INFORMATION AND CONTROL 22, 132–138 (1973)

A Fast Expected Time Algorithm for Boolean Matrix Multiplication and Transitive Closure

PATRICK E. O'NEIL*

Massachusetts Institute of Technology, Department of Electrical Engineering, Cambridge, Massachusetts

AND

ELIZABETH J. O'NEIL

University of Massachusetts, Department of Mathematics, Boston, Massachusetts

A probabilistic algorithm is presented to calculate the Boolean product of two $n \times n$ Boolean matrices using an expected number of elementary operations of $0(n^2)$. Asymptotically in n, almost all pairs of matrices may be multiplied using this algorithm in $0(n^{2+\epsilon})$ elementary operations for any $\epsilon > 0$.

I. INTRODUCTION

We define the binary operators $\vee$ and $\wedge$, called Boolean addition and Boolean multiplication respectively, on the set $\{0, 1\}$. If $a, b \in \{0, 1\}$

$$a \vee b = \begin{cases} 0 \text{ if } a = 0 \text{ and } b = 0 \\ 1 \text{ otherwise} \end{cases}$$

$$a \wedge b = \begin{cases} 1 \text{ if } a = 1 \text{ and } b = 1 \\ 0 \text{ otherwise.} \end{cases}$$

Given two $n \times n$ $(0, 1)$–matrices A and B, the Boolean product, AB, is an $n \times n$ matrix C such that

$$C_{ij} = \bigvee_{1 \leqslant k \leqslant n} (A_{ik} \wedge B_{kj}). \tag{1.1}$$

* The work of this author was supported by the Office of Naval Research Grants N00014-67-A-0204-0034 and N00014-70-A-0362-002.

In what follows, the products of matrices shall be Boolean products unless otherwise specified.

Given a relation R on n objects, $\{1, 2,..., n\}$, we speak of its Boolean companion matrix A: $A(i, j) = 1$ iff iRj. Note that the relation R may not be transitive, so it is possible that iRk and kRj but not iRj. We wish to find the transitive closure of R, R^*, defined as the relation with the minimum number of related pairs which contains R and is transitive. If R has companion matrix A we speak also of the transitive closure of the matrix A, A^*, which is the companion matrix of R^*. It is easily shown [see Furman (1970)] that $A^* = A(I \vee A)^k$, for any $k \geqslant n - 1$.

The calculation of $A(I \vee A)^k$, $k \geqslant n - 1$ may be done using successive squaring in $O(\log_2 n)$ Boolean matrix multiplications. From this it is immediate:

Remark 1.1. If the Boolean product of two $n \times n$ matrices is computable in $O(n^\beta)$ elementary operations (e.g. additions, multiplications, comparisons) we may find the transitive closure of any $n \times n$ Boolean matrix A in $O(n^\beta \cdot \log_2 n)$ elementary operations.

An improvement of Munro (1971) lowers the number of elementary operations to find the transitive closure to $O(n^\beta)$. Fischer and Meyer (1971) demonstrate the converse: that if the transitive closure is computable in $O(n^\beta)$ operators, then so is the Boolean product.

At present there is only one algorithm known to multiply Boolean matrices in $O(n^\beta)$ operations, $\beta < 3$. It has been observed [see Fisher and Meyer (1971), Furman (1970), Munro (1971)] that two Boolean matrices may be multiplied in $O(n^{\log_2 7})$ operations. One uses the method of Strassen (1969) to obtain the real integer product of A and B and normalizes the result by replacing all nonzero entries with 1.

In this paper, we present a new algorithm for the Boolean multiplication of matrices. The algorithm is probabilistic in nature; the number of elementary operations it performs in multiplying two $n \times n$ matrices A and B depends explicitly on the structure of A and B. For randomly chosen Boolean matrices A and B, we show that this algorithm is performed with an expected number of elementary operations of $O(n^2)$. Since the Strassen algorithm requires $cn^{\log_2 7}$ operations, and $\log_2 7 \sim 2.8$, this is a substantial improvement in expected operating time. A worse case example is constructed which requires this algorithm to perform cn^3 operations. Such cases are rare; asymptotically in n, almost all Boolean matrix products are performed more quickly by this algorithm than by Strassen's.

We present this algorithm with an eye toward actual computer applications.

It is easily coded and requires very little extra storage. The prospective user is advised to "remove" all cycles in the directed graph of a relation R if his intention is to calculate R^*. This may be done in $O(n^2)$ operations [see Munro (1971)]. Since the algorithm merges the vertices of a cycle to a single vertex, it may have the effect of tremendously reducing the problem.

II. The Algorithm

Given two $n \times n$ (0, 1)–matrices A and B, we wish to perform the Boolean multiplication $AB = C$. First calculate n_A and n_B, the number of 1s in the matrices A and B respectively. If $n_A > n_B$, one should use the following algorithm to perform the multiplication $B^tA^t = C^t$, thus assuring that the matrix with a lower density of 1s will be on the left hand side of the product. This precalculation requires only $O(n^2)$ operations. For notation purposes, we assume that the problem is still the multiplication $AB = C$.

For each row i of the matrix A, prepare a list, $k_1{}^i < k_2{}^i < \cdots < k^i_{L(i)}$, of positions in the row where a 1 occurs. That is, $k_j{}^i$ is in the list the entry of A (i, $k_j{}^i$) is a 1; clearly $L(i)$ is the number of 1s in row i. It is only necessary to hold in storage the calculated list for one row of A at a time, passing on to the next when the values for the corresponding row of C have been determined; it is clear that the number of operations needed to find all of these lists is $O(n^2)$.

Given the list $k_1{}^i < k_2{}^i < \cdots < k^i_{L(i)}$, we may calculate the value C_{ij} as follows. Set $k = k_1{}^i$ and examine the entry B_{kj}. Now proceed inductively: If $B_{kj} = 1$, set $C_{ij} = 1$; otherwise set k to the next element in the list and repeat this step. If at some point the elements of the list have been exhausted without setting C_{ij} to 1, then set $C_{ij} = 0$. Repeat this entire procedure for each value of j, $1 \leqslant j \leqslant n$, until the entire i–th row of C has been determined. Proceed to a new row i until all the rows $1 \leqslant i \leqslant n$ have been considered; at this point the matrix C will be determined.

It is clear that this is a very straightforward algorithm which takes advantage of a peculiar feature of Boolean arithmetic in taking vector inner products; once a pair of 1s have been found in matching positions of the two vectors A_{i*} and B_{*j}, we need proceed no further, since we only require one such pair to set the product to 1.

Consider the following pair of matrices, A and B.

$$A_{ij} = \begin{cases} 1 \text{ if } j \text{ is even} \\ 0 \text{ if } j \text{ is odd;} \end{cases} \qquad B_{ij} = \begin{cases} 1 \text{ if } i \text{ is odd} \\ 0 \text{ if } i \text{ is even.} \end{cases}$$

It is clear that the product AB is a matrix which is zero in all entries, and moreover that the algorithm we have presented will execute cn^3 operations in multiplying A and B. Thus, a worse case analysis is disappointing. In the next section, however, we show that for "random" matrices A and B, the expected number of operations to form the product AB using this algorithm is $O(n^2)$.

III. Analysis of the Algorithm

We begin this section with a definition.

Definition. A random Boolean matrix of density p is an $n \times n$ matrix X whose entries X_{ij} are independent random variables, taking on the value 1 with probability p and the value 0 with probability $q = 1 - p$.

It should be emphasized that a random Boolean matrix X is not a Boolean matrix, since its entries are random variables. Any Boolean matrix A is a possible *value* of the random Boolean matrix X and has an associated probability.

Remark 3.1. Let X be a random Boolean matrix of density p, and let A be a Boolean matrix which contains m 1s. Then the probability P that X takes on the value A is given by:

$$P = p^m q^{n^2-m} \tag{3.1}$$

Proof. This follows immediately from the definition, since m random variables X_{ij} are required to take on the value 1 (each does so with probability p), and $n^2 - m$ random variables take on the value 0, each with probability q.

It is clear that a random Boolean matrix X of density p imposes a probabilistic sample space on the set of Boolean matrices. The probability assigned to each specific matrix is given by Remark 3.1.

Remark 3.2. In this sample space all matrices containing exactly m 1s, for any m, are equally probable.

Proof. Immediate from Remark 3.1.

We shall derive the expected number of operations needed by the algorithm of Section II to multiply two Boolean matrices A and B, where A is a value of the random Boolean matrix X of density p and B is a value of the random Boolean matrix Y of density p'. We assume $p \leqslant p'$, i.e. the lower density

matrix should be on the left in the multiplication; this was provided for in the algorithm.

Let us find the expected number, E_o, of operations to take the Boolean inner product of the two vectors A_{i*} and B_{*j}, thus determining C_{ij}. We may add these E_o for each $i, j = 1,..., n$ to find the expected number of operations to calculate the product $C = AB$. This may be stated as follows:

Remark 3.3. The expected number of operations to find the product AB is given by $n^2E_o + O(n^2)$. The second term takes account of the operations we perform in precalculation.

In computing the Boolean inner product of A_{i*} and B_{*j}, we may take advantage of the precalculation of the positions of 1s in A_{i*}, given by the list $k_1{}^i < k_2{}^i < \cdots < k_j{}^i$. Here we assume $j = L(i)$. We shall deal with this assumption later.

Clearly, the expected number of operations E_o to multiply A_{i*} and B_{*j} will be proportional to the expected number of times the entries of the vector B_{*j} corresponding to members of the list $k_1{}^i < k_2{}^i < \cdots < k_j{}^i$ are accessed. Referring to the algorithm of Section II we see that the probability that the first such entry is accessed is 1; the probability that the second entry is accessed is equal to the probability that the first entry accessed in B_{*j} is 0 and this is given by $1 - p'$. In general, the probability that the m–th entry is accessed is given by $(1 - p')^{m-1}$. Since the number of accesses to the m–th entry is either 0 or 1, the expected number of accesses to the m–th entry of B_{*j} is $(1 - p')^{m-1}$. The expected total number of accesses is the sum of the expected number of accesses in each possible location and is therefore given by:

$$\sum_{m=1}^{j} (1 - p')^{m-1} = (1 - (1 - p')^j)/p'. \tag{3.2}$$

Now we have designated the number of 1s in the i–th row of A by j. In actuality, this number is a random variable, and is equal to j with probability $\binom{n}{j} p^j(1 - p)^{n-j}$. To complete our calculation of the number E_o of operations to determine C_{ij}, we need merely calculate the sum

$$E_0 = \sum_{j=0}^{n} \binom{n}{j} p^j(1 - p)^{n-j}[(1 - (1 - p')^j)/p'], \qquad 0 < p \leqslant p' < 1. \tag{3.3}$$

Thus, manipulation of the sum (3.3) yields

$$\begin{aligned} E_0 &= 1/p' - \left[(1 - p)^n/p' \sum_{j=0}^{n}\right] \binom{n}{j} [p(1 - p')/(1 - p)]^j \\ &= (1 - (1 - pp')^n)/p' \end{aligned} \tag{3.4}$$

THEOREM 3.1. *Under the assumption that all Boolean matrices are equally likely choices for A and B, the expected number of operations to perform the multiplication AB is $O(n^2)$.*

Proof. By Remark 3.1, if all entries of a random matrix take on the values 0 or 1 with equal probability $\frac{1}{2}$, then all matrices occur with equal probability 2^{-n^2}. By Remark 3.3 we need merely show that $n^2E_o = O(n^2)$, when $p = p' = \frac{1}{2}$. By Eq. (3.4), $E_o = 2(1 - (\frac{3}{4})^n) = O(1)$, and the proof is complete.

COROLLARY. *With the above assumption almost all matrix products AB may be calculated in $O(n^{2+\epsilon})$ operations, for any $\epsilon > 0$.*

Proof. By Theorem 1, the relative proportions of pairs of matrices which require more than $n^{2+\epsilon}$ operations for their multiplication must be $O(n^{-\epsilon})$.

A more involved calculation under the assumption of Theorem 1 reveals that, asymptotically in n, the value of n^2E_o is normally distributed about $2n^2$, with a standard deviation of $\sqrt{2} \cdot n$. Clearly any matrix products requiring $cn^{2+\epsilon}$ operations, with $\epsilon > 0$, are pathological examples.

In some applications, there may be reason to believe that the matrices A and B will not have a density of ones equal to $\frac{1}{2}$. If the probabilities of 1s in entries of A and B are independent and take on values p and p' respectively, $0 < p \leqslant p' < 1$, then Remark 3.3 and Eq. (3.4) will suffice to calculate the expected number of operations to take the product AB. For some very small values of p and p', the expected number of operations to take the product may approach $n^{5/2}$.

Using Eq. (3.4), we can show that the maximum value of E_o over the region $0 < p \leqslant p' < 1$ will occur when $p = p' = xn^{-1/2}$, where x is the solution to the equation $1 + 2x^2 = e^{x^2}$. Approximating, the maximum of E_o is $0.23n^{1/2}$ when $p = p' = 1.12n^{-1/2}$. Thus, for $0 < p \leqslant p' < 1$,

$$E_o = O(n^{1/2}). \tag{3.5}$$

THEOREM 3.2. *If the probabilities of occurrences of* 1s *in entries of A and B are independent and take on values p and p' respectively, $0 < p \leqslant p' < 1$, p and p' possibly dependent on n, then the expected number of operations needed to find the product AB is $O(n^{5/2})$. Furthermore, this estimate is sharp since for $p = p' = 1.12n^{-1/2}$, the expected number is $0.23n^{5/2}$.*

Proof. Immediate from Eq. (3.5), Remark 3.3 and the above discussion.

RECEIVED: January 13, 1972

643/22/2-3

References

FISCHER, M. J., AND MEYER, A. R. (1971), Boolean matrix multiplication and transitive closure, *I.E.E.E. Twelfth Symp. on Switching and Automata.*

FURMAN, M. E. (1970), Application of a method of fast multiplication of matrices in the problem of finding the transitive closure of a graph, *Dokl. Akad. Nauk SSSR*194 **3**, and *Soviet Math. Dokl.* 11 **5**, 1252.

MUNRO, I. (1971), Efficient determination of the strongly connected components and transitive closure of a directed graph, *Information Processing Letters* 1 **2**, 56–58.

STRASSEN, V. (1969), Gaussian elimination is not optimal, *Numer. Math.* **13**, 354–356.