



# Defying Dimensions Modulo 6

Preliminary Version

Vince Grolmusz \*

## Abstract

We show that a certain representation of the matrix-product can be computed with  $n^{o(1)}$  multiplications. We also show, that similar representations of matrices can be compressed enormously with the help of simple linear transforms.

## 1 Introduction

Gigantic matrices emerge in applications in numerous fields of mathematics, computer science, statistics and engineering. The storage and the basic linear-algebraic operations of these matrices is a difficult task which should be handled in data-mining, signal-processing, image-processing and various other applications day by day.

Here we describe a method which, by the simplest tools, may help in these tasks. We describe how to store large matrices or do linear algebraic operations on them if it is sufficient to retrieve the matrices or the results of the linear-algebraic operations only in a certain form, called 1-a-strong representation modulo 6 in [Gro03].

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

- The computation of  $n^{o(1)} \times n^{o(1)}$  matrix  $A'$  with elements from set  $\{0, 1, 2, 3, 4, 5\}$ , such that from  $A'$ , one can retrieve the 1-a-strong representation of the  $n \times n$  matrix  $A$ ; here both operations are simple linear transformations (here  $o(1)$  denotes a sequence of positive numbers going to 0 while  $n$  goes to the infinity).
- The computation of the 1-a-strong representation of the matrix-product  $AB$  with  $n^{o(1)}$  multiplications, significantly improving our earlier result of computing the 1-a-strong representation of the matrix-product with  $n^{2+o(1)}$  multiplications [Gro03].

### 1.1 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

ISSN 1433-8092

---

\*Department of Computer Science, Eötvös University, Budapest, Pázmány P. stny. 1/C, H-1117 Budapest, Hungary; E-mail: grolmusz@cs.elte.hu

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 [Gro03] we gave an algorithm with  $n^{2+o(1)}$  multiplications for computing a *representation* of the matrix product modulo non-prime power composite numbers (e.g., 6). The algorithm was a straightforward 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 [Gro03], we gave an algorithm for computing a representation of the product of two  $n \times n$  matrices with only  $n^{o(1)}$  multiplications.

### 1.1.1 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 multi-variable 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.

## 1.2 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 with very fast algorithms.

## 2 Preliminaries

First we need to review several definitions and statements from [Gro03].

### 2.1 A-strong representations

In [Gro02] 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 [Gro02].

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  $Z_m$ :

$$f(x_1, x_2, \dots, x_n) = \sum_{I \subset \{1, 2, \dots, n\}} a_I x_I,$$

where  $a_I \in Z_m$ ,  $x_I = \prod_{i \in I} x_i$ . Then we say that

$$g(x_1, x_2, \dots, x_n) = \sum_{I \subset \{1, 2, \dots, n\}} b_I x_I,$$

is an

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

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

- 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

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

in [Gro02], 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

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

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

In [Gro02], we proved the following theorem:

**Theorem 4 ([Gro02])** (i) Let  $m = p_1 p_2$ , where  $p_1 \neq p_2$  are primes. Then a degree-2 0-a-strong representation of

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

modulo  $m$ :

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

can be computed on a bilinear  $\Sigma\Pi\Sigma$  circuit of size

$$\exp(O(\sqrt{\log n \log \log n})).$$

Moreover, this representation satisfies that  $\forall i \neq j : a_{ij} = a_{ji}$ .

(ii) Let the prime decomposition of  $m = p_1^{e_1} p_2^{e_2} \dots p_r^{e_r}$ . Then a degree-2 0-a-strong representation of  $S_n^2(x, y)$  modulo  $m$  of the form (1) can be computed on a bilinear  $\Sigma\Pi\Sigma$  circuit of size

$$\exp\left(O\left(\sqrt[r]{\log n (\log \log n)^{r-1}}\right)\right).$$

Moreover, this representation satisfies that  $\forall i \neq j : a_{ij} = a_{ji}$ .

□

**Corollary 5** The 0-a-strong representation of (2) can be computed using

$$\exp(O(\sqrt{\log n \log \log n}))$$

multiplications.

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

**Theorem 6 ([Gro03])** (i) 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

$$f(x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n) = \sum_{i=1}^n x_i y_i$$

can be computed with

$$\exp(O(\sqrt{\log n \log \log n})) \tag{3}$$

multiplications.

(ii) Let the prime decomposition of  $m = p_1^{e_1} p_2^{e_2} \dots p_r^{e_r}$ . Then a degree-2 1-a-strong representation of the dot-product  $f$  modulo  $m$  can be computed using

$$\exp\left(O\left(\sqrt[r]{\log n (\log \log n)^{r-1}}\right)\right) \tag{4}$$

multiplications.

(iii) Moreover, the representations of (i) and (ii) can be computed on bilinear  $\Sigma\Pi\Sigma$  circuits of size (3), and (4), respectively.

We reproduce here the short proof:

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

$$h(x, y) = (x_1 + x_2 + \dots + x_n)(y_1 + y_2 + \dots + y_n) - g(x, y) \quad (5)$$

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-power divisor of  $m$ . Consequently, by Definition 1,  $h(x, y)$  is a 1-a-strong representation of the dot-product  $f(x, y)$ .  $\square$

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 [Gro03] 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)).

Let us stress 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 matrix compression

For simplicity, we prove our result here only for modulus 6; for other moduli, the proof is very similar, using Theorem 6, part (ii) instead of part (i).

If we do not say otherwise, 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

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

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  $j = 1, 2, \dots, t$ ,

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

is a linear combination of variables  $x_i$ . 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  $i^{\text{th}}$  column, for  $i = 1, 2, \dots, n$  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  $\nu$  is equal to

$$x_\nu + 3g_\nu(X) + 4h_\nu(X)$$

where  $g_\nu(X)$  and  $h_\nu(X)$  are linear combinations of the columns of  $X$  such that none of which contains  $x_\nu$  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

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

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** *For any non-prime-power  $m > 1$  and for the  $n \times n$  matrix  $X = \{x_{ij}\}$ , there exist effectively computable constant  $n \times t$  matrices  $B$  and  $C$ , such that*

$$XBC^T$$

*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.

An easy corollary of Theorem 8, that

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

Our main result in this section is the following

**Theorem 10** *For any non-prime-power  $m > 1$  and for the  $n \times n$  matrix  $X = \{x_{ij}\}$ , there exist effectively computable constant  $n \times t$  matrices  $B$  and  $C$ , such that*

$$B^T X B$$

*is a  $t \times t$  matrix, where  $t$  is equal to quantity (4), that is,  $t = n^{o(1)}$ , and matrix*

$$C B^T X B C^T$$

*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  $B^T X B$ , and from this, again with simple linear transformations,  $n \times n$  matrix  $C B^T X B C^T$ , where it is a 1-a-strong representation of matrix  $X$  modulo  $m$ .

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

## 4 Our result for matrix multiplication

### 4.1 Preliminaries

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

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

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

*for some  $a_{i,j} \in R$ . Let  $U = \{u_{ij}\}$  be an  $u \times n$  matrix over ring  $R$ , and let  $V = \{v_{kl}\}$  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*

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

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*

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

and let

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

be homogeneous bilinear functions over the ring  $R$ . Let  $U = \{u_{ij}\}$  be an  $u \times n$  matrix, and let  $V = \{v_{kl}\}$  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.  $\square$

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  $t = n^{o(1)}$  multiplications, 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

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

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

$$CB^T X(f)YBC^T = (CB^T Y)(f)(YBC^T)$$

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,  $CB^T X(f)YBC^T$  is a 1-a-strong representation of  $XY$ .  $\square\square$

#### Acknowledgment.

The author acknowledges the partial support of the Széchenyi István fellowship.

## 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.
- [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.
- [Gro02] Vince Grolmusz. Computing elementary symmetric polynomials with a sub-polynomial number of multiplications. Technical Report TR02-052, ECCC, 2002. To appear in the *SIAM Journal on Computing*.
- [Gro03] Vince Grolmusz. Near quadratic matrix multiplication modulo composites. Technical Report TR03-001, ECCC, 2003. <ftp://ftp.eccc.univ-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.