average) Combinatorial Algorithms for Boolean Matrix Product Witnesses, Computing the Diameter. In: Rolim, J.D.P., Serna, M., Luby, M. (eds.) RANDOM 1998. LNCS, vol. 1518, pp. 218–231. Springer, Heidelberg (1998)
22. Yuster, R., Zwick, U.: Fast sparse matrix multiplication. ACM Transactions on Algorithms (TALG) 1(1), 2–13 (2004) (Preliminary version in proc. ESA 2004)