RC 21224 (94793) July 14, 1998
Computer Science/Mathematics

IBM Research Report

Fast Practical Algorithms for the Boolean-Product-Witness-Matrix Problem

Anshul Gupta, **Pankaj Rohatgi**, and **Ramesh Agarwal**

IBM Research Division
T. J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY 10598

{*anshul, rohatgi,* and *agarwal*}*@watson.ibm.com*

Research Division
Almaden · Austin · China · Delhi · Haifa · Tokyo · Watson · Zurich

Fast Practical Algorithms for the Boolean-Product-Witness-Matrix Problem

Anshul Gupta, Pankaj Rohatgi, and Ramesh Agarwal

IBM T. J. Watson Research Center
P. O. Box 218, Yorktown Heights, NY 10598

{*anshul, rohatgi,* and *agarwal*}*@watson.ibm.com*

RC 21224 (94793), July 14, 1998

1 Introduction

Given two $n \times n$ Boolean matrices A and B, their Boolean Product Witness Matrix (BPWM) is an $n \times n$ integer matrix W with $w_{ij} =$ some k such that $a_{ik} = b_{kj} = 1$, $1 \leq i, j, k \leq n$ and $w_{ij} = 0$ if no such k exists. The best known algorithms to date for the BPWM problem are due to Alon et al. [3, 4] and Seidel [16], and have worst-case time-complexities of $O(n^\omega \log n)$, where $O(n^\omega)$ is the time required to multiply two $n \times n$ integer matrices. Currently, the best known value of ω is 2.376 and is due to Coppersmith and Winograd [9]. Investigation of the BPWM problem in [3, 16] was motivated by its application to finding witnesses for transitive closure and shortest paths and the $O(n^\omega \log n)$ worst-case algorithms constructed are not practical due to their reliance on the Coppersmith-Winograd algorithm for integer matrix multiplication. Apart from its application in finding shortest paths and transitive closure, BPWM, and the related problem of Boolean matrix multiplication (without witnesses) are interesting problems in themselves.

In practical settings, rather than focusing on the worst-case complexity, it is often more useful to focus on average-case complexity (e.g., Quicksort [11]). In this paper, we describe a relatively simple algorithm for solving the BPWM problem which takes $O(n^2 \log n)$ time for a large class of input matrix pairs. We use an adversarial model to describe our algorithm's performance. In our model, A and B are drawn from a probability distribution created by an adversary in which the adversary has a large measure of control. For instance, the adversary is free to completely specify A. In addition, the adversary is allowed to specify the number of 1's in each of the columns of B. It can also introduce some dependence amongst the entries of B but is constrained by the fact that the probability distribution for B should be such that for any column i of B, if the adversary has mandated that it contain B_i 1's, then all possible n-bit binary vectors having B_i 1's should be equally likely candidates for being column i of B. This constraint is necessary to prevent the adversary from introducing structure into the columns of B on the basis of the rows of the matrix A it has chosen. However, this constraint does not prevent the adversary from mandating dependence between columns of B; e.g., the adversary can mandate that two columns of B be identical or complementary. We prove that our algorithm terminates in $O(n^2 \log n)$ steps with a high probability, even in this adversarial setting. The same result also holds if the roles of A and B are reversed and the adversary has full control of B but not of the rows of A. Although, instances of the BPWM problem can be constructed for which our algorithm does not terminate in $O(n^2 \log n)$ steps (it can take up to $O(n^3)$ steps in the worst case), it is clear from our analysis that such instances are extremely rare and only occur when there is a strong dependence between the rows of A and the columns of B. It should be noted, however, that dependence between the rows of A and the columns of B is a necessary, but *not* sufficient condition for our algorithm to exceed $O(n^2 \log n)$ steps in finding a BPWM. Therefore, we believe that our algorithm should be applicable in a number of practical settings.

In addition to Alon et al. [3, 4] and Seidel [16], other researchers have investigated Boolean matrix multiplication, although without finding witnesses in many cases. Galil and Margalit [10] present a BPWM algorithm with a complexity of $O(n^{(\omega+\epsilon)})$ for a positive ϵ. O'neil and O'neil [14] present an algorithm with $O(n^2)$ expected complexity for random Boolean matrices, where each entry is 1 with probability 0.5. Some researchers have considered computing the Boolean product matrix without witnesses using table look-up or related techniques that have a space-time tradeoff [2, 5, 17, 1].

The paper is organized as follows. In Section 2, we introduce our technique by describing how the BPWM problem can be solved in $O(n^2 \log n)$ time with high probability in the *simple* case where both the matrices A and B and their entries are drawn independently, with entries in A being 1 with probability p_a and entries in B being 1 with probability p_b. We then describe some experimental results which confirm our analysis in this case. In Section 3, we extend our algorithm to the more general case where the choice of the input matrices is made in the adversarial setting described above. Section 4 contains concluding remarks. The Appendix at the end of the paper gives detailed proofs of some of the theorems used in the paper.

2 The Simple Case

In this section, we introduce our algorithm by describing how the BPWM problem can be solved in $O(n^2 \log n)$ time with a high probability in the *simple* case where both the matrices A and B and their entries are drawn independently, with entries in A being 1 with probability p_a and entries in B being 1 with probability p_b. We assume that only the 1's are stored in the input Boolean matrices A and B, which are available in the standard sparse matrix format [7] in both compressed sparse row (CSR) and compressed sparse column (CSC) storage. We also assume that the column indices in the rows of A and the row indices in the columns of B are sorted. These data structures can be easily constructed in $O(n^2 \log n)$ time.

We first examine two classical approaches that can be used for solving the BPWM and analyze their expected complexity in the simple case.

2.1 The outer-product approach

This approach initializes the witness matrix W with all zeros and mimics the algorithm to compute the integer matrix product $C = A \times B$ using the sum of n outer products. Recall that

$$C = \sum_{k=1}^{n} a_{*k} \times b_{k*},$$

where a_{*k} is a $n \times 1$ vector corresponding to the k-th column of A and b_{k*} is a $1 \times n$ vector corresponding to the k-th row of B. Each nonzero entry in $a_{*k} \times b_{k*}$ contributes to an entry in C and k is a witness for that entry.

Therefore, while computing the k-th outer product we can place k as a witness in all (i, j) locations in W where the k-th outer product is nonzero. Since A and B are available in both CSR and CSC formats, computing each outer product is simply a matter of taking the Cartesian product of the set of all indices i such that $a_{ik} = 1$ with the set of all indices j such that $b_{kj} = 1$. Note that the witnesses may be overwritten and the nonzero entries in the final W matrix depend on the order in which the k's are processed.

Since entries in A and B are independent with each entry in A being 1 with probability p_a and each entry in B being 1 with probability p_a, the expected sum of all entries in the integer product $A \times B$ is $n^3 p_a p_b$. Clearly, this is also the expected number of steps needed by the algorithm described above, which is efficient for small values of $p_a p_b$.

2.2 The inner-product approach

The second algorithm explicitly searches for a witness for each w_{ij}. For each (i, j) pair, it traverses the i-th row of A and the j-th column of B. The algorithm starts by setting a pointer at k_a and k_b each, such that k_a and k_b are the smallest indices for which a_{ik_a} and $b_{k_b j}$ are 1. If $k_a = k_b$, then a witness w_{ij} has been, else the smaller of the two pointers is moved to next position where there is a 1 (note that since A and B are stored in CSR and CSC formats, moving the pointer to the next location is a constant time operation). If no such matching k_a and k_b are found, then w_{ij} is set to 0.

Since the probability that for a given (i, j, k), both a_{ik} and b_{kj} are 1 is $p_a p_b$, the expected number of k's for which both both a_{ik} and b_{kj} are 1 for a given row i of A and a given column j of B is $np_a p_b$. The expected number of 1's in a given row (column) of A (B) is np_a (np_b).

Therefore, the total number of distinct k's between row i of A and column j of B is $np_a + np_b - np_a p_b$. Since $np_a p_b$ matching k's are interspersed randomly in a set of $np_a + np_b - np_a p_b$ indices, on an average, the total number of indices examined before hitting upon a matching k is $(np_a + np_b - np_a p_b)/np_a p_b = 1/p_a + 1/p_b - 1$. On the other hand, $n(p_a + p_b)$ is an expected upper-bound on the total number of indices examined and is the likely amount of work required to determine the value of a given w_{ij} when p is very small. Thus, the inner-product algorithm requires approximately $n^2 \times \min(1/p_a + 1/p_b - 1, n(p_a + p_b))$ steps.

The inner-product algorithm is efficient for large values of p_a, p_b. For example, if $p_a = p_b = 0.5$, the inner-product approach is expected to take $O(n^2)$ steps, a fact observed in [14].

2.3 Hybrid algorithm

In the previous section, we saw that the BPWM problem can be solved in a small number of steps for random matrices with uniform densities of 1's equal to p_a and p_b either when both p_a and p_b are close to 0 (using outer products) or when both are close to 1 (using inner products). In fact, from the expressions derived above, it is easy to show that the expected number of steps is $O(n^2 \log n)$ for $0 \le p_a p_b \le \frac{\log n}{n}$. In this section, we strive to reduce the time-complexity of finding the BPWM of random Boolean matrices for the case when $p_a p_b > \frac{\log n}{n}$. We will use components of both the outer-product and the inner-product approaches to develop a hybrid algorithm.

We first introduce a key idea which forms the basis of our technique. It is easy to convert the outer product algorithm into a random sampling algorithm that can sample each of the entries in the integer product $C = A \times B$ with a frequency proportional to the value of the entry. In addition, each sample point corresponds to a witness. To see this, note that the integer product C is the sum of n outer-products. Let $\mathcal{S}$ denote the sum of all entries in C. Then $\mathcal{S} = \sum_{k=1}^{n} S_k$ where S_k is the sum of all entries in the k-th outer product. Therefore, if we pick an index $k \in \{1, \ldots, n\}$ with probability of $k = t$ being S_t/S and randomly pick (with uniform probability) an (i, j) pair such that a_{ik} and b_{kj} are nonzero, then we get a sample in C with the required probability distribution. Note that this process is very efficient given that the S_k's can be computed in a single pass involving n integer multiplications and the random picking of an (i, j) pair corresponding to the witness k is facilitated by the CSR and CSC storage of A and B.

If $\mathcal{S} \leq O(n^2 \log n)$, we use the outer-product formulation directly to compute witnesses; otherwise, we start the sampling process. While sampling, we keep track of the number of samples already taken and the number of empty slots remaining in W; i.e, the number of (i, j) pairs that have not been picked up in any of the trials. Thus, at any given time, we know the number of steps we have already performed (say, r), and number of steps that would be required to explicitly examine (by the inner-product method) the slots for which a witness has not been found. We know that examining each such slot can take a maximum of n steps. So we switch from sampling to explicit examination when the number of empty slots x in C is such that $xn = r$. Thus, we strive to apportion the number of operations equally between the two phases. Since we can keep track of the current cost and the estimated remaining cost of the algorithm at all times during the execution, it is possible to apportion the work between the two phases in any specific ratio. In practice, this ratio should be inversely related to the relative cost of operations in each phase.

2.4 Sampling without replacement

The process of selection of (i, j) entries in the algorithm described in Section 2.3 can pick the same (i, j) multiple number of times. Although this does not have any bearing on the asymptotic complexity, in practice, it is possible to sample the (i, j) entries in the witness matrix without replacement in a pseudo-random manner. As we will see in Section 2.6, this does reduce the number of steps required to solve the BPWM problem.

We first construct a special data structure for each k between 1 and n. Let the number of i's (such that $a_{ik} = 1$) and j's (such that $b_{kj} = 1$) associated with a given k be *inum*$_k$ and *jnum*$_k$, respectively. An array J_k of size *jnum*$_k$ contains the j indices in random order and an array I_k of size *inum*$_k$ contains the i indices. In addition, an array *ptr*$_k$ of size *inum*$_k$ contains random pointers into J_k (one for each entry in I_k) and and array *count*$_k$ of size *inum*$_k$ contains all 0's (initially). In order to draw a sample, we first select a k such that the probability of selecting a given k is proportional to the total number of witnesses corresponding to that k. Let $d_k =$ *inum*$_k\times$ *jnum*$_k$ and let $\mathcal{S} = \sum_{k=1}^{n} d_k$. Then k is selected such that in r trials, a given k is selected rd_k/d times. After selecting a k, we randomly select an integer z between 1 and *inum*$_k$. The resulting randomly selected entry (i, j) in W is the pair $(I_k[z], J_k[\textit{ptr}_k[z]])$. After having completed the actions corresponding to selecting the (i, j) entry, *ptr*$_k[z]$ and *count*$_k[z]$ are incremented. If *ptr*$_k[z]$ becomes equal to *jnum*$_k$, then *ptr*$_k[z]$ is wrapped around and set to 1. If *count*$_k[z]$ becomes equal to *jnum*$_k$ (indicating that the i index in $I_k[z]$ has paired with all possible j indices), then this index is deleted from the corresponding data-structures of the selected k. This is accomplished by replacing $I_k[z]$, *ptr*$_k[z]$, and *count*$_k[z]$ by $I_k[\textit{inum}_k]$, *ptr*$_k[\textit{inum}_k]$, and *count*$_k[\textit{inum}_k]$, respectively, and then decrementing *inum*$_k$.

2.5 Complexity of the hybrid algorithm

In this section, we prove that our algorithm terminates with high probability within $O(n^2 \log n)$ steps. Our proof is based on Theorems 1 and 2.

Theorem 1 *Let A and B be two random Boolean matrices with dimensions $m \times n$ and $n \times l$, respectively. Each entry in A is chosen independently (of other entries in A and of B) with a probability p_a of being 1 and $1 - p_a$ of being 0. Similarly, each entry in B is chosen independently (of other entries in B and of A) with a probability p_b of being 1 and $1 - p_b$ of being 0. Let C denote the $m \times l$ integer matrix product of A and B. Let $\mu = np_ap_b$. Then,*

1. *If $\mu \leq 10\log n$, then, for any fixed (i,j), $Prob(c_{ij} \; > \; 20\log n) \; \leq n^{-5}$.*

2. *If $\mu > 10\log n$, then, for any fixed (i,j),*

   - $Prob(c_{ij} \; > \; 2\mu) \; \leq n^{-5}$.
   - $Prob(c_{ij} \; < \; \mu/3) \; \leq \; n^{-4}$.

Proof Summary: *In this section we provide only an outline of the proof of Theorem 1; a complete detailed proof appears in the Appendix. For any x, y let $\mathcal{A}_{x,y}$ and $\mathcal{B}_{x,y}$ denote the indicator random variables corresponding to the entries a_{xy} and b_{xy} respectively. For any fixed (i,j), let $\mathcal{C}_{i,j}$ denote the random variable corresponding the the integer value of c_{ij}. For $k = 1, \ldots, n$, define new indicator variables $\mathcal{D}_{i,j,k}$ to be $\mathcal{A}_{i,k}\mathcal{B}_{k,j}$. Then we have*

$$\mathcal{C}_{i,j} = \sum_{k=1}^{n} \mathcal{D}_{i,j,k}.$$

From the underlying independence assumptions, it follows that the indicator random variables $\mathcal{D}_{i,j,k}$, $k = 1, \ldots, n$ are mutually independent with mean $p_a p_b$. $\mathcal{C}_{i,j}$ is a sum of independent indicator random variables with $E(\mathcal{C}_{i,j}) = np_a p_b$. Therefore, we can apply the Chernoff-Hoeffding [6, 12] bounds to $\mathcal{C}_{i,j}$ to strongly bound any deviation of $\mathcal{C}_{i,j}$ from its expected value. The Chernoff-Hoeffding bounds state that if X is the sum of independent bounded random variables with mean μ then for any $\varepsilon > 0$,

$$Prob\ (\ X \ \geq \ (1+\varepsilon)\mu\ ) \ \leq \ F(\mu, \varepsilon)$$

and for any $\varepsilon, 0 < \varepsilon < 1$,

$$Prob\ (\ X \ \leq \ (1-\varepsilon)\mu\ ) \ \leq \ F(\mu, -\varepsilon),$$

where $F(\mu, \varepsilon)$ is defined to be the expression $\left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{(1+\varepsilon)}}\right)^{\mu}$. The claims of the theorem follow directly by substituting for the values of μ and ε in the above bounds (see Appendix).

Now let us examine how this theorem can be used in the analysis of our approach. If $\mu = np_a p_b < 10\log n$, (i.e., A and B are sparser than a certain threshold) then we use only the outer product algorithm. The outer product algorithm takes as many steps as the sum of all entries in C. From Theorem 1 it follows that this sum exceeds $20n^2\log n$ only with at most a vanishingly small probability of n^{-3} and therefore with very high probability our algorithm takes less than $20n^2\log n$ steps.

When $\mu = np_a p_b > 10\log n$, we apply the hybrid algorithm. Theorem 1 guarantees that the probability that *any* entry in C exceeds 2μ or is less than $\mu/3$ is less than n^{-2}, which is again vanishingly small. Therefore, with exceedingly high probability the matrix C has all entries within the range $\{\mu/3, \ldots, 2\mu\}$. We now present a theorem that can be used to prove that the random sampling approach used by the hybrid algorithm is extremely effective on such matrices with bounded entries.

Theorem 2 *Consider a series of r trials where each trial involves throwing a fresh dart uniformly at random on a t-slot dart-board. Let X_r denote the number of slots that did not receive a single dart after r trials. Then, for large t,*

- $E(X_r) = te^{-r/t}$.
- $Prob(\ X_r > (1+\varepsilon)(E(X_r))\ ) \ \leq \ F(E(X_r), \varepsilon)$, *where F is the Chernoff-Hoeffding bound.*

Proof: *Consider the indicator random variables $\mathcal{D}_i$ (for $i \in \{1, \ldots t\}$) defined as follows: $\mathcal{D}_i = 1$ if and only if the i'th slot does not receive any dart; else $\mathcal{D}_i = 0$. The first part of the theorem is equivalent to the Coupon Collector's problem in [13], which also contains the proof.*

If the random variables $\mathcal{D}_i$ for all $i \in \{1, \ldots t\}$ were independent then the result would follow by a direct application of Chernoff-Hoeffding bounds. However the random variables $\mathcal{D}_i$ are not independent although the r darts were are thrown independently. However we argue (as was done in [15]) that the variables $\mathcal{D}_i$ for all $i \in \{1, \ldots, t\}$ are possess the property of being self-weakening *with parameter $\lambda = 1$. To see this, note that for any i and for any $(j_1, \ldots, j_k) : \{j_1, \ldots, j_k\} \subseteq 1, 2, \ldots, i-1$*

$$Prob(D_i \;=\; 1 \mid D_{j_1} \;=\; D_{j_2} \;=\; \ldots \;=\; D_{j_k} \;=\; 1) = (1 - 1/(t-k))^r \;\leq\; (1 - 1/t)^r.$$

Therefore the Chernoff-Hoeffding upper-tail bounds are applicable and the result follows.

In order to apply Theorem 2 to the analysis of our hybrid algorithm, we now define our dart-board. As we mentioned earlier the matrix C on which the hybrid algorithm works has all entries in the range $\mu/3$ and 2μ with high probability. In the first stage of the hybrid algorithm, these entries are randomly sampled with frequencies which are directly proportional to the entries themselves.

Let the minimum entry in C have a value of k where $k \geq \mu/3$. Each entry c_{ij} in C is between k and $6k$ and in fact has a value of $x_{ij}k$ where for some real number x_{ij} where $1 \leq x_{ij} \leq 6$ with a high probability. We allocate q_{ij} slots on the dart-board for this entry, where $q_{ij} = \lfloor x_{ij} \rfloor$. Now the analogy between the random sampling in the hybrid algorithm and the dart throwing in Theorem 2 becomes apparent, i.e., picking an (i,j) entry in hybrid algorithm sampling process is equivalent to picking one of the q_{ij} slots corresponding to the (i,j) entry in the dart board.[1] The total number of slots t in Theorem 2 is at most $6ml$ and at least ml and if we chose r equal to $t \lg n$ which is at most $6ml \lg n$, then $E(X_r)$ is t/n which is at most $6ml/n$. Also, from Theorem 2 and it follows that $Prob(X_r > 2t/n) \leq 2^{-t/2n}$.

Qualitatively, the above result indicates that after throwing no more than $6ml \lg n$ darts, the number of empty slots (slots in which no dart has fallen) on the new board is less than $12ml/n$ with a very high probability. In the worst case, this corresponds to $12ml/n$ entries of C, which can be explicitly examined in a total time of at most $12ml$. For square matrices where $m = l = n$, the number of steps in the hybrid algorithm does not exceed $6n^2 \lg n$ (for random sampling) + $12n^2$ (for explicit inner products) with a very high probability.

2.6 Experimental results

In this section we experimentally verify some of the analytical expressions derived in the previous sections. We present some results of running the outer-product, the inner-product, and the hybrid algorithms on random square Boolean matrices in which each entry is 1 with probability p.

Figure 1 shows how the operation counts of the BPWM algorithms discussed so far vary with p for 1000×1000 input matrices. The solid lines depict the operation counts of the two implementations of the hybrid algorithm. The lower of the two solid lines (with slightly smaller operation count) corresponds to the version, in which, the sampling is done without replacement (Section 2.4). The upper solid line corresponds to the version described in Section 2.3, in which sampling is done with replacement. In this experiment, we resort to the hybrid algorithm only if the cost of the outer-product algorithm is higher than the expected cost of the hybrid algorithm; otherwise, we use the pure outer-product algorithm. The

[1] The discrepancy introduced due to the rounding can be ignored since it only has a minor effect that is readily remedied by increasing the number of darts thrown by a small compensatory constant factor.

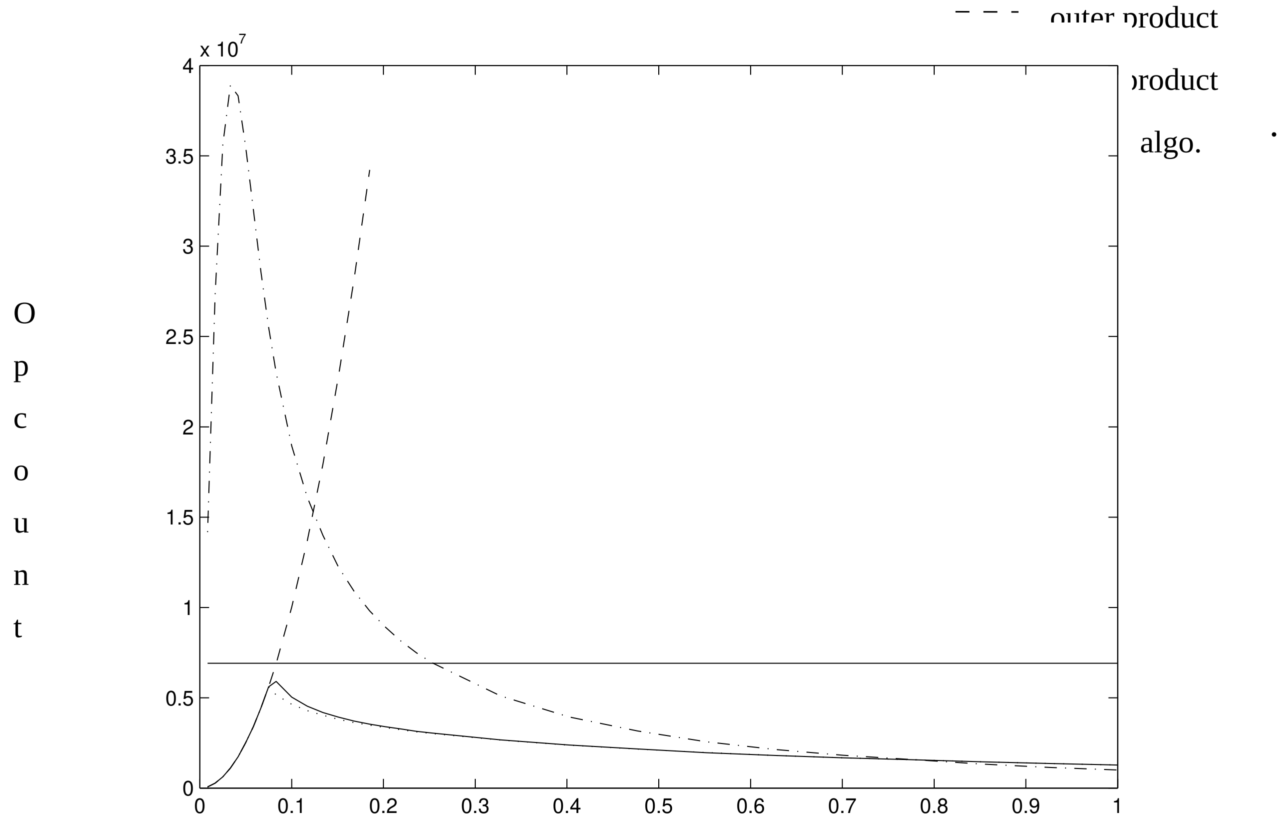

Figure 1: Number of operations vs. p (density of 1's in input matrices) for BPWM with different algorithms.

analysis in Section 2.5 suggests that we do not need to throw more than $6n^2 \log n$ darts (with a very high probability). However, as described in Section 2.3, the initial random sampling phase can be terminated even sooner. We terminate the dart throwing phase as soon as the number of samples drawn is equal to the expected number of operations required to examine the remaining slots. Thus, we apportion the number of operations equally between the two phases.

The curves corresponding to the pure outer- and inner-product algorithms are close to expected expressions of n^3p^2 and $n^2 \times \min(2/p - 1, 2np)$, respectively, as derived analytically in Section 2. The outer- and inner-product curves are expected to meet when $p = (2/n)^{-3}$, at which point, the operation count of both the algorithms is expected to be $2^{2/3}n^{7/3}$. Therefore, the expected point of intersection of the two curves for $n = 1000$ is $(0.126, 1.59 \times 10^7)$. The actual point in the simulation is approximately $(0.126, 1.57 \times 10^7)$.

In Figure 1, both the solid-line curves, which depict the performance of the hybrid algorithm (with and without replacement, respectively) peak at certain points. However, they always stay well below the $n^2 \log n$ line. In Figure 2, we plot the maximum number of operations for any p required by the hybrid algorithms to find the BPWM against the size of input matrices expressed as n^2. In other words, Figure 2 plots the peaks of the operation-count vs. p curves for different matrix sizes. The $n^2 \log n$ and $n^{7/3}$ (which is the maximum number of operations required by the best of the outer-product and inner-product algorithms) curves are included for comparison. The figure shows that the point corresponding to the peak of the number of operations vs. p curve stays below $n^2 \log n$ as the size of the input matrices increases. Thus it experimentally verifies the $O(n^2 \log n)$ bound for finding the BPWM of uncorrelated square matrices with uniformly distributed 1's and 0's.

3 Solving BPWM for Arbitrary Mutually Independent Matrices

Having introduced our technique in the last section, we now describe how it can be modified to handle arbitrary mutually independent matrices. The assumption that A and B are randomly chosen matrices with independently chosen entries being 1 with probability p_a and p_b respectively are somewhat unrealistic and may not hold in practice. We would like to create an algorithm which solves the BPWM problem in $O(n^2 \log n)$ time for a much wider class of matrices that would cover most practical applications. At a minimum, our algorithm should be able to handle density variations in the number of 1's in the various rows and columns of A and B. This density variation immediately has an adverse effect on our simple hybrid algorithm described in Section 2.3 since at the random sampling stage, most of our witnesses would be for those entries in C that are inner products of dense rows of A with dense columns of B. Thus, it would miss a lot of elements of C that are inner products of sparse rows of A with sparse columns of B. In addition, we would like to argue that for any fixed A (or B) we can solve the BPWM problem for a very wide choice of B (or A) matrices in $O(n^2 \log n)$ time. To cover all these cases and to prove that our algorithm can work for a wide range of distributions of A and B, we define an adversarial model, where an adversary decides the distribution of A and B, within some constraints and we prove our result in this adversarial model.

3.1 The adversarial model

Assume the existence of an adversary who produces the following:

- An $n \times n$ Boolean matrix A.
- A list of the number of 1's that must be there in each column of B.
- A probability distribution on possible $n \times n$ Boolean matrices B consistent with the column density requirement. This probability distribution is constrained so that it must ensure that for any column i of B, if the adversary has mandated it to have t 1's, then all possible n-bit vectors with t 1's should be equally likely to be column i of B.

The constraint on the probability distribution is necessary to prevent the adversary from introducing structure into the columns of B on the basis of the rows of the matrix A it has chosen. Note that this constraint does not prevent the adversary from mandating dependence between columns of B; e.g., the adversary can mandate that two columns of B be identical or complementary.

3.2 The extended algorithm

We make only a small change to our simple hybrid algorithm described in Section 2.3, mostly to take care of varying row densities in A (column densities in B). With this small modification, our algorithm works in $O(n^2 \log n)$ time even in the adversarial setting described above.

The extended algorithm first sorts the rows of A and the columns of B in increasing order of the number of 1's in them. It partitions A row-wise into $(\lceil \log_2 n \rceil + 1)$ parts of dimension $m_i \times n$, where $0 \leq i \leq \lceil \log_2 n \rceil$. Submatrix A_i contains all rows in which the number of 1's is between 2^{i-1} (excluding 2^{i-1}) and 2^i (including 2^i). A_0 has only those rows that have one or no 1's. We partition the matrix

B similarly, but column-wise, into submatrices of dimension $n \times l_j$, where $0 \le j \le \lceil \log_2 n \rceil$. With this partitioning of A and B, the problem of computing the $n \times n$ BPWM matrix C is reduced to computing $(1 + \lceil \log_2 n \rceil)^2$ individual submatrices $C_{i,j}$, where the $m_i \times l_j$ submatrix $C_{i,j}$ is the BPWM of the $m_i \times n$ submatrix A_i and the $n \times l_j$ submatrix B_j. We then use the simple hybrid BPWM algorithm as described in Section 2.3 to compute witnesses for all these submatrix multiplications.

3.3 Analysis of the extended algorithm

Theorem 3 is at the core of our proof that our extended algorithm works in $O(n^2 \log n)$ time with high probability. A symmetric argument can be used to show that our result would still hold if the roles of A and B are were reversed and the adversary has full control of B but not of the rows of A. Theorem 3 is directly applicable to the simple hybrid algorithm working on any of the submatrix witness matrix problem that the original problem was decomposed to. In these cases the constant c, which is a parameter in the theorem, turns out to be 2.

Theorem 3 *Let P be any $m \times n$ Boolean matrix chosen by an adversary having between N_p and cN_p 1's in each row (for any fixed constant $c > 1$) and Q be a $n \times l$ random matrix having between N_q and cN_q 1's in each column. Moreover, assume Q is chosen according to the following criteria: First, the number 1's in each column of Q is chosen by the adversary between N_q and cN_q, where N_q is a given constant between 0 and n. Then, the elements of Q are picked (not necessarily independently) according to a probability distribution chosen by the adversary which must ensure that each column j of Q is uniformly distributed over all possible n-bit vectors with exactly n_j 1's ($N_q \le n_j \le cN_q$). Let $\mu_u = c^2 N_p N_q / n$ and $\mu_l = N_p N_q / n$. The following then holds for the integer matrix product $D = P \times Q$:*

1. *If $\mu_l \le 10 \log n$, then for any fixed (i, j), $Prob(d_{ij} > 20c^2 \log n) \le n^{-5}$.*
2. *If $\mu_l > 10 \log n$, then for any any fixed (i, j),*
   - $Prob(d_{ij} > 2\mu_u) \le n^{-5}$.
   - $Prob(d_{ij} < \mu_l / 3) \le n^{-4}$.

Proof Summary: *Note that the claims of Theorem 3 are remarkably similar to those of Theorem 1. In fact if the constant c, which determines the variation in the number of 1's in the rows of P and columns of Q, were to be set to 1 (meaning no variation), then the claims are identical. This is no coincidence; in fact, the main idea in the proof of this theorem is that even though the independence requirements needed to apply Chernoff-Hoeffding bounds directly (as was done in Theorem 1) are not met in this setting, the random variables in question possess a* self-weakening *property that permits these bounds to be applied [15]. We give a detailed proof of this theorem in the Appendix.*

Due to the similarity of the claims of Theorem 3 to those of Theorem 1, by following a procedure similar to that described in Section 2.5, one can show the simple hybrid algorithm of Section 2.3 will compute the BPWM of the matrices P and Q described in Theorem 3 in $c^2(6ml \log n + 12ml)$ steps with a very high probability. In the particular application where P and Q are submatrices resulting from the decomposition of a BPWM problem by the extended algorithm (in Section 3), the constant c is 2 and the BPWM problem takes $24ml \log n + 48ml$ steps.

Now let us examine the complexity of the extended algorithm. Counting the number of 1's in the rows of A and columns of B and sorting the rows of A and columns of B in increasing order of density takes at most $O(n^2)$ time. Computing any $C_{i,j}$ in which either $i = 0$ or $j = 0$ is trivial because each row of A_0

and column of B_0 have at most one 1. So any entry of $C_{0,j}$ and $C_{i,0}$ can be looked up in constant time. As discussed in the previous paragraph, a high-probability bound on the number of steps for computing any given $C_{i,j}$ is $24m_il_j \lg n + 48m_il_j$. Note that in some of the submatrix multiplications, only the outer-product algorithm may suffice due to sufficiently low density of either or both the input submatrices A_i and B_j. Working only with the dominant cost term, the overall cost of computing all the submatrices is $O(\sum_{i=1}^{\lceil \log_2 n \rceil} \sum_{j=1}^{\lceil \log_2 n \rceil} m_il_j \log n) = O(\sum_{i=1}^{\lceil \log_2 n \rceil} m_in \log n) = O(n^2 \log n)$. This is the highest order term in the entire operation, and hence the overall time-complexity of the extended BPWM algorithm described in this section.

4 Concluding Remarks and Future Work

In this paper, we describe a practical algorithm for computing the witness matrix W of two independent Boolean matrices A and B with independent patterns of 0's and 1's in them. A high-probability bound on the complexity of our algorithm is $O(n^2 \log n)$ for $n \times n$ square input matrices and $O(ml \log n)$ when the input matrices are rectangular with dimensions $m \times n$ and $n \times l$, respectively. This is smaller than the worst-case complexity of the best known BPWM algorithm to date. Although we have analyzed the complexity of our algorithm in an adversarial setting where the input matrices A and B are uncorrelated; i.e., after having chosen A and having determined the number of 1's (say t) in a column of B, the adversary must chose this column at random uniformly over all n-bit vectors with t 1's, a violation of this condition does not necessarily break the analysis down. Note that a necessary and sufficient condition for the applicability of Theorem 2 is that the ratio between the smallest and the largest elements of the submatrix $C_{i,j}$—the integer product of submatrices $A_{i,k}$ and $B_{k,j}$—in the extended algorithm (Section 3) be a constant. This can be true even for correlated matrices, as long as the bias produced in the entries of the integer product is not large or is in the same direction of the expected value of the entries for a majority of the entries. Therefore, we believe that our algorithm is adequate for most practical problems, although instances of BPWM problems can be constructed by aligning the 1's among the rows of A and columns of B in special patterns that may cause the algorithm to perform an asymptotically higher number of steps ($O(n^3)$ in the worst case). An appropriate analogy for this situation would be the Quicksort algorithm [11]. Quicksort is one the most widely used sorting algorithm and has an average-case complexity of $O(n \log n)$ for sorting n numbers. However, special sequences can be constructed to force it to take $O(n^2)$ time.

The applicability of our algorithm or a similar approach to the problem of transitive closure and all-pairs shortest paths needs further investigation and analysis in the light of the possible dependence between matrices that arise in these settings. As we alluded earlier, our approach works for more cases than we have proven and it may be possible to show that our approach, or a simple modification of it, can help solve these problems within high-probability time bounds similar to the one in this paper in most problem instances. We leave this as future work.

In addition to the fast BPWM algorithm, our approach can also be modified to approximate the integer product of two uncorrelated Boolean Matrices. However, efficient approaches already exist for this problem, even for general matrix multiplication [8].

References

[1] Karl Abrahamson. A time-space tradeoff for Boolean matrix multiplication. In *Proceedings of 30th Annual Symposium on Foundations of Computer Science*, pages 412–419, 1990.

[2] Leonard Adleman, Kellog Booth, Franco Preprata, and Walter Ruzzo. Improved time and space bounds for Boolean matrix multiplication. *Acta Informatica Complexity*, 11:61–75, 1978.

[3] Noga Alon, Zvi Galil, Oded Margalit, and Moni Naor. Witnesses for Boolean matrix multiplication and for shortest paths. In *Proceedings of 32nd Annual Symposium on Foundations of Computer Science*, pages 417–426, 1992.

[4] Noga Alon and Moni Naor. Derandomization, witnesses for Boolean matrix multiplication and construction of perfect hash functions. *Algorithmica*, 16:434–449, 1996.

[5] Michael D. Atkinson and N. Santoro. A practical algorithm for Boolean matrix multiplication. *Information Processing Letters*, 29:37–38, September 1988.

[6] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–509, 1952.

[7] E. Chu, Alan George, Joseph W.-H. Liu, and Esmond G.-Y. Ng. Users guide for SPARSPAK–A: Waterloo sparse linear equations package. Technical Report CS-84-36, University of Waterloo, Waterloo, IA, 1984.

[8] Edith Cohen and David Lewis. Approximating matrix multiplication for pattern recognition tasks. In *Proceedings of the 8'th ACM-SIAM Symposium on Discrete Algorithms*, 1997.

[9] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9:251–280, 1990.

[10] Zvi Galil and Olded Margalit. Witnesses for Boolean matrix multiplication and for transitive closure. *Journal of Complexity*, 9:201–221, 1993.

[11] C. A. R. Hoare. Quicksort. *Computer Journal*, 5(1):10–15, 1962.

[12] W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, 58:13–30, 1963.

[13] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[14] Patrick E. O'Neil and Elizabeth J. O'Neil. A fast expected time algorithm for Boolean matrix multiplication and transitive closure. *Information and Control*, 22:132–138, 1973.

[15] A. Panconesi and A. Srinivasan. Fast randomized algorithms for distributed edge coloring. In *Proceedings of ACM Symposium on Principles of Distributed Computing*, pages 251–262, 1992.

[16] Raimund Seidel. On the all-pairs-shortest-path problem. In *Proceedings of the 24th ACM Symposium on Theory of Computing*, pages 745–749, 1992.

[17] J. Vyskoc. A note on Boolean matrix multiplication. *Information Processing Letters*, 19:249–251, September 1984.

Appendix

Theorem 1 *Let A and B be two random Boolean matrices with dimensions $m \times n$ and $n \times l$, respectively. Each entry in A is chosen independently (of other entries in A and of B) with a probability p_a of being 1 and $1 - p_a$ of being 0. Similarly, each entry in B is chosen independently (of other entries in B and of A) with a probability p_b of being 1 and $1 - p_b$ of being 0. Let C denote the $m \times l$ integer matrix product of A and B. Let $\mu = np_a p_b$. Then,*

1. *If $\mu \leq 10 \log n$, then, for any fixed (i, j), $Prob(c_{ij} \ > \ 20 \log n) \ \leq n^{-5}$.*
2. *If $\mu > 10 \log n$, then, for any fixed (i, j),*
   - $Prob(c_{ij} \ > \ 2\mu) \ \leq n^{-5}$.
   - $Prob(c_{ij} \ < \ \mu/3) \ \leq \ n^{-4}$.

Proof: Consider the entry c_{ij} for any fixed (i, j). For any (i, j) let $\mathcal{A}_{i,j}$ and $\mathcal{B}_{i,j}$ be indicator random variables corresponding to the entries a_{ij} and b_{ij} respectively and let $\mathcal{C}_{i,j}$ denote the random variable corresponding the the integer entry c_{ij}. Clearly,

$$\mathcal{C}_{i,j} = \sum_{k=1}^{n} \mathcal{A}_{i,k} \mathcal{B}_{k,j}$$

Denote a new indicator variable $\mathcal{D}_{i,j,k}$ to be the product $\mathcal{A}_{i,k}\mathcal{B}_{k,j}$. Then we have

$$\mathcal{C}_{i,j} = \sum_{k=1}^{n} \mathcal{D}_{i,j,k}.$$

From the underlying independence assumptions on the random process used to generate A and B we have that for any fixed i, j, the indicator random variables $\mathcal{D}_{i,j,k}$ where $k = 1, \ldots, n$ are independent. Also we have for any k,

$$Prob(\mathcal{D}_{i,j,k} = 1) = p_a p_b$$

and

$$E(\mathcal{C}_{i,j}) = np_a p_b.$$

Moreover the Chernoff-Hoeffding [6, 12] bounds are applicable to $\mathcal{C}_{i,j}$ and strongly bound the probability that c_{ij} will be close to its expected value $\mu = np_a p_b$.

$$Prob(c_{ij} \ \geq \ (1 + \varepsilon)\mu) \ \leq F(\mu, \varepsilon),$$

where, $F(\mu, \varepsilon) = \left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{(1+\varepsilon)}}\right)^{\mu}$ is the Chernoff-Hoeffding bound on $Prob(c_{ij} \ \geq \ (1 + \varepsilon)\mu)$.
Similarly,

$$Prob(c_{ij} \leq (1 - \varepsilon)\mu) \leq F(\mu, -\varepsilon).$$

Note: The lower-tail Chernoff-Hoeffding bounds for binary random variables $X_1, \ldots, X_n$ actually follow from the upper-tail bounds on the complementary random variables $Y_1, \ldots, Y_n$, which are defined as $Y_i = 1 - X_i$. This fact will play a major role in our later proofs.

Substituting the parameters of Theorem 1, we get the desired results.

- If $\mu = np_a p_b \leq 10 \log n$, then for any fixed (i, j),

$$Prob(c_{ij} > 20 \log n) \leq (e/4)^{10 \log n} \leq n^{-5}$$

- If $\mu = np_a p_b > 10 \log n$, then for any fixed (i, j),

$$Prob(c_{ij} > 2\mu) \leq (e/4)^{\mu} \leq (e/4)^{10 \log n} \leq n^{-5},$$

$$Prob(c_{ij} < \mu/3) \leq (0.7405)^{\mu} \leq n^{-4}.$$

Theorem 3 *Let A be any $m \times n$ Boolean matrix chosen by an adversary having between N_a and cN_a 1's in each row (for any fixed constant $c > 1$) and B be a $n \times l$ random matrix having between N_b and cN_b 1's in each column. Moreover, assume B is chosen according to the following criteria: First, the number 1's in each column of B is chosen by the adversary between N_b and cN_b, where N_b is a given constant between 0 and n. Then, the elements of B are picked (not necessarily independently) according to a probability distribution chosen by the adversary which must ensure that each column j of B is uniformly distributed over all possible n-bit vectors with exactly n_j 1's ($N_b \leq n_j \leq cN_b$). Let $\mu_u = c^2 N_a N_b / n$ and $\mu_l = N_a N_b / n$. The following then holds for the integer matrix product $C = A \times B$:*

1. *If $\mu_l \leq 10 \log n$, then for any fixed (i, j), $Prob(c_{ij} > 20 \log n) \leq n^{-5}$.*
2. *If $\mu_l > 10 \log n$, then for any any fixed (i, j),*
   - $Prob(c_{ij} > 2\mu_u) \leq n^{-5}$.
   - $Prob(c_{ij} < \mu_l/3) \leq n^{-4}$.

Proof: First we prove the case when $(N_a N_b)/n < 10 \log n$. Take any fixed (i, j) and concentrate on the entry c_{ij}. Let the number of 1's in the i'th row of A be $a_i = l_i N_a$ where $1 \leq l_i \leq c$. Set $\mathcal{S}$ denote the set of indices t, for which a_{it} is 1. Assume the number of 1's in the j'th column of B are specified by the adversary to be $b_j = m_j N_b$, where $1 \leq m_j \leq c$. Let $\mathcal{B}_{i,j}$ denote the indicator random variable which is 1 if b_{ij} is 1 and 0 otherwise. Let $\mathcal{C}_{i,j}$ be the random variable denoting the value of c_{ij}.

By definition,

$$\mathcal{C}_{i,j} = \sum_{t \in \mathcal{S}} \mathcal{B}_{t,j}.$$

Therefore,

$$E(\mathcal{C}_{i,j}) = \sum_{t \in \mathcal{S}} E(\mathcal{B}_{t,j}).$$

For any t, $b_{tj} = 1$ with probability

$$\frac{\begin{pmatrix} n-1 \\ b_j - 1 \end{pmatrix}}{\begin{pmatrix} n \\ b_j \end{pmatrix}} = \frac{b_j}{n}$$

Therefore,

$$E(\mathcal{C}_{i,j}) = (a_i b_j)/n.$$

Note that the random variables $\mathcal{B}_{t,j}$ $(t \in \mathcal{S})$ are not mutually independent. If they were, then the result would directly follow from Chernoff-Hoeffding bounds. To see this, note that Chernoff-Hoeffding bounds will give

$$Prob(c_{ij} > (1+\varepsilon)E(\mathcal{C}_{i,j})) < F(a_i b_j/n, \varepsilon).$$

Clearly, $c_{ij} > 20c^2 \log n$ implies that $c_{ij} > E(\mathcal{C}_{i,j})$ since $E(\mathcal{C}_{i,j})$ is at most $(c^2 N_a N_b)/n$ which is at most $10c^2 logn$. The exact bound will depend on the value of ε and $E(\mathcal{C}_{i,j})$ in the above equation. Note that $F(\mu, \varepsilon)$ decreases with increasing values of μ and ε. Note that $c_{ij} > 20c^2 \log n$ implies that the value of ε is at least 1 since the maximum value of $E(\mathcal{C}_{i,j})$ is at most $10c^2 \log n$. Similarly, $E(\mathcal{C}_{i,j})$ is at least $10 \log n$. From this it follows that

$$Prob(c_{ij} > 20c^2 \log n) \;\leq\; F(10 \log n, 1) \;\leq\; n^{-5}.$$

However, we now show that although the random variables $\mathcal{B}_{t,j}(t \in \mathcal{S})$ not mutually independent they do possess the *self-weakening* property (see [15]) which allows us to use the Chernoff-Hoeffding bounds on the upper tail of their distribution. Informally, this means that although $\mathcal{C}_{i,j}$ is a sum of *dependent* indicator random variables, each with an expected value of b_j/n, their mutual dependence is such that the chances of a particular random variable being 1 decreases if other random variables in the sum are known to be 1. In other words, the sum of these dependent random variables is even less likely to exceed its expected value than if these random variables were independent. Therefore, the Chernoff-Hoeffding bound for the upper tail which holds for independent variables also holds for these variables.

More formally, *self-weakening* variables are defined as follows:

Definition 1 : [15] Let $X_1, \ldots, X_n$ denote n $0{-}1$ valued random variables. $X_1, \ldots, X_n$ are *self-weakening* with parameter λ if

$$Prob(X_{i1} \;=\; X_{i2} \;=\; \ldots \;=\; X_{ik} \;=\; 1) \;\leq\; \lambda \prod_{j=1}^{k} Prob(X_{ij} \;=\; 1).$$

for all distinct indices $i1, i2, \ldots, ik$ in the range $\{1, 2, \ldots, n\}$. A sufficient condition for this to hold with $\lambda = 1$ is that $\forall i : 1 \leq i \leq n$, and $\forall (j_1, \ldots, j_k) : \{j_1, \ldots, j_k\} \subseteq 1, 2, \ldots, i-1$:

$$Prob(X_i \;=\; 1 \mid X_{j_1} \;=\; X_{j_2} \;=\; \ldots \;=\; X_{j_k} \;=\; 1) \;\leq\; Prob(X_i \;=\; 1).$$

In [15] the following lemma is proved which shows the applicability of Chernoff-Hoeffding upper-tail bound to the case of *self-weakening* random variables.

Lemma 1 *[15]* Consider n $0-1$ random variables $X_1, \ldots, X_n$ which are self-weakening with parameter λ for some $\lambda > 0$. Let $X = \sum_{i=1}^{n} X_i$ and let $E(X) \leq \mu$ for some μ. Then

$$Prob(X > (1+\varepsilon)\mu) \;\leq\; \lambda F(\mu, \varepsilon).$$

Based on the above lemma and preceding discussion, it is clear that we can prove the case for $(c^2 N_a N_b)/n < 10 \log n$ if we can prove that $\mathcal{B}_{t,j}(t \in \mathcal{S})$ are *self-weakening*. We now argue that this is the case with $\lambda = 1$. Rename $\mathcal{B}_{t,j}(t \in \mathcal{S})$ as $X_1, \ldots, X_N$ for $N = |S|$. From the Definition 1, we must prove that $\forall i : 1 \leq i \leq N$ and $\forall (j1, \ldots, jk) : \{j1, \ldots, jk\} \subseteq \{1, 2, ..., i-1\}$,

$$Prob(X_i = 1 | X_{j1} = X_{j2} = \ldots = X_{jk} = 1) \;\leq\; Prob(X_i = 1) \;=\; b_j/n.$$

Pick any i, $1 \le i \le N$, and any $(j1, \ldots, jk) : \{j1, \ldots, jk\} \subseteq \{1, \ldots, i-1\}$. The cases where $k > b_j$, the condition $X_{j1} = X_{j2} = \ldots = X_{jk} = 1$ cannot hold so there is nothing to prove. In the case where $k = b_j$, $Prob(X_i = 1 | X_{j1} = \ldots = X_{jk} = 1) = 0$, since only b_j of the X_i's can be 1 and the condition is satisfied. Let us focus on the case where $k < b_j$.

$$\begin{aligned}
& Prob(X_i = 1 | X_{j1} = \ldots = X_{jk} = 1) \\
= \; & Prob(X_i = X_{j1} = \ldots = X_{jk} = 1) / Prob(X_{j1} = \ldots = X_{jk} = 1) \\
= \; & \frac{\dfrac{\binom{n-(k+1)}{b_j-(k+1)}}{\binom{n}{b_j}}}{\dfrac{\binom{n-k}{b_j-k}}{\binom{n}{b_j}}} \\
= \; & \frac{\binom{n-(k+1)}{b_j-(k+1)}}{\binom{n-k}{b_j-k}} \\
= \; & \frac{b_j-k}{n-k}
\end{aligned}$$

Since $\frac{b_j-k}{n-k} \le b_j/n$, for all $k \in \{0, \ldots, b_j - 1\}$ and for all $b_j \le n$, the random variables $\mathcal{B}_{t,j} (t \in \mathcal{S})$ are self weakening with parameter $\lambda = 1$ and the upper tail follows.

Now let us focus on the case when $(N_a N_b)/n > 10 \log n$. Let $\mu_u = (c^2 N_a N_b)/n$ and $\mu_l = \mu_u/c^2 = N_a N_b/n$. Using exactly the same argument as for the case when $N_a N_b/n < 10 \log n$ (applicability of Chernoff-Hoeffding bounds for binding the upper tail of self-weakening random variables) one can show that

$$Prob(c_{ij} > 2\mu_u) \;\le\; F(\mu_l, 1) \;\le\; (e/4)^{\mu_l} \;\le\; = n^{-5}.$$

The more interesting proof is to show a bound on the lower tail; i.e., to show that

$$Prob(c_{ij} < \mu_l/3) \;\le\; (0.7405)^{\mu_l} \;\le\; n^{-4}.$$

As before, fix any (i, j) and concentrate on the entry c_{ij} and let a_i denote the number of 1's in the i'th row of A and let $\mathcal{S}$ denote the set of indices t, for which a_{it} is 1. Assume that the adversary sets the number of 1's in the j'th column of B to be b_j. As before let $\mathcal{B}_{i,j}$ be indicator random variables for entries in b_{ij} and let $\mathcal{C}_{i,j}$ be the random variable denoting the value of c_{ij}. We therefore have

$$\mathcal{C}_{i,j} = \sum_{t \in \mathcal{S}} \mathcal{B}_{t,j}$$

and

$$E(\mathcal{C}_{i,j}) = \sum_{t \in \mathcal{S}} E(\mathcal{B}_{t,j}) = a_i b_j / n.$$

As before, if it were the case that the random variables $\mathcal{B}_{t,j}$ were independent binary random variables with expected value b_j/n, then the result would follow from the Chernoff-Hoeffding bounds for the lower tails, i.e.,

$$Prob(c_{ij} \ \leq \ (1-\varepsilon)\mu) \ \leq \ F(\mu, -\varepsilon),$$

where $\mu = a_i b_j/n$. To see that, note that

$$Prob(c_{ij} \ < \ \mu_l/3) \ < \ Prob(c_{ij} < \mu/3) \ \leq \ F(\mu, -2/3)$$

and

$$F(\mu, -2/3) \ \leq \ (0.7405)^{\mu} \ \leq \ (0.7405)^{\mu_l} \ \leq \ (0.7405)^{10 \log n} \ \leq \ n^{-4}.$$

However, the random variables $\mathcal{B}_{t,j}$ are not independent and even the *self-weakening* property of these random variables does not help in giving a bound for the lower tail. To prove our result we would need to revisit the proof of Lemma 1 itself. As we have noted earlier, the lower-tail Chernoff-Hoeffding bounds for binary random variables follow from the upper-tail Chernoff-Hoeffding bounds for the complementary variables. A careful examination of the proof of Lemma 1 in [15] shows that if one considers the complementary variables and that if the complementary variables have the *self-weakening* property then the Chernoff-Hoeffding bound holds for the upper tail of their sum. This bound on the upper-tail of the sum of these complemented variables translates to Chernoff-Hoeffding like bound on the lower-tail of the sum of the uncomplemented variables. In the special case when the self-weakening parameter is 1, this bound is exactly the Chernoff-Hoeffding bound for the lower tail of independent binary random variables.

We present a precise statement of this result as Lemma 2.

Lemma 2 Consider n $0-1$ random variables $X_1, \ldots, X_n$. Let $Y_1, \ldots, Y_n$ be the corresponding complementary binary random variables, i.e., $Y_i = 1 - X_i$. Let $X = \sum_{i=1}^{n} X_i$ and let $E[X] = \mu$ for some μ. If $Y_1, \ldots, Y_n$ are self-weakening with parameter $\lambda = 1$, then

$$Prob(X < (1-\varepsilon)\mu) \ \leq \ F(\mu, -\varepsilon)$$

Given the lemma above, all that remains to complete our proof is to show that the complementary variables $1 - \mathcal{B}_{t,j}(t \in \mathcal{S})$ are *self-weakening* with parameter 1.

Rename $\mathcal{B}_{t,j}(t \in \mathcal{S})$ as $X1, \ldots, X_N$ for $N = |S|$. Define the complementary variables $Y_1, \ldots, Y_N$ are $Y_i = 1 - X_i (1 \leq i \leq N)$.

From Definition 1, we must prove that $\forall i : 1 \leq i \leq N$ and $\forall (j1, \ldots, jk) : \{j1, \ldots, jk\} \subseteq \{1, \ldots, i-1\}$,

$$Prob(Y_i = 1 \mid Y_{j1} = \ldots = Y_{jk} = 1) \ \leq \ Prob(Y_i = 1) = 1 - b_j/n$$

Or in other words, $\forall i : 1 \leq i \leq N$, and $\forall (j1, \ldots, jk) : \{j1, \ldots, jk\} \ \subseteq \ \{1, \ldots, i-1\}$,

$$Prob(X_i = 0 \mid X_{j1} = ... = X_{jk} = 0) \ \leq \ Prob(X_i = 0) = 1 - b_j/n$$

Pick any i, $1 \leq i \leq N$, and any $(j1, \ldots, jk) : \{j1, \ldots, jk\} \ \subseteq \ \{1, \ldots, i-1\}$. The cases where $k > n - b_j$, the condition $X_{j1} = \ldots = X_{jk} = 0$ cannot hold so there is nothing to prove. In the case where $k = n - b_j$, $Pr(X_i = 0 \mid X_{j1} = \ldots = X_{jk} = 0) = 0$, since only $n - b_j$ of the X_i's can be 0 and the condition is satisfied. Let us focus on the case where $k < n - b_j$.

$$
\begin{aligned}
& Prob(X_i = 0 \mid X_{j1} = \ldots = X_{jk} = 0) \\
= \; & \frac{Prob(X_i = X_{j1} = \ldots = X_{jk} = 0)}{Prob(X_{j1} = \ldots = X_{jk} = 0)} \\
= \; & \frac{\begin{pmatrix} n-(k+1) \\ b_j \end{pmatrix}}{\begin{pmatrix} n-k \\ b_j \end{pmatrix}} \\
= \; & (n-k-b_j)/(n-k) \\
= \; & 1 - b_j/(n-k).
\end{aligned}
$$

Since $k \geq 0$ and $b_j \leq n$, $(1 - b_j/(n-k)) \;\leq\; (1 - b_j/n)$. Hence the complementary random variables $Y_1, \ldots, Y_N$ are *self-weakening* with parameter $\lambda = 1$ and the lower tail follows. This completes the proof of Theorem 3.

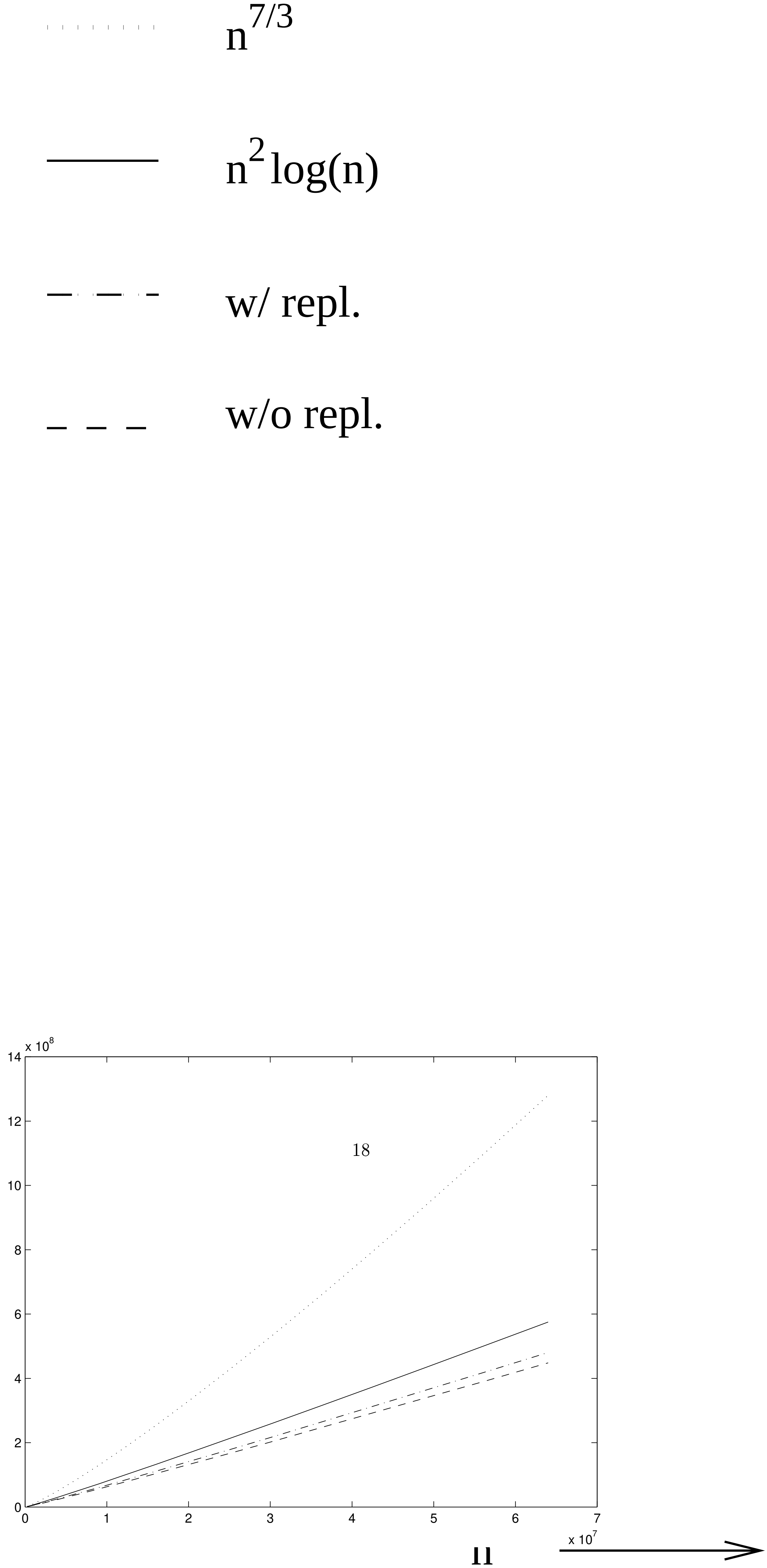

Figure 2: Maximum number of operations performed by the hybrid algorithms vs. the matrix size.