Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 58 (2003)

Defying Dimensions Modulo 6

Preliminary Version

Vince Grolmusz *
Department of Computer Science
Eötvös University, Budapest

Abstract

We consider here a certain modular representation of polynomials. The well-known modulo 6 representation of polynomial g is just polynomial $g + 6e$. The 1-a-strong representation of g modulo 6 is polynomial $g + 3f + 4h$, where no two of g, f and h has common monomials. In some cases the retrieval of g from its 1-a-strong representation is not hard: e.g., from $x_1 + 3x_2 + 4x_3$ one can get back x_1 simply by running through the values of x_i on the set $\{0, 1, 2, 3, 4, 5\}$, and noticing that only x_1 has period 6, ($3x_2$ has period 2, $4x_3$ has period 3).

Using this representation, we describe some surprising applications: we show how to encode $x_1, x_2, \ldots, x_n$ into $n^{o(1)}$ numbers[1], and how to retrieve the 1-a-strong representations of the x_i's from them using simple linear transformations. We show that $n \times n$ matrices can be converted to $n^{o(1)} \times n^{o(1)}$ matrices and from these tiny matrices we can retrieve similar representations of the original ones, also with linear transformations. We call this phenomenon a dimension-defying property of 1-a-strong representations. We also show that a 1-a-strong representation of the matrix-product can be computed with only $n^{o(1)}$ multiplications. We post the following open problem: by using this matrix product upto $O(n^2)$ times (even for different matrices) compute the (exact) matrix product of two $n \times n$ matrices. Solution for this open problem would yield a matrix-multiplication algorithm with only $O(n^{2+o(1)})$ multiplications.

We are building here to the results of paper (V. Grolmusz, Computing Elementary Symmetric Polynomials with a Sub-Polynomial Number of Multiplications, SIAM Journal on Computing, Vol. 32. No. 6.(2003), pp. 1475-1487).

1 Introduction

We consider here a certain modular representation of polynomials. The well-known modulo 6 representation of polynomial g is just any polynomial of the form $g + 6e$. Note, that a modulo 6 representation of polynomial g is also a modulo 3 and modulo 2 representation at the same time. This means, that if we examine the properties of the modulo 6 representations of polynomials (or, equivalently, polynomials over ring Z_6), it is not probable that we get more interesting properties over Z_6 than over fields F_2 or F_3.

Over composite, non-prime-power moduli (say 6), however, we can consider different representations as well. We will define 1-a-strong representations of polynomials formally in

*Address: Pázmány P. stny. 1/C, H-1117 Budapest, Hungary; E-mail: grolmusz@cs.elte.hu, http://www.cs.elte.hu/~grolmusz

ISSN 1433-8092

[1] Quantity $o(1)$ here denotes a positive number which goes to 0 as n goes to infinity

the next section, but now it is enough to say that the 1-a-strong representation of g modulo 6 is a polynomial $g + 3f + 4h$, where no two of g, f and h has common monomials. The last restriction is necessary, since otherwise constant 0 would be the 1-a-strong representation modulo 6 of an arbitrary polynomial g, simply because $0 = g + 3g - 4g$.

In some cases the retrieval of g from its 1-a-strong representation is not hard; let us call this procedure *filtering*:

Filtering

Suppose, that we know the value of the sum $A = x_1 + 3x_2 + 4x_3$ for some fixed $x_i = a_i \in \{0, 1\}$, $i = 1, 2, 3$, and we would like to learn the value of a_1. Now, one can get back the value of $x_1 = a_1$ simply by running through the values of $x_i = \alpha a_i$ for $i = 1, 2, 3$ one after the other, where $\alpha \in \{0, 1, 2, 3, 4, 5\}$, and observing A each time. Obviously, if we learn that A has period 6, then $a_1 = 1$, since only x_1 may have period 6. ($3x_2$ has period 2, $4x_3$ has period 3). If we do not observe the period of A to be 6, then $a_1 = 0$.

A similar polynomial representation modulo non-prime-power composites was considered in [Gro03a]. There we proved, that a representation (what we call a 0-a-strong representation in the next section) of the elementary symmetric polynomials can be computed dramatically faster than over prime moduli. This result plays a main rôle in the proofs of the present work.

1.1 Our Results

1.1.1 "Hyperdense coding"

Let $x_1, x_2, \ldots, x_n$ be a sequence of n variables. Then, using linear transforms modulo 6, we can compute $t = n^{o(1)}$ linear forms of the x_i's, denoted by $z_1, z_2, \ldots, z_t$, such that using another linear transform to these z_j's, we get back the 1-a-strong representations of the x_i's, namely

$$x_i + 3(x_{i_1} + x_{i_2} + \cdots + x_{i_u}) + 4(x_{j_1} + x_{j_2} + \cdots + x_{j_v}),$$

for $i = 1, 2, \ldots, n$, where the different indices mean different numbers.

Clearly, such phenomenon is impossible with representations of the form $x_i + 6\sum x_j$, since $x_i + 6\sum x_j$, seen modulo 3 is x_i, and this would contradict the dimension non-increasing property of linear transforms (over the 3-element field): clearly, linear forms $z_1, z_2, \ldots, z_t$ generates a space of dimension at most t, while linear forms $x_1, x_2, \ldots, x_n$ has dimension n.

Note, that if the values of $x_1, x_2, \ldots, x_n$ are bits, then z_i's are elements of Z_6, and the $(x_1, x_2, \ldots, x_n) \to (z_1, z_2, \ldots, z_t)$ correspondence is *not* an injection over Z_6.

From $x_i + 6\sum x_j$ it is easy to get back x_i: one should just reduce modulo 6. However, from our representation one can get back the actual x_i's by using filtering, described in the introduction. Naturally, this phenomenon does not violate obvious information-theoretical bounds, since in the course of filtering one should several times re-compute the z_i's.

1.1.2 Matrix-results

For $n \times n$ matrices X and Y with elements from set $\{0, 1, 2, 3, 4, 5\}$, our results include:

- The computation of $n^{o(1)} \times n^{o(1)}$ matrix X' with elements from set $\{0, 1, 2, 3, 4, 5\}$, such that from X', one can retrieve the 1-a-strong representation of the $n \times n$ matrix X; here both operations are simple linear transformations.

- The computation of the 1-a-strong representation of the product matrix XY, with only $n^{o(1)}$ multiplications, significantly improving our earlier result of computing the 1-a-strong representation of the matrix-product with $n^{2+o(1)}$ multiplications [Gro03b].

1.2 Earlier Results: Superdense Coding

It is one of the first tasks in any undergraduate information theory or computer science course to show that general n-bit sequences cannot be compressed to a shorter sequence or cannot be encoded by less than n bits. The proof of these statements are based on the fact that any injective image of a 2^n-element set must contain exactly 2^n elements, and 2^n elements cannot be encoded with binary sequences of length less than n.

However, using some fascinating physical phenomena and different models of computation, superdense coding is possible. Bennet and Wiesner [BW92], using Einstein-Podolski-Rosen entangled pairs, showed that n classic bits can be encoded by $\lceil n/2 \rceil$ quantum bits.

1.3 Earlier Results: Matrix Product

The matrix multiplication is a basic operation in mathematics in applications in almost every branch of mathematics itself, and also in the science and engineering in general. An important problem is finding algorithms for fast matrix multiplication. The natural algorithm for computing the product of two $n \times n$ matrices uses n^3 multiplications. The first, surprising algorithm for fast matrix multiplication was the recursive method of Strassen [Str69], with $O(n^{2.81})$ multiplications. After a long line of results, the best known algorithm today was given by Coppersmith and Winograd [CW90], requiring only $O(n^{2.376})$ multiplications. Some of these methods can be applied successfully in practice for the multiplication of large matrices [Bai88].

The best lower bounds for the number of needed multiplications are between $2.5n^2$ and $3n^2$, depending on the underlying fields (see [Blä99], [Bsh89], [Shp01]). A celebrated result of Raz [Raz02] is an $\Omega(n^2 \log n)$ lower bound for the number of multiplications, if only bounded scalar multipliers can be used in the algorithm.

In [Gro03b] we gave an algorithm with $n^{2+o(1)}$ multiplications for computing the 1-a-strong representation of the matrix product modulo non-prime power composite numbers (e.g., 6). The algorithm was an application of a method of computing the representation of the dot-product of two length-n vectors with only $n^{o(1)}$ multiplications.

In the present work, we significantly improve the results of [Gro03b], we give an algorithm for computing the 1-a-strong representation of the product of two $n \times n$ matrices with only $n^{o(1)}$ multiplications.

1.3.1 Why do we count only the multiplications?

In algebraic algorithms it is quite usual to count only the multiplications in a computation. The reason for this is that the multiplication is *considered* to be a harder operation than the addition in most practical applications, and moreover, the multiplication is *proven* to be harder in most theoretical models of computation.

For example, computing the PARITY is reduced to computing the multiplication of two n-bit sequences, and, consequently, two n-bit sequences cannot be multiplied on a polynomial-size, constant-depth Boolean circuit [FSS84], while it is well known, that two n-bit sequences can be added in such a circuit.

Another example is the communication complexity [KN97] of the computation of multivariable polynomials. Here two players, Alice and Bob, want to co-operatively compute the value of a $2n$-variable polynomial $f(x,y)$ modulo m, where the polynomial f is known for both players, while x is known only for Alice, and y is known only for Bob. If f is a linear polynomial, then they can compute $f(x,y)$ with communicating only $O(\log m)$ bits: Indeed, if $f(x,y) = \sum_i a_i x_i + \sum_j b_j y_j$, then Alice communicates to Bob the value $\sum_i a_i x_i \bmod m$, then Bob will know the value of $f(x,y)$. Similarly, if $f(x,y)$ can be given as the sum of u products, each with v clauses:

$$f(x,y) = \sum_{\mu=1}^{u} \prod_{\nu=1}^{v} \sum_{i,j} a_{i\mu\nu} x_i + b_{j\mu\nu} y_j,$$

then $f(x,y)$ can be computed with $O(uv \log m)$ bits of communication. This example also shows that in communication complexity the multiplications may be harder than the additions. Chor and Goldreich [CG85] proved that even approximating the dot-product function $f(x,y) = \sum_{i=1}^{n} x_i y_i$ needs $\Omega(n)$ bits of communication; consequently, multiplications are really hard in the communication model.

1.4 Earlier Results: Matrix Storage

There are lots of results concerning sparse matrix-storage and operations. Here we do not assume anything about the sparsity of our input matrices.

We should mention here the works of Frieze and Kannan [FK99] and Frieze, Kannan and Vempala [FKV98] on approximations of large matrices with small rank matrices, having short descriptions, with very fast algorithms.

Our results give a fast way to compress large matrices, but we cannot retrieve the same matrix from this much smaller matrix, just the 1-a-strong representation of the entries of the matrix. If we need to know the exact values of the entries, then we should perform some filtering procedures, similar to the example in the Introduction.

2 Preliminaries

2.1 A-strong representations

In [Gro03a] we gave the definition of the *a-strong (i.e., alternative-strong) representation* of polynomials. Here we define the *alternative*, and the *0-a-strong* and the *1-a-strong* representations of polynomials. Note that the 0-a-strong representation, defined here, coincides with the a-strong representation of the paper [Gro03a].

Note also, that for prime or prime-power moduli, polynomials and their representations (defined below), coincide. Perhaps that is the reason that such representations were not defined before.

Definition 1 *Let m be a composite number $m = p_1^{e_1} p_2^{e_2} \cdots p_\ell^{e_\ell}$. Let Z_m denote the ring of modulo m integers. Let f be a polynomial of n variables over Z_m:*

$$f(x_1, x_2, \ldots, x_n) = \sum_{I \in \{0,1,2,\ldots,d\}^n} a_I x_I,$$

where $a_I \in Z_m$, $x_I = \prod_{i=1}^{n} x_i^{\nu_i}$, *where* $I = \{\nu_1, \nu_2, \ldots, \nu_n\} \in \{0, 1, 2, \ldots, d\}^n$. *Then we say that*

$$g(x_1, x_2, \ldots, x_n) = \sum_{I \in \{0,1,2,\ldots,d\}^n} b_I x_I,$$

is an

- alternative representation *of* f *modulo* m, *if*

$$\forall I \in \{1, 2, \ldots, d\}^n \; \exists j \in \{1, 2, \ldots, \ell\} : \quad a_I \equiv b_I \pmod{p_j^{e_j}};$$

- 0-a-strong representation *of* f *modulo* m, *if it is an alternative representation, and, furthermore, if for some* i, $a_I \not\equiv b_I \pmod{p_i^{e_i}}$, *then* $b_I \equiv 0 \pmod{p_i^{e_i}}$;

- 1-a-strong representation *of* f *modulo* m, *if it is an alternative representation, and, furthermore, if for some* i, $a_I \not\equiv b_I \pmod{p_i^{e_i}}$, *then* $a_I \equiv 0 \pmod{m}$;

Example 2 *Let* $m = 6$, *and let* $f(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_1x_3$, *then*

$$g(x_1, x_2, , x_3) = 3x_1x_2 + 4x_2x_3 + x_1x_3$$

is a 0-a-strong representation of f *modulo 6;*

$$g(x_1, x_2, , x_3) = x_1x_2 + x_2x_3 + x_1x_3 + 3x_1^2 + 4x_2$$

is a 1-a-strong representation of f *modulo 6;*

$$g(x_1, x_2, , x_3) = 3x_1x_2 + 4x_2x_3 + x_1x_3 + 3x_1^2 + 4x_2$$

is an alternative representation modulo 6.

In other words, for modulus 6, in the alternative representation, each coefficient is correct either modulo 2 or modulo 3, but not necessarily both.

In the 0-a-strong representation, the 0 coefficients are always correct both modulo 2 and 3, the non-zeroes are allowed to be correct either modulo 2 or 3, and if they are not correct modulo one of them, say 2, then they should be 0 mod 2.

In the 1-a-strong representation, the non-zero coefficients of f are correct for both moduli in g, but the zero coefficients of f can be non-zero either modulo 2 or modulo 3 in g, but not both.

Example 3 *Let* $m = 6$. *Then* $0 = xy - 3xy + 2xy$ *is* **not** *a 1-a-strong representation of* xy. *Similarly, polynomial* $f + 2g + 3h$ *is a mod 6 1-a-strong representation of polynomial* f *if and only if* g *and* h *do not have common monomials with* f, *and* g *does not have common monomials with* h.

2.2 Previous results for a-strong representations

We considered elementary symmetric polynomials

$$S_n^k = \sum_{\substack{I \subset \{1,2,\ldots,n\} \\ |I|=k}} \prod_{i \in I} x_i$$

in [Gro03a], and proved that for constant k's, 0-a-strong representations of elementary symmetric polynomials S_n^k can be computed dramatically faster over non-prime-power composites than over primes: we gave a depth-3 multilinear arithmetic circuit with sub-polynomial number of multiplications (i.e., $n^{\varepsilon}, \forall \varepsilon > 0$), while over fields or prime moduli computing these polynomials on depth-3 multilinear circuits needs polynomial (i.e., $n^{\Omega(1)}$) multiplications.

Here depth-3 multi-linear, homogeneous arithmetic circuits computes polynomials of the form

$$\sum_{j=1}^{t} \prod_{k=1}^{\ell} \sum_{i=1}^{n} a_{jki} x_i \tag{1}$$

These circuits are sometimes called $\Sigma\Pi\Sigma$ circuits.

In [Gro03a], we proved the following theorem:

Theorem 4 ([Gro03a]) *Let $m = p_1 p_2$, where $p_1 \neq p_2$ are primes. Then a degree-2 0-a-strong representation of*

$$S_n^2(x,y) = \sum_{\substack{i,j \in \{1,2,\ldots,n\} \\ i \neq j}} x_i y_j,$$

modulo m:

$$\sum_{\substack{i,j \in \{1,2,\ldots,n\} \\ i \neq j}} a_{ij} x_i y_j \tag{2}$$

can be computed on a bilinear $\Sigma\Pi\Sigma$ circuit of size $\exp(O(\sqrt{\log n \log\log n})) = n^{o(1)}$. Moreover, this representation satisfies that $\forall i \neq j : a_{ij} = a_{ji}$.

□

Corollary 5 *The 0-a-strong representation (2) can be computed using $\exp(O(\sqrt{\log n \log\log n})) = n^{o(1)}$ multiplications.*

The following result is the basis of our theorems in the present paper.

Theorem 6 ([Gro03b]) *Let $m = p_1 p_2$, where $p_1 \neq p_2$ are primes. Then a degree-2, 1-a-strong representation of the dot-product $f(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n) = \sum_{i=1}^{n} x_i y_i$ can be computed with $t = \exp(O(\sqrt{\log n \log\log n}))$ multiplications of the form*

$$\sum_{j=1}^{t} \left(\sum_{i=1}^{n} b_{ij} x_i \right) \left(\sum_{i=1}^{n} c_{ij} y_i \right)$$

Proof: Let $g(x,y) = g(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n)$ be the degree-2 polynomial from Theorem 8 which is a 0-a-strong representation of $S_n^2(x,y)$. Then consider polynomial

$$h(x,y) = (x_1 + x_2 + \ldots + x_n)(y_1 + y_2 + \ldots + y_n) - g(x,y) \tag{5}$$

In $h(x,y)$, the coefficients of monomials $x_i y_i$ are all 1's modulo m, and the coefficients of monomials $x_i y_j$, for $i \neq j$ are 0 at least for one prime divisor of m, and if it is not 0 for some prime divisor, then it is 1. Consequently, by Definition 1, $h(x,y)$ is a 1-a-strong representation of the dot-product $f(x,y)$. □

The following definition is a natural generalization of the a-strong representations for matrices:

Definition 7 *Let $A = \{a_{ij}\}$ and $B = \{b_{ij}\}$ be two $n \times n$ matrices over Z_m. Then $C = \{c_{ij}\}$ is the alternative (1-a-strong, 0-a-strong) representation of the matrix A if for $1 \leq i, j \leq n$, the polynomial c_{ij} of n^2 variables $\{a_{ij}\}$ is an alternative (1-a-strong, 0-a-strong) representation of polynomial a_{ij}.*

Consequently, we say that matrix $D = \{d_{ij}\}$ is an alternative (1-a-strong, 0-a-strong) representation of the product-matrix AB, if for $1 \leq i, j \leq n$, d_{ij} is an alternative (1-a-strong, 0-a-strong) representation of polynomial $\sum_{k=1}^{n} a_{ik} b_{kj}$ modulo m, respectively.

In [Gro03b] we proved by n^2 applications of Theorem 6 that the 1-a-strong representations modulo 6 of the product of two $n \times n$ can be computed with $n^{2+o(1)}$ multiplications. Here we significantly improve this result: we show that the 1-a-strong representation can be computed by $n^{o(1)}$ multiplications. Moreover, the computation can be performed on a depth-3 homogeneous, bi-linear $\Sigma\Pi\Sigma$ circuit (bi-linear means that $\ell = 2$ in (1)).

Note, that we emphasize that the computation can be performed on such a circuit because this circuit is perhaps the simplest model of computation in which matrix-product can be computed; the main result is that a certain representation of the matrix product *can be computed by so few multiplications.*

3 Our Result for Hyperdense Coding

For simplicity, we prove our result here only for modulus 6; for other moduli, the poof is very similar, and is given in the full version of this work.

If we do not say otherwise from now on, the computations are modulo 6.

By Theorem 6, a 1-a-strong representation of the dot-product $\sum_{i=1}^{n} x_i y_i$ can be computed as

$$\sum_{i=1}^{n} x_i y_i + 3g(x,y) + 4h(x,y) = \sum_{j=1}^{t} \left(\sum_{i=1}^{n} b_{ij} x_i \right) \left(\sum_{i=1}^{n} c_{ij} y_i \right) \tag{6}$$

where $b_{ij}, c_{ij} \in \{0,1\}$ and where both g and h has the following form: $\sum_{i \neq j} a_{ij} x_i y_j$, $a_{ij} \bmod 6 \in \{0,1\}$, and no term $x_i y_j$ appears in both f and g; and $t = \exp(O(\sqrt{\log n \log\log n})) = n^{o(1)}$.

Now, let us observe that for each $j = 1, 2, \ldots, t$,

$$z_j = \sum_{i=1}^{n} b_{ij} x_i \tag{7}$$

is a linear combination of variables x_i.

Let these $t = n^{o(1)}$ linear forms be the encoding of the n 0-1 variable $x_i's$. The decoding is done also from (6): the 1-a-strong representation of x_i can be computed by plugging in

$$y^i = (0, 0, \ldots, \overbrace{1}^{i}, 0, \ldots, 0).$$

Obviously, on the LHS of (6) we get the 1-a-strong representation of x_i, and on the RHS we get a linear combination of the z_j's of (7).

4 Our Result for Matrix Compression

If we plug in vectors instead of just variables in these homogeneous linear forms, then we will get linear combinations of the vectors.

Now, let X denote an $n \times n$ matrix with entries as variables $\{x_{ij}\}$, and let x_i denote its $i^{\texttt{th}}$ column, for $i = 1, 2, \ldots, n$ and let $B = \{b_{ij}\}$, an $n \times t$ matrix of (6) and let $C = \{c_{ij}\}$ an $n \times t$ matrix of (6).

The columns of the $n \times t$ matrix-product XB gives the linear combinations specified by (7). However, XBC^T is an $n \times n$ matrix, such that its column ν is equal to $x_\nu + 3g_\nu(X) + 4h_\nu(X)$ where $g_\nu(X)$ and $h_n u(X)$ are linear combinations of the columns of X such that none of which contains x_ν and they do not contain the same column with non-zero coefficients. The proof of this fact is obvious from (6), observing that plugging in again

$$y^\nu = (0, 0, \ldots, \overbrace{1}^{\nu}, 0, \ldots, 0)$$

we simply generate some linear combinations of the columns of matrix XB, and the coefficient in these combinations are nothing else that the rows of C.

Consequently, we proved the following implication of Theorem 6:

Theorem 8 *There exist effectively computable constant $n \times t$ matrices B and C, such that for any $n \times n$ matrix $X = \{x_{ij}\}$, XBC^T is a 1-a-strong representation of matrix X modulo m, where t is equal to quantity (4), that is, $t = n^{o(1)}$.*

The dimension-defying implication of Theorem 8 is that X is an $n \times n$ matrix, XB is an $n \times n^{o(1)}$ matrix, and XBC^T is again an $n \times n$ matrix, all over the 6 element ring.

An easy corollary of Theorem 8, that

Corollary 9 *With the notations of Theorem 8, CB^TX is a 1-a-strong representation of matrix X modulo m, where t is equal to formula (4), that is, $t = n^{o(1)}$.*

Our main result in this section is the following

Theorem 10 *For any non-prime-power $m > 1$, there exist effectively computable constant $n \times t$ matrices B and C, such that for any matrix $X = \{x_{ij}\}$, B^TXB is a $t \times t$ matrix, where t is equal to quantity (4), that is, $t = n^{o(1)}$, and matrix CB^TXBC^T is a 1-a-strong representation of matrix X modulo m.*

The dimension-defying implication of Theorem 10 is that from the $n \times n$ matrix X with simple linear transformations we make the tiny $n^{o(1)} \times n^{o(1)}$ matrix B^TXB, and from this, again with simple linear transformations, $n \times n$ matrix CB^TXBC^T, where it is a 1-a-strong representation of matrix X modulo m.

Proof: From Theorem 8, XBC^T is a 1-a-strong representation of matrix X modulo m. Moreover, every column of XBC^T is a linear combination of the columns of X. By Corollary 9, for any $n \times n$ Y, CB^TY is a 1-a-strong representation of matrix Y modulo m, and, every row of CB^TY is a linear combination of the rows of Y. Plugging in $Y = XBC^T$, we get that the matrix CB^TXBC^T is a 1-a-strong representation of the matrix XBC^T, moreover, its rows are linear combinations of the rows of XBC^T, that is, CB^TXBC^T is also a 1-a-strong representation of the original X. □

5 Our result for matrix multiplication

5.1 Preliminaries

We need to define a sort of generalization of the matrix-product:

Definition 11 *$f : R^{2n} \to R$ is a homogeneous bilinear function over ring R if*

$$f(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n) = \sum_{1 \le i,j \le n} a_{ij} x_i y_j$$

for some $a_{i,j} \in R$. Let $U = \{u_{ij}\}$ be an $u \times n$ matrix over ring R, and let $V = \{v_{k\ell}\}$ be an $n \times v$ matrix over R. Then $U(f)V$ denotes the $u \times v$ matrix over R with entries $w_{i\ell}$, where

$$w_{i\ell} = f(u_{i1}, u_{i2}, \ldots, u_{in}, v_{1\ell}, v_{2\ell}, \ldots, v_{n\ell}).$$

Note, that if f is the dot-product, then $U(f)V$ is just the simple matrix-product.

First we need a simple lemma, stating that the associativity of the matrix multiplication is satisfied also for the "strange" matrix-multiplication defined in Definition 11:

Lemma 12 *Let*

$$f(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n) = \sum_{1 \le i,j \le n} a_{ij} x_i y_j$$

and let

$$g(x_1, x_2, \ldots, x_v, y_1, y_2, \ldots, y_v) = \sum_{1 \le i,j \le v} b_{ij} x_i y_j$$

be homogeneous bilinear functions over the ring R. Let $U = \{u_{ij}\}$ be an $u \times n$ matrix, and let $V = \{v_{k\ell}\}$ be an $n \times v$ matrix, and $W = \{w_{ij}\}$ be a $v \times w$ matrix over R, where u, n, w are positive integers. Then $(U(f)V)\,(g)W = U(f)\,(V(g)W)$, that is, the "strange" matrix-multiplication, given in Definition 11, is associative.

Proof 1: The proof is obvious from the homogeneous bi-linearity of f and g.
Proof 2: We also give a more detailed proof for the lemma. The entry of row i and column k of matrix $U(f)V$ can be written as

$$\sum_{z,t} a_{zt} u_{iz} v_{tk}.$$

Consequently, the entry in row i and column r of $(U(f)V)(g)W$ is

$$\sum_{k,\ell} b_{k\ell} \left(\sum_{z,t} a_{zt} u_{iz} v_{tk} \right) w_{\ell r}.$$

On the other hand, entry (t,r) in $V(g)W$ is

$$\sum_{k,\ell} b_{k\ell} v_{tk} w_{\ell r},$$

and entry (i,r) in $U(f)(V(g)W)$ is

$$\sum_{z,t} a_{zt} u_{iz} \sum_{k,\ell} b_{k\ell} v_{tk} w_{\ell r},$$

and this proves our statement.□

Now we are in the position of stating and proving our main theorem for matrix multiplications:

Theorem 13 *Let X and Y two $n \times n$ matrices, and let $m > 1$ be a non-prime-power integer. Then the 1-a-strong representation of the matrix-product XY can be computed with $t = n^{o(1)}$ multiplication, where t is given by (4).*

Proof: We use Theorem 8 and Corollary 9. Let us consider $t \times n$ matrix $B^T X$ and $t \times n$ matrix YB; these matrices can be computed without any multiplications from X and Y (we do not count multiplications by constants). Let $h(x,y)$ be the homogeneous bi-linear function (5). Then

$$B^T X(h) Y B$$

can be computed with $n^{o(1)}$ multiplications (Note, that because of Lemma 12, the associativity holds). Now compute matrix

$$CB^T X(f) Y B C^T = (CB^T Y)(f)(Y B C^T)$$

without any further (non-constant) multiplication. By Theorem 8 and Corollary 9, $CB^T X$ and YBC^T is a 1-a-strong representations of X and Y respectively, and they are the linear combinations of the rows of X and columns of Y, respectively. Consequently, using Theorem 6, $CB^T X(f) Y B C^T$ is a 1-a-strong representation of XY. □

6 Open Problem

It is a great challenge to prove or disprove the computability of the matrix product with only $n^{2+o(1)}$ multiplication. We post here the following problem:

By using our computation of the 1-a-strong representation of the matrix product upto $O(n^2)$ times (even for different matrices), compute the (exact, not the representation) matrix product of two $n \times n$ matrices.

Solution for this open problem would yield a matrix-multiplication algorithm with only $O(n^{2+o(1)})$ multiplications.

References

[Bai88] David H. Bailey. Extra high speed matrix multiplication on the Cray-2. *SIAM J. Sci. Statist. Comput.*, 9(3):603–607, 1988.

[Blä99] Markus Bläser. A $\frac{5}{2}n^2$-lower bound for the rank of $n \times n$-matrix multiplication over arbitrary fields. In *40th Annual Symposium on Foundations of Computer Science (New York, 1999)*, pages 45–50. IEEE Computer Soc., Los Alamitos, CA, 1999.

[Bsh89] Nader H. Bshouty. A lower bound for matrix multiplication. *SIAM J. Comput.*, 18(4):759–765, 1989.

[BW92] C.H. Bennet and S.J. Wiesner. Communication via one- and two particle operators on Einstein-Podolski-Rosen states. *Phys. Rev. Lett.*, 69:2881–2884, 1992.

[CG85] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. In *Proc. 26th Ann. IEEE Symp. Found. Comput. Sci.*, pages 429–442, 1985. Appeared also in SIAM J. Comput. Vol. 17, (1988).

[CW90] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Comput.*, 9(3):251–280, 1990.

[FK99] Alan Frieze and Ravi Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999.

[FKV98] Alan M. Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. In *IEEE Symposium on Foundations of Computer Science*, pages 370–378, 1998.

[FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits and the polynomial time hierarchy. *Math. Systems Theory*, 17:13–27, 1984.

[Gro03a] Vince Grolmusz. Computing elementary symmetric polynomials with a subpolynomial number of multiplications. *SIAM Journal on Computing*, 32(6):1475–1487, 2003.

[Gro03b] Vince Grolmusz. Near quadratic matrix multiplication modulo composites. Technical Report TR03-001, ECCC, 2003. ftp://ftp.eccc.uni-trier.de/pub/eccc/reports/2003/TR03-001/index.html.

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication Complexity.* Cambridge University Press, 1997.

[Raz02] Ran Raz. On the complexity of matrix product. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing.* ACM Press, 2002.

[Shp01] Amir Shpilka. Lower bounds for matrix product. In *IEEE Symposium on Foundations of Computer Science*, pages 358–367, 2001.

[Str69] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'