Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 39 (2014)

Vector convolution in $O(n)$ steps and matrix multiplication in $O(n^2)$ steps :-)

Andrzej Lingas[1] * and Dzmitry Sledneu[2]

[1] Department of Computer Science, Lund University
`Andrzej.Lingas@cs.lth.se`
[2] Centre for Mathematical Sciences, Lund University
`Dzmitry.Sledneu@math.lu.se`

Abstract. We observe that if we allow for the use of division and the floor function besides multiplication, addition and subtraction then we can compute the arithmetic convolution of two n-dimensional integer vectors in $O(n)$ steps and perform the arithmetic matrix multiplication of two integer $n \times n$ matrices in $O(n^2)$ steps. Hence, any method for proving superlinear (in the input size) lower bounds for the aforementioned problems has to assume a more restricted set of arithmetic operations and/or an upper bound on the size of allowed integers.

1 Introduction

Two $n \times n$ integer matrices can be multiplied using $O(n^3)$ additions and multiplications following the definition of matrix product. Similarly, the convolution of two n-dimensional vectors can be computed using $O(n^2)$ additions and multiplications. Both are optimal if neither other operations nor negative constants are allowed [6, 8, 10]. If additionally subtraction or negative constants are allowed then the so called fast matrix multiplication algorithms can be implemented using $O(n^{\omega})$ operations [3, 11, 14], where $\omega < 3$. They rely on algebraic equations following from the possibility of term cancellation. Vassilevska has recently shown the exponent ω of fast matrix multiplication to be smaller than 2.373 in [14]. Next, the convolution of two n-dimensional vectors over a commutative ring with the so called principal n-th root of unity can be computed via Fast Fourier Transform using $O(n \log n)$ operations of the ring (section 7.2 in [1], for bit complexity of FFT see section 7.3). On the other hand, Raz proved that if only addition, multiplication and products with constants of absolute value not exceeding one are allowed then $n \times n$ matrix multiplication requires $\Omega(n^2 \log n)$ operations [9].

Yval was first to describe a reduction of the distance matrix product (equivalently, the $(\min, +)$ matrix product) of two $n \times n$ matrices to matrix multiplication of two $n \times n$ matrices over a ring, using $O(n^2)$ operations [13] (cf. [12]). The idea of the reduction is relatively simple [2, 15]. Two input $n \times n$ matrices $A = (a_{i,j})$ and $B = (b_{i,j})$ with integer entries in $[-M, M]$ are transformed to two

* Research supported in part by Swedish Research Council grant 621-2011-6179.

$n \times n$ matrices $A' = ((n+1)^{M-a_{i,j}})$ and $B' = ((n+1)^{M-b_{i,j}})$. It is not too difficult to see that if $C = (c_{i,j})$ is the distance product of A and B and $C' = (c'_{i,j})$ is the arithmetic matrix product of A' and B', then $c_{i,j} = 2M - \lfloor \log c'_{i,j} \rfloor$. Note that the reduction uses the exponentiation, logarithm and floor functions besides the arithmetic ring operations.

By combining the reduction with fast matrix multiplication, we obtain an algorithm for the distance matrix product, using $O(n^{\omega})$ multiplications, additions and subtractions, and $O(n^2)$ exponentiation, logarithm and floor operations.

Since the entries in the transformed matrices A', B' are huge numbers that require $O(M)$ computer words of $\log n$ bits each, the matrix multiplication of A' and B' requires $O(Mn^{\omega})$ algebraic operations on $O(\log n)$ bit numbers [2, 15]. For this reason, the described algorithm for distance matrix product is interesting solely for smaller values of M and approximation purposes [15].

Recently, also a nondeterministic algorithm for $n \times n$ matrix multiplication using $O(n^2)$ arithmetic operations has been presented [5]. It results from a derandomization of Freivalds' randomized algorithm for matrix product verification [4]. Simply, the algorithm guesses first the product matrix and then verifies its correctness. Again, the derandomization involves huge numbers requiring $O(n)$ times more bits than the input numbers [5].

In this short paper, we observe that if we allow for the use of division and the floor function (or exponentiation, logarithm and the floor function) besides multiplication, addition and subtraction then we can compute the arithmetic convolution of two n-dimensional integer vectors in $O(n)$ steps and perform the arithmetic matrix multiplication of two integer $n \times n$ matrices in $O(n^2)$ steps. Similarly, as in the case of the reduction of distance matrix product to the arithmetic one, the trick is to use numbers requiring about n times more bits than any entry in the input matrices. If we combine the reduction with our algorithm for matrix multiplication, we obtain an algorithm for the distance matrix product using solely $O(n^2)$ operations on huge numbers requiring about Mn words of $\log n$ bits each. Our results also indicate that any method for proving superlinear (in the input size) lower bounds for the aforementioned problems has to assume a more restricted set of arithmetic operations and/or an upper bound on the size of allowed integers.

2 Preliminaries

For two n-dimensional vectors with integer coordinates $a = (a_0, ..., a_{n-1})$ and $b = (b_0, ..., b_{n-1})$ their dot product $\sum_{i=0}^{n-1} a_i b_i$ is denoted by $a \odot b$. The convolution of the vectors a and b is a vector $c = (c_0, ..., c_{2n-2})$, where $c_i = \sum_{l=\max\{i-n+1,0\}}^{\min\{i,n-1\}} a_l b_{i-l}$ for $i = 0, ..., 2n-2$. Note that the $(n-1)$-th coordinate c_{n-1} of the convolution is equal to $a \odot b^R$, where $b^R = (b_{n-1}, ..., b_0)$.

For an integer $n \times n$ matrix $A = (a_{i,j})$, its i-th row $(a_{i,1}, ..., a_{i,n})$ is denoted by $A_{i,*}$. Similarly, the j-th column $(a_{1,j}, ..., a_{n,j})$ of A is denoted by $A_{*,j}$. Given another integer $n \times n$ matrix B, the matrix product $A \times B$ of A with B is a matrix $C = (c_{i,j})$, where $c_{i,j} = A_{i,*} \odot B_{*,j}$ for $1 \leq i, j \leq n$.

3 The algorithms

For an n-dimensional vector $a = (a_0, ..., a_{n-1})$ with integer coordinates let $a(x)$ denote the polynomial $\sum_{k=0}^{n-1} a_k x^k$. The following lemma is folklore (see section 7.4 in [1]).

Lemma 1. *For $k = 0, ..., 2n-2$, the k-th coordinate c_k of the convolution of the vectors $a = (a_0, ..., a_{n-1})$ and $b = (b_0, ..., b_{n-1})$ is the coefficient at x^k in the polynomial $a(x)b(x)$. Consequently, the coefficient at x^{n-1} is the dot product of a and the reversed vector $b^R = (b_{n-1}, ..., b_0)$, i.e., $\sum_{i=0}^{n-1} a_i b_{n-1-i}$.*

By Lemma 1, we obtain a linear algorithm for the convolution of integer vectors, see Fig. 1.

```
Input: a natural number $M$ and two $n$-dimensional vectors $a = (a_0, ..., a_{n-1})$ and
    $b = (b_0, ..., b_{n-1})$ with integer coordinates in $[-M, M]$.
Output: the convolution $c = (c_0, ...., c_{2n-2})$ of $a$ and $b$.
1: $d \leftarrow 4nM^2 + 1$
2: $a(d) \leftarrow \sum_{l=0}^{n-1} a_l d^l$
3: $b(d) \leftarrow \sum_{l=0}^{n-1} b_l d^l$
4: $c(d) \leftarrow a(d)b(d)$
5: for $i = 0$ to $2n-2$ do
6:     $c_i \leftarrow \lfloor c(d)/d^i + \frac{1}{2} \rfloor - d\lfloor c(d)/d^{i+1} + \frac{1}{2} \rfloor$
7: end for
8: $c \leftarrow (c_0, ..., c_{2n-2})$
9: return $c$
```

Fig. 1. A linear algorithm for computing the convolution c of two n-dimensional integer vectors a and b.

Theorem 1. *Let n, M, d be natural numbers such that $d \geq 4nM^2 + 1$. For $k = 0, ..., 2n-2$, the k-th coordinate c_k of the convolution $c = (c_0, ..., c_{2n-2})$ of two integer vectors $a = (a_0, ..., a_{n-1})$ and $b = (b_0, ..., b_{n-1})$, each with n coordinates in $[-M, M]$, is equal to $\lfloor a(d)b(d)d^{-k} + \frac{1}{2} \rfloor - d\lfloor a(d)b(d)d^{-k-1} + \frac{1}{2} \rfloor$. Consequently, the convolution c of the n-dimensional vectors a and b can be computed using $O(n)$ additions, subtractions, multiplications, divisions and floor operations.*

Proof. By Lemma 1, we have $a(d)b(d) = \sum_{l=0}^{2n-2} c_l d^l$. Hence, $\lfloor a(d)b(d)d^{-k} + \frac{1}{2} \rfloor = \sum_{l=k}^{2n-2} c_l d^{l-k} + \lfloor \sum_{l=0}^{k-1} c_l d^{l-k} + \frac{1}{2} \rfloor$. On the other hand, for $l = 0, ..., 2n-2$, $|c_l| < 2nM^2$ and $d \geq 4nM^2 + 1$ hold. Let $p = d - 1$. For $k \geq 1$, we obtain

$$|\sum_{l=0}^{k-1} c_l d^l| \leq \frac{p}{2} \sum_{l=0}^{k-1} (p+1)^l = \frac{1}{2} p \frac{(p+1)^k - 1}{(p+1) - 1} = \frac{1}{2}(p+1)^k - \frac{1}{2} < \frac{1}{2}(p+1)^k \leq \frac{1}{2} d^k.$$

It follows that $\lfloor \sum_{l=0}^{k-1} c_l d^{l-k} + \frac{1}{2} \rfloor = 0$ and consequently $\lfloor a(d)b(d)d^{-k} + \frac{1}{2} \rfloor = \sum_{l=k}^{2n-2} c_l d^{l-k}$. Analogously, we have $\lfloor a(d)b(d)d^{-k-1} + \frac{1}{2} \rfloor = \sum_{l=k+1}^{2n-2} c_l d^{l-k-1}$.

This and the previous inequality yield the equality
$c_k = \lfloor a(d)b(d)d^{-k} + \frac{1}{2}\rfloor - d\lfloor a(d)b(d)d^{-k-1} + \frac{1}{2}\rfloor$. Hence, we obtain the algorithm for the convolution vector c depicted in Fig. 1. It uses a linear number of additions, subtractions, multiplications, divisions and floor operations. If M is not given as an input to the algorithm, we can upper bound M^2 by the sum of squares of the coordinates in the vectors a and b. □

Assuming the notation of Theorem 1, we obtain the following corollary.

Corollary 1. *The dot product of two integer vectors* $a = (a_0, ..., a_{n-1})$ *and* $b = (b_0, ..., b_{n-1})$, *each with* n *coordinates in* $[-M, M]$, *is equal to* $\lfloor a(d)b^R(d)d^{-n+1} + \frac{1}{2}\rfloor - d\lfloor a(d)b^R(d)d^{-n} + \frac{1}{2}\rfloor$, *where* $b^R = (b_{n-1}, ..., b_0)$.

Corollary 1 yields in turn a quadratic algorithm for the matrix product of two $n \times n$ integer matrices, see Fig. 2.

```
Input: a natural number $M$ and two $n \times n$ matrices $A = (a_{i,j})$ and $B = (b_{i,j})$ with
    integer entries in $[-M, M]$.
Output: the matrix product $C = (c_{i,j})$ of $A$ and $B$.
 1: $d \leftarrow 4nM^2 + 1$
 2: for $i = 1$ to $n$ do
 3:     $A_{i,*}(d) \leftarrow \sum_{l=1}^{n} a_{i,l} d^{l-1}$
 4: end for
 5: for $j = 1$ to $n$ do
 6:     $B_{*,j}(d) \leftarrow \sum_{l=1}^{n} b_{l,j} d^{n-l}$
 7: end for
 8: for $i = 1$ to $n$ do
 9:     for $j = 1$ to $n$ do
10:         $C_{i,j}(d) \leftarrow A_{i,*}(d)B_{*,j}(d)$
11:         $c_{i,j} \leftarrow \lfloor C_{i,j}(d)/d^{n-1} + \frac{1}{2}\rfloor - d\lfloor C_{i,j}(d)/d^{n} + \frac{1}{2}\rfloor$
12:     end for
13: end for
14: $C \leftarrow (c_{i,j})$
15: return $C$
```

Fig. 2. A quadratic algorithm for computing the matrix product C of two integer $n \times n$ matrices A and B.

Theorem 2. *The matrix product* C *of two* $n \times n$ *integer matrices* A *and* B *can be computed using* $O(n^2)$ *additions, subtractions, multiplications, divisions and floor operations.*

Proof. Our algorithm depicted in Fig. 2 is as follows. We set the constant d to $4nM^2 + 1$. If the range $[-M, M]$ of the entries in A or B is not known, we can upper bound M^2 by taking the sum of the squares of entries in the matrices A and B. It requires $O(n^2)$ additions and multiplications in total.

For $i = 1, ..., n$, for the i-th row $(a_{i,1}, ..., a_{i,n})$ of the matrix A, we consider the polynomial $A_{i,*}(x) = \sum_{k=1}^{n} a_{i,k}x^{k-1}$ and compute $A_{i,*}(d)$.

Asymmetrically, for $i = 1, ..., n$, for the j-th column $(b_{1,j}, ..., b_{n,j})$ of the matrix B, we consider the polynomial $B_{*,j}(x) = \sum_{k=1}^{n} b_{k,j}x^{n-k}$ and compute $B_{*,j}(d)$.

The computation of $A_{i,*}(d)$ and $B_{*,j}(d)$, for $1 \leq i,\ j \leq n$, requires $O(n^2)$ multiplications and additions.

Finally, for $1 \leq i,\ j \leq n$, we compute the products $A_{i,*}(d)B_{*,j}(d)$, and then $\lfloor A_{i,*}(d)B_{*,j}(d)/d^{n-1}\rfloor - d\lfloor A_{i,*}(d)B_{*,j}(d)/d^{n}\rfloor$. It requires $O(n^2)$ multiplications, floor operations and subtractions.

By Corollary 1, in this way, we obtain the correct values of the entries $c_{i,j}$ of the product matrix C. □

By combining the reduction of the distance matrix product to the arithmetic one outlined in the introduction [2, 12, 13, 15], we obtain also the following corollary.

Corollary 2. *The matrix product C of two $n \times n$ matrices A and B over the semi-ring $(\mathbb{Z}, \min, +)$ can be computed using $O(n^2)$ additions, subtractions, multiplications, divisions, and exponentiation, logarithm and floor operations.*

4 Final remarks

In our matrix multiplication algorithm (Algorithm 2 in Fig. 2), we need to compute solely the n-th coordinates of the convolutions of $(a_{i,1}, ..., a_{i,n})$ with $(b_{n,j}, ..., b_{1,j,})$. Hence, it is easy to observe that d can be decreased to $2nM^2 + 1$ in the algorithm.

We can decrease the bit length of the integers occurring in the computation of matrix product from $O(n(\log n + \log M))$ to $O(t(\log n + \log M))$ at the cost of increasing the time complexity by $O(n/t)$ factor as follows. To simplify the argument assume that n is divisible by t. We simply split each row vector $A_{i,*}$ and each column vector $B_{*,J}$ into n/t chunks with t coordinates. Next for each pair of corresponding column and row chunks, we compute their dot product by reversing the other chunk and computing the $(t-1)$-th coordinate of the convolution of the resulting pair. It remains to sum the n/t partial dot products.

The idea of coding a row or column of the input matrix with a polynomial is known in the literature [7]. Also, the Fast Fourier Transform is used to compute the outer products of rows and columns in [7].

Acknowledgments

We thank Christos Levcopoulos for valuable comments.

References

1. A.V. Aho, J.E. Hopcroft and J. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley Publishing Company, Reading, 1974.
2. N. Alon, Z. Galil and O. Margalit. *On the exponent of all pairs shortest path problem.* J. Comput. System Sci. , 54 (1997), pp. 25-51.
3. D. Coppersmith, S. Winograd. *Matrix Multiplication via Arithmetic Progressions.* J. of Symbolic Computation 9, 251–280 (1990)
4. R. Freivalds. *Probabilistic Machines Can Use Less Running Time.* IFIP Congress 1977, pp. 839–842.
5. I. Korec and J. Widermann. *Deterministic Verification of Integer Matrix Multiplication in Quadratic Time.* SOFSEM 2014: Theory and Practice of Computer Science, Lecture Notes in Computer Science Volume 8327, pp 375-382, 2014.
6. K. Mehlhorn and Z. Galil. *Monotone Switching Circuits and Boolean Matrix Product.* Computing 16, pp. 99-111, 1976.
7. R. Pagh *Compressed matrix multiplication.* TOCT 5(3): 9 (2013).
8. R. Pratt. *The Power of Negative Thinking in Multiplying Boolean Matrices.* SIAM J. Comput. 4(3), pp. 326-330, 1975.
9. R. Raz. On the complexity of matrix product. Proc. STOC 2002, pp. 144-151.
10. C.-P. Schnorr. *A Lower Bound on the Number of Additions in Monotone Computations.* Theor. Comput. Sci. 2(3), pp. 305-315, 1976.
11. V. Strassen. *Gaussian elimination is not optimal.* Numerische Mathematik 13, pp. 354-356, 1969.
12. F. Romani. *Shortest path problem is not harder than matrix multiplication.* Information Processing Letters vol. 11(3), pp. 134-136, 1980.
13. G. Yuval. *An algorithm for finding all shortest paths using $N^{2.81}$ infinite-precision multiplication.* Information Processing Letters vol. 11(3), pp. 155-156 , 1976.
14. V. Vassilevska Williams, *Multiplying matrices faster than coppersmith-winograd*, In: Proc. of STOC 2012, pp. 887–898.
15. U. Zwick. All pairs shortest paths using bridging rectangular matrix multiplication. Journal of the ACM, 49(3), pp. 289-317, 2002.