A PRACTICAL ALGORITHM FOR BOOLEAN MATRIX MULTIPLICATION

by M.D. Atkinson & N. Santoro

SCS-TR-116

June 1987

School of Computer Science
Carleton University
Ottawa, Ontario
CANADA K1S 5B6

This research was supported by the National Science and Engineering Research Council of Canada.

A PRACTICAL ALGORITHM FOR BOOLEAN MATRIX MULTIPLICATION

M. D. Atkinson and N. Santoro

School of Computer Science
Carleton University
Ottawa, K1S 5B6
CANADA

Abstract. An algorithm is given for multiplying two nxn Boolean matrices. It has time complexity $O(n^3/(\log n)^{1.5})$ and requires $n \log_2 n$ bits of auxiliary storage.

Keywords: Boolean matrix multiplication, time *vs* storage tradeoffs

Research on fast methods for multiplying two nxn Boolean matrices has followed two directions. On the one hand there are the asymptotically fast methods derived from algorithms to multiply matrices with integer entries. On the other hand, it has been of interest to devise algorithms which, while inferior in the asymptotic sense, are simple enough to be advantageously implemented. This paper addresses the latter class of algorithms. The first such algorithm (apart from the classical straightforward $O(n^3)$ algorithm) was given in [3] and is known, at least in Western literature, as the Four Russians' algorithm. It requires time $O(n^3/\log n)$ and auxiliary storage of $O(n^3/\log n)$ bits; simple modifications can reduce the amount of auxiliary storage to $O(n^2)$. More recent work on the problem has focussed on reducing the amount of auxiliary storage [5] and on exploiting properties of the arrays to be multiplied [4,6].

The purpose of this note is to point out that a rather simple and known technique can be used to derive a Boolean matrix multiplication algorithm which has a lower time complexity and requires less auxiliary storage than the Four Russians' method. This technique was originally proposed to compute matrix products over the field of integers modulo two [2, problem 6.16], and has been employed to reduce the complexity of Boolean implementations of asymptotically fast matrix multiplication algorithms [1].

The essential idea is to partition each of the nxn matrix factors into $(n/k)^2$ kxk square submatrices, calculate the product by block matrix multiplication using $(n/k)^3$ multiplications and

$(n/k)^3$ additions of kxk matrices, and do these matrix multiplications and additions by table look-up. Specifically the algorithm is as follows:

Algorithm Multiply
(A, B : nxn input Boolean matrices; C : the matrix product AxB);

Choose a suitable submatrix block size k;

for every kxk matrix P and every kxk matrix Q
Product [P, Q] := PxQ;
Sum [P, Q] := P+ Q;
end for;

Partition each of A and B into $(n/k)^2$ kxk submatrices A_{ij} and B_{ij} and build the corresponding partition of the product by:

for every pair (i,j) with $1 \leq i,j \leq n/k$
$C_{ij} := 0$;
for every r with $1 \leq r \leq n/k$
$C_{ij} :=$ *Sum* [C_{ij}, *Product* [A_{ir},B_{rj}]]
end for;
end for

The two arrays *Product* and *Sum* are slightly unusual in having subscripts which are themselves square Boolean arrays. However, since arrays are generally stored as vectors (row major or column major storage) it is not difficult to interpret a Boolean array as an integer row subscript or column subscript into the arrays *Product* and *Sum*. An implementation in a high-level language might adopt the policy of encoding every kxk submatrix of A and B as an integer and representing *Product* and *Sum* as integer arrays with integer subscripts.

Proposition The algorithm requires $2k^2 4^k$ bits of auxiliary storage and has time complexity $O(k^3 4^k + (n/k)^3)$

Proof. Each of the tables *Product* and *Sum* has 2^k rows and 2^k columns, and so has 4^k entries each of which is a kxk Boolean matrix. Thus *Product* and *Sum* occupy a total of $2k^2 4^k$ bits of storage. To build the table *Product* requires 4^k kxk Boolean matrix products to be calculated. The time required for this is $O(k^3 4^k)$. Similarly the construction of *Sum* requires time $O(k^2 4^k)$. The final part of the algorithm accesses the tables *Product* and *Sum* $(n/k)^3$ times and each access requires constant time. The other execution time costs (such as the matrix initialisations $C_{ij} := 0$, and the placing of each C_{ij} in the correct position in the output array) are all $O(n^2)$. []

The choice of the parameter k offers a tradeoff between time and space complexity. Taking $k=\sqrt{\log_4 n}$ results in an auxiliary space requirement of $2n \log_4 n = n \log_2 n$ bits. For this choice of k the time complexity of the algorithm is easily seen to be

$$c_k \cdot c_A (n^3/(\log_2 n)^{1.5}) + \text{lower order terms},$$

where $c_k = \sqrt{8} = 2.828...$, and c_A is the (implementation-dependent) constant representing the time required to perform an access of the form *Sum* [X, *Product* [Y,Z]]. The value c_k can be reduced by taking k to be larger. For example $k = \sqrt{\log_2 n}$ yields $c_k = 1$ and still results in the modest space requirement of $2n^2 \log_2 n$ bits of auxiliary storage.

This technique can obviously be applied to other small algebras apart from the {0,1} Boolean algebra, as in the case described in [2, problem 6.16]. For example, if the entries of the matrices to be multiplied are integers in Z_m for some small modulus m, we could construct the multiplication and addition tables for kxk matrices over Z_m; taking $k=\sqrt{(\log_m n)/2}$ these would occupy just $2m^{2k} k^2 = n \log_m n$ integer locations of storage (or $n \log_2 n$ bits of storage). Once these tables are available the nxn matrix product can be found in $n^3/k^3 = 2.828\, n^3/(\log_m n)^{1.5}$ table accesses of the type *Sum* [X, *Product* [Y,Z]].

Also in this case a trade-off between the space and the (multiplicative constant in the) time complexity is implied from the choice of parameter k.

Acknowledgment

This research was supported by the National Science and Engineering Research Council of Canada under Grants A2419 and A2415.

References

[1] L. Adleman, K.S. Booth, F.P. Preparata, and W.L. Ruzzo, "Improved time and space bounds for Boolean matrix multiplication, Acta Informatica 11 (1978), 61-75.

[2] A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design and Analysis of Computer Algorithms (Addison-Wesley, Reading, MA, 1974).

[3] V.L. Arlazarov, E.A. Dinic, M.A. Kronrod, and I.A. Faradzev, On economical construction of the transitive closure of a directed graph, Dokl. Akad. Nauk SSSR 194 (1970), 487-488 (in Russian).

[4] N. Santoro, Four $O(N^2)$ multiplication methods for sparse and dense Boolean matrices, Proc. 10th Conf. on Numerical Mathematics and Computing, Congressus Numerantium Vol. 31 (Utilitas Mathematica, Winnipeg, 1981) 241-252.

[5] N. Santoro and J. Urrutia, An improved algorithm for Boolean matrix multiplication, Computing 36 (1986), 375-382.

[6] J. Vyskoc, A note on Boolean matrix multiplication, Inf. Proc. Letters 19 (1984), 249-251.

Carleton University, School of Computer Science
Bibliography of Technical Reports
Publications List (1985 -->)

School of Computer Science
Carleton University
Ottawa, Ontario, Canada
KIS 5B6

SCS-TR-66 **On the Futility of Arbitrarily Increasing Memory Capabilities of Stochastic Learning Automata**
_____ B.J. Oommen, October 1984. Revised May 1985.

SCS-TR-67 **Heaps in Heaps**
_____ T. Strothotte, J.-R. Sack, November 1984. Revised April 1985.

SCS-TR-68 **Partial Orders and Comparison Problems**
out-of-print M.D. Atkinson, November 1984. See Congressus Numerantium 47 ('86), 77-88

SCS-TR-69 **On the Expected Communication Complexity of Distributed Selection**
_____ N. Santoro, J.B. Sidney, S.J. Sidney, February 1985.

SCS-TR-70 **Features of Fifth Generation Languages: A Panoramic View**
_____ Wilf R. LaLonde, John R. Pugh, March 1985.

SCS-TR-71 **Actra: The Design of an Industrial Fifth Generation Smalltalk System**
_____ David A. Thomas, Wilf R. LaLonde, April 1985.

SCS-TR-72 **Minmaxheaps, Orderstatisticstrees and their Application to the Coursemarks Problem**
_____ M.D. Atkinson, J.-R. Sack, N. Santoro, T. Strothotte, March 1985.

SCS-TR-73 **Designing Communities of Data Types**
Wilf R. LaLonde, May 1985.
Replaced by SCS-TR-108

SCS-TR-74 **Absorbing and Ergodic Discretized Two Action Learning Automata**
out-of-print B. John Oommen, May 1985. See IEEE Trans. on Systems, Man and Cybernetics, March/April 1986, pp. 282-293.

SCS-TR-75 **Optimal Parallel Merging Without Memory Conflicts**
_____ Selim Akl and Nicola Santoro, May 1985

SCS-TR-76 **List Organizing Strategies Using Stochastic Move-to-Front and Stochastic Move-to-Rear Operations**
_____ B. John Oommen, May 1985.

SCS-TR-77 **Linearizing the Directory Growth in Order Preserving Extendible Hashing**
_____ E.J. Otoo, July 1985.

SCS-TR-78 **Improving Semijoin Evaluation in Distributed Query Processing**
_____ E.J. Otoo, N. Santoro, D. Rotem, July 1985.

Carleton University, School of Computer Science
Bibliography of Technical Reports

SCS-TR-79 _____
On the Problem of Translating an Elliptic Object Through a Workspace of Elliptic Obstacles
B.J. Oommen, I. Reichstein, July 1985.

SCS-TR-80 _____
Smalltalk - Discovering the System
W. LaLonde, J. Pugh, D. Thomas, October l985.

SCS-TR-81 _____
A Learning Automation Solution to the Stochastic Minimum Spanning Circle Problem
B.J. Oommen, October 1985.

SCS-TR-82 _____
Separability of Sets of Polygons
Frank Dehne, Jörg-R. Sack, October 1985.

SCS-TR-83 out-of-print
Extensions of Partial Orders of Bounded Width
M.D. Atkinson and H.W. Chang, November 1985. See Congressus Numerantium, Vol. 52 (May 1986), pp. 21-35.

SCS-TR-84 _____
Deterministic Learning Automata Solutions to the Object Partitioning Problem
B. John Oommen, D.C.Y. Ma, November 1985

SCS-TR-85 out-of-print
Selecting Subsets of the Correct Density
M.D. Atkinson, December 1985. To appear in Congressus Numerantium, Proceedings of the 1986 South-Eastern conference on Graph theory, combinatorics and Computing.

SCS-TR-86 _____
Robot Navigation in Unknown Terrains Using Learned Visiblity Graphs. Part I: The Disjoint Convex Obstacles Case
B. J. Oommen, S.S. Iyengar, S.V.N. Rao, R.L. Kashyap, February 1986

SCS-TR-87 _____
Breaking Symmetry in Synchronous Networks
Greg N. Frederickson, Nicola Santoro, April 1986

SCS-TR-88 _____
Data Structures and Data Types: An Object-Oriented Approach
John R. Pugh, Wilf R. LaLonde and David A. Thomas, April 1986

SCS-TR-89 _____
Ergodic Learning Automata Capable of Incorporating Apriori Information
B. J. Oommen, May 1986

SCS-TR-90 _____
Iterative Decomposition of Digital Systems and Its Applications
Vaclav Dvorak, May 1986.

SCS-TR-91 _____
Actors in a Smalltalk Multiprocessor: A Case for Limited Parallelism
Wilf R. LaLonde, Dave A. Thomas and John R. Pugh, May 1986

SCS-TR-92 _____
ACTRA - A Multitasking/Multiprocessing Smalltalk
David A. Thomas, Wilf R. LaLonde, and John R. Pugh, May 1986

SCS-TR-93 _____
Why Exemplars are Better Than Classes
Wilf R. LaLonde, May 1986

SCS-TR-94 _____
An Exemplar Based Smalltalk
Wilf R. LaLonde, Dave A. Thomas and John R. Pugh, May 1986

SCS-TR-95 _____
Recognition of Noisy Subsequences Using Constrained Edit Distances
B. John Oommen, June 1986

Carleton University, School of Computer Science
Bibliography of Technical Reports

SCS-TR-96 **Guessing Games and Distributed Computations in Synchronous Networks**
J. van Leeuwen, N. Santoro, J. Urrutia and S. Zaks, June 1986.

SCS-TR-97 **Bit vs. Time Tradeoffs for Distributed Elections in Synchronous Rings**
M. Overmars and N. Santoro, June 1986.

SCS-TR-98 **Reduction Techniques for Distributed Selection**
N. Santoro and E. Suen, June 1986.

SCS-TR-99 **A Note on Lower Bounds for Min-Max Heaps**
A. Hasham and J.-R. Sack, June 1986.

SCS-TR-100 **Sums of Lexicographically Ordered Sets**
M.D. Atkinson, A. Negro, and N. Santoro, May 1987.

SCS-TR-102 **Computing on a Systolic Screen: Hulls, Contours, and Applications**
F. Dehne, J.-R. Sack and N. Santoro, October 1986.

SCS-TR-103 **Stochastic Automata Solutions to the Object Partitioning Problem**
B.J. Oommen and D.C.Y. Ma, November 1986.

SCS-TR-104 **Parallel Computational Geometry and Clustering Methods**
F. Dehne, December 1986.

SCS-TR-105 **On Adding *Constraint Accumulation* to Prolog**
Wilf R. LaLonde, January 1987.

SCS-TR-107 **On the Problem of Multiple Mobile Robots Cluttering a Workspace**
B. J. Oommen and I. Reichstein, January 1987.

SCS-TR-108 **Designing Families of Data Types Using Exemplars**
Wilf R. LaLonde, February 1987.

SCS-TR-109 **From Rings to Complete Graphs - Θ(n logn) to Θ(n) Distributed Leader Election**
Hagit Attiya, Nicola Santoro and Shmuel Zaks, March 1987.

SCS-TR-110 **A Transputer Based Adaptable Pipeline**
Anirban Basu, March 1987.

SCS-TR-111 **Impact of Prediction Accuracy on the Performance of a Pipeline Computer**
Anirban Basu, March 1987.

SCS-TR-112 **ε-Optimal Discretized Linear Reward-Penalty Learning Automata**
B.J. Oommen and J.P.R. Christensen, May 1987.

SCS-TR-113 **Angle Orders, Regular n-gon Orders and the Crossing Number of a Partial Order**
N. Santoro and J. Urrutia, June 1987.

SCS-TR-115 **Time is Not a Healer: Impossibility of Distributed Agreement in Synchronous Systems with Random Omissions**
N. Santoro, June 1987.

SCS-TR-116 **A Practical Algorithm for Boolean Matrix Multiplication**
M.D. Atkinson and N. Santoro, June 1987.

Carleton University, School of Computer Science
Bibliography of Technical Reports

SCS-TR-117 **Recognizing Polygons, or How to Spy**
James A. Dean, Andrzej Lingas and Jörg-R. Sack, August 1987.

SCS-TR-118 **Stochastic Rendezvous Network Performance - Fast, First-Order Approximations**
J.E. Neilson, C.M. Woodside, J.W. Miernik, D.C. Petriu, August 1987.