Integer and Polynomial Multiplication: Towards Optimal Toom-Cook Matrices

Marco Bodrato, Alberto Zanoni

Centro Interdipartimentale "Vito Volterra"
Università degli Studi di Roma "Tor Vergata"
Via Columbia 2, 00133 Roma (Italy)

{bodrato, zanoni}@volterra.uniroma2.it

ABSTRACT

Karatsuba and Toom-Cook are well-known methods used to multiply efficiently long integers. There have been different proposal about the interpolating values used to determine the matrix to be inverted and the sequence of operations to invert it. A definitive word about which is the optimal matrix (values) and the (number of) basic operations to invert it seems still not to have been said. In this paper we present some particular examples of useful matrices and a method to generate automatically, by means of optimised exhaustive searches on a graph, the best sequence of basic operations to invert them.

Categories and Subject Descriptors

F.2.1 [**Analysis of algorithms and program complexity**]: Numerical algorithms and problems—*Computations on polynomials*; G.1.1 [**Numerical analysis**]: Interpolation—*Interpolation formulas*; G.2.3 [**Discrete mathematics**]: Applications; I.1.2 [**Computing methodologies**]: Algorithms—*Algebraic algorithms*

General Terms

Algorithms, Performance, Theory

Keywords

Integer and polynomial multiplication, squaring, Karatsuba, Toom-Cook, interpolation, matrix inversion

1. INTRODUCTION

Starting with the works of Karatsuba [9], Toom [12] and Cook [5], who found methods to lower asymptotic complexity for polynomial multiplication from $O(n^2)$ to $O(n^{1+\epsilon})$ with $0 < \epsilon < 1$, many efforts have been done in finding optimised implementations in arithmetic software [6, 8, 10].

The family of so-called Toom-Cook methods is an infinite set of algorithms (called Toom-3, Toom-4, etc. - we identify Karatsuba with Toom-2). Each of them may be viewed as a polynomial interpolation problem, for which the base points are not specified a priori, from which a matrix to be inverted rise. We indicate the matrix related to Toom-n method with $A_n \in GL(\mathbb{Z}, 2n-1)$.

ISSAC'07, July 29–August 1,2007, Waterloo, Ontario, Canada.

2. TOOM-COOK METHODS

For brevity, we call Toom-n the Toom-Cook method splitting operands in n parts. Standard analysis tells that, for a fixed n, the complexity of Toom-n is $O(m^{log_n(2n-1)})$.

In practice, only very small values of n (as 2, 3, 4) are used, because of the asymptotically better $O(n \log n \log\log n)$ complexity Schönhage-Strassen method [11], which has many implementation issues [7]. The thresholds indicating the convenience of one method in comparison with another one depend very much on the implementation, mainly depending on the choice of A_n and of the exact sequence of operations.

2.1 The classical point of view

We briefly recall the Toom-n multiplication algorithm as generalised in [1]. Let $\mathbf{R}$ be $\mathbb{Z}$ or $\mathbb{Z}[X]$, and $u, v \in \mathbf{R}$. To compute the product $u \cdot v = w \in \mathbf{R}$, the five steps below are needed.

Splitting : Fix an appropriate basis $B \in \mathbf{R}$ and represent the two operands by two polynomials in $a, b \in \mathbf{R}[x]$ of degree $d = n - 1$:

$$a(x) = \sum_{i=0}^{d} a_i x^i \qquad ; \qquad b(x) = \sum_{i=0}^{d} b_i x^i$$

such that $u = a(x)|_{x=B}$ and $v = b(x)|_{x=B}$. One assumes that the factors have equal degrees, pad the lower-degree one with zero coefficients otherwise.

Evaluation : Choose $2n-1$ values $v_i \in \mathbf{R}$, evaluate both operands on all of them, obtaining $a(v_i)$, $b(v_i)$.

Recursion : Compute $w_i = a(v_i) \cdot b(v_i)$ recursively. Let $\mathbf{w} = (w_i)$ be the so obtained values vector.

Interpolation : Solve the interpolation problem $c(v_i) = w_i$ inverting the obtained Vandermonde matrix A_n generated by the v_i values and computing $\mathbf{c} = A_n^{-1}\mathbf{w}$, where $\mathbf{c} = (c_i)$ is the vector of $c(x)$ coefficients.

Recomposition : Once all the coefficient are computed, it's enough to evaluate back $w = c(x)|_{x=B}$.

Note that this way we can only obtain matrices A_n having odd dimension $2n-1 = 2d+1$.

2.2 The unbalanced point of view

In this paper, we will also analyse "intermediate" versions of Toom-Cook methods, which consider $a(x), b(x)$ with different degrees d_1, d_2. We indicate with $n_1 = d_1 + 1, n_2 = d_2 + 1$ the number of necessary subdivisions. The principle remains basically the same, but now the result has degree $\widetilde{d} = d_1 + d_2$ and we need $\widetilde{d} + 1 = n_1 + n_2 - 1$ evaluation points v_i. The obtained matrices – indicated with A_{n_1,n_2} – have dimension $\widetilde{d}+1$, that can then also be even.

They depend only on the sum n_1+n_2 (for the dimension) and on the chosen points v_i (for their entries), not on n_1, n_2 separately.

Examples : (w.l.o.g. we suppose $n_1 > n_2$)

$\boxed{n_2 = 1}$ Here $b(x) = b_0$: it is more convenient to compute directly the product coefficients $c_i = a_i \cdot b_0$.

$\boxed{(n_1, n_2) = (3, 2)}$ Here $c(x)$ has degree 3, and $A_{3,2}$ has order 4. In section 8 we call this method Toom-2.5

$\boxed{(n_1, n_2) = (4, 2)}$ In this case $A_{4,2}$ has order 5, and A_3 may well be chosen as optimal choice $A_{4,2}$

$\boxed{\left.\begin{matrix} n_1 + n_2 \\ d_1 + d_2 \end{matrix}\right\} \equiv 0 \ (2)}$ Generalisation: a "fall back" to the classical Toom-$((d_1+d_2)/2+1)$ method: $A_{\frac{n_1+n_2}{2}}$ may be used as A_{n_1,n_2}

The (4,2) case suits very well in practice. A program/library implementing both Toom-3 and Toom-4 *should* use Toom-3 when one factor has 4 parts and the other one just 2. More generally, it is interesting to compare different methods when applied to integers a, b whose lengths are in certain ratio. In the following table we indicate some of the possibilities. The last column contains the number of needed multiplications of two (sub)-factors, whose length is the indicated fraction of the longest among a, b.

Ratio	Method	Multiplications
2 : 1	2 × Karatsuba	6 of 1/4
4 : 2	Toom-3	5 of 1/4
2 : 1	2 × Toom-3	10 of 1/6
6 : 3	Toom-4.5	8 of 1/6
3 : 1	3 × Karatsuba	9 of 1/6
6 : 2	Toom-4	7 of 1/6

Different methods can be applied for different possible splittings of the factors, with different complexities. Thresholds should then be considered in choosing the most appropriate one.

3. THE GOAL

In this paper we study the interpolation phase, consisting in multiplying the inverse of a matrix and a vector. We look for the matrix with the most efficient inversion procedure, as explained below.

In software implementations, a set Op of very efficiently implemented basic operations (typically sums, subtractions, bit shiftings, multiplication and division by small numbers, etc.) is given, and the idea is to use them to invert A_n by means of a sequence of elementary row operations. A particular implementation of Toom-n method must then specify

1. The interpolation points v_i, determining A_n.
2. The sequence of operations in Op to invert the matrix (we call it *inversion sequence*, or IS, for short; when we want to emphasise the dimension, we will use IS_n).

In the scientific literature there is still not a definitive word on which is the best matrix to be used and which is the corresponding sequence of basic operations to invert it. For Karatsuba method, a exhaustive search on a "reasonable" set of interpolation points is very easy, while already for Toom-3 things are less clear. For example, only recently the GMP library (from version 4.2) changed implementation for Toom-3, choosing a different IS_3, more efficient than the one in the precedent release.

It is not trivial at all to prove the optimality of a matrix A_n with respect to the related Toom-n method, for there is an infinite number of possibilities for v_i values, and for the IS's. Some heuristics for v_i choice are discussed in Zuras' paper [13], but, to the best of our knowledge, the final word has still not been said.

We introduce some optimality criteria to measure goodness of inversion sequences. We present an algorithm to automatically search for an optimal IS starting from a given matrix A_n. We consider a model based on:

- Minimal use of extra memory (no temporary variables).
- No matrix support cardinality increase.
- Only linear combinations between couples of lines, or
- exact divisions of a single line by small integers.

4. THE MATRICES

We note the identity matrix of order r with I_r; if the order is understood or not relevant, simply with I.

Even if Toom-n works for whatever choice of the v_i, it is better to choose them in order to minimise the matrix inversion overhead as much as possible. The inverse is usually computed by a sequence of elementary row operations à la Gauss, and therefore we should search the "shortest" (least number of elementary operations) and easiest (fastest elementary operations) way to compute A_n^{-1}.

It is possible to consider rational v_i values even when working only with integers. Infact, if $v_i = N_i/D_i$, we have

$$a(v_i)\cdot b(v_i) = c(v_i) \Rightarrow \left[\sum_{k=0}^{d_1} a_k \left(\frac{N_i}{D_i}\right)^k\right] \cdot \left[\sum_{k=0}^{d_2} b_k \left(\frac{N_i}{D_i}\right)^k\right] = \sum_{k=0}^{\widetilde{d}} c_k \left(\frac{N_i}{D_i}\right)^k$$

and multiplying by $D_i^{\widetilde{d}}$ we may get rid of the denominators:

$$\left(\sum_{k=0}^{d_1} a_k N_i^k D_i^{d_1-k}\right) \cdot \left(\sum_{k=0}^{d_2} b_k N_i^k D_i^{d_2-k}\right) = \sum_{k=0}^{\widetilde{d}} c_k N_i^k D_i^{\widetilde{d}-k}$$

so that the i^{th} line of A_n is $(N_i^{\widetilde{d}}, N_i^{\widetilde{d}-1}D_i, \ldots, N_i D_i^{\widetilde{d}-1}, D_i^{\widetilde{d}})$. In particular,

- integers $v_i = N_i$ give $(N_i^{\widetilde{d}}, N_i^{\widetilde{d}-1}, \ldots, N_i, 1)$
- integer reciprocals $v_i = 1/D_i$ give $(1, D_i, \ldots, D_i^{\widetilde{d}-1}, D_i^{\widetilde{d}})$

We use ∞ as interpolation "value" to indicate $a_d \cdot b_d$ product computation, which in a certain sense represents interpolation on the reciprocal of zero, or, more precisely,

$$a(\infty) = \lim_{x\to\infty} \frac{a(x)}{x^{d_1}} = a_{d_1} \qquad ; \qquad b(\infty) = \lim_{x\to\infty} \frac{b(x)}{x^{d_2}} = b_{d_2}$$

Following the literature, we will always consider the interpolating values $v_0 = 0$ and $v_{\widetilde{d}} = \infty$, so that A_n will have the following shape:

$$A_n = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ \mathrm{X} & \mathrm{X} & \cdots & \mathrm{X} & \mathrm{X} \\ \vdots & & \vdots & & \vdots \\ \mathrm{X} & \mathrm{X} & \cdots & \mathrm{X} & \mathrm{X} \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \quad \mathrm{X} \equiv \text{non zero entry} \qquad (1)$$

and $c_0 = w_0$, $c_{\widetilde{d}} = w_{\widetilde{d}}$. Our inversion analysis will focus mainly on inner lines. Vandermonde determinant's formula applied to A_n matrices gives:

THEOREM 1. *For a Vandermonde-like Toom matrix A_n generated by the $r = 2n-1$ values $\left\{\infty, \frac{N_2}{D_2}, \ldots, \frac{N_{r-1}}{D_{r-1}}, 0\right\}$*

$$\det(A_n) = \left(\prod_{i=2}^{r-1} N_i D_i\right) \cdot \prod_{1<i<j<r} (N_i D_j - N_j D_i)$$

PROOF. Starting from Vandermonde formula, and remembering that in order to have integer entries denominators are cleared in every line of A_n multiplying it by D_i^{r-1}, we have

$$\begin{aligned} \det(A_n) &= \left(\prod_{i=1}^{r} D_i^{r-1}\right) \cdot \prod_{i<j} \left(\frac{N_i}{D_i} - \frac{N_j}{D_j}\right) \\ &= \left(\prod_{i=1}^{r} D_i^{r-1}\right) \cdot \prod_{i<j} \left(\frac{N_i D_j - N_j D_i}{D_i D_j}\right) \\ &= \left(\prod_{i=1}^{r} D_i^{r-1}\right) \cdot \frac{\prod_{i<j}(N_i D_j - N_j D_i)}{\prod_{i<j} D_i D_j} \end{aligned}$$

In the denominator $\prod_{i<j} D_i D_j$ every term D_k appears $r-k$ times in the first place ($k = i, j = k+1, \ldots, r$) and $k-1$ times in the second ($j = k, i = 1, \ldots, k-1$), so that the result is $\prod_{i=1}^{r} D_i^{r-1}$. Canceling out, we obtain

$$\left(\prod_{i=1}^{r} D_i^{r-1}\right) \cdot \frac{\prod_{i<j}(N_i D_j - N_j D_i)}{\prod_{i=1}^{r} D_i^{r-1}} = \prod_{1\leqslant i<j\leqslant r} (N_i D_j - N_j D_i)$$

The value 0 corresponds to $N_r = 0, D_r = 1$, while ∞ to $N_1 = 1, D_1 = 0$ (it is easy to prove that the above formula is valid in this latter case, too). Considering these two cases ($i = 1, j = r$) apart, we have

$$\left(D_r \prod_{j=2}^{r-1} D_j\right)\left(\prod_{1<i<j<r} (N_i D_j - N_j D_i)\right)\left(N_1 \prod_{i=2}^{r-1} N_i\right)$$

which gives the desired result. □

LEMMA 1. *Let $\alpha = \pm 2^a \neq \beta = \pm 2^b$, with $a, b \in \mathbb{Z}$. Then $\gamma = \alpha - \beta$ is a power of 2 only if $\alpha = -\beta$ or $\alpha = 2\beta$ or $\alpha = \beta/2$.*

PROOF. There are two cases:

(1) $\alpha\beta > 0$: we have $\pm\gamma = 2^a - 2^b = 2^b(2^{a-b} - 1)$, and $2^{a-b} - 1$ is a power of 2 iff $|a-b| \leqslant 1$. Excluding the case $a = b$, corresponding to $\alpha = \beta$, only the cases $\alpha = 2\beta$, $\alpha = \beta/2$ remain.

(2) $\alpha\beta < 0$: we have $\pm\gamma = 2^a + 2^b = 2^b(2^{a-b} + 1)$, and $2^{a-b} + 1$ is a power of 2 iff $a = b$, that is $\alpha = -\beta$ □

PROPOSITION 1. *For each Toom matrix A_n with $n \geqslant 3$, its determinant is not a power of 2.*

PROOF. By contradiction, we should have that $\det(A_n) = \left(\prod_{i=2}^{r-1} N_i D_i\right) \prod_{1<i<j<r} (N_i D_j - N_j D_i)$ should be a power of 2, according to proposition 1. Looking at the first factor, this means that all the N_i, D_i should be powers of 2, say $N_i = \pm 2^{e_i}, D_i = 2^{f_i}$ and the same for all the factors $(N_i D_j - N_j D_i)$. Supposing $(N_i, D_i) = 1$ ($e_i \cdot f_i = 0$), we have

$$\begin{aligned} (N_i D_j - N_j D_i) &= D_i D_j \frac{(N_i D_j - N_j D_i)}{D_i D_j} \\ &= D_i D_j \left(\frac{N_i}{D_i} - \frac{N_j}{D_j}\right) = \pm 2^f (x_i - x_j) \end{aligned}$$

Fixing $x_1 = 2^a$, any power of 2, by lemma 1 we should choose the other x_i in the set $\{-2^a, 2^{a+1}, 2^{a-1}\}$, but for any two of them we never have $x_i - x_j$ being a power of 2. □

For $n \geqslant 3$, a division (not just a simple shifting) is therefore necessary in whatever IS of a Toom matrix.

THEOREM 2. *Let A_n be the Toom matrix generated by $\{\infty, 1, -1, v_4, \ldots, v_{r-1}, 0\}$. Then the number of necessary divisions Δ_n in a minimal IS is less than or equal to $r-4 = 2n-5$.*

PROOF. An IS can be the following one:

$$A_n = \begin{pmatrix} 1 & 0 \cdots 0 & 0 \\ 1 & 1 \cdots 1 & 1 \\ \pm 1 & \mp 1 \cdots 1 & -1 \\ \cdots & \cdots & \cdots \\ 0 & 0 \quad 0 & 1 \end{pmatrix} \vdots \Rightarrow \begin{pmatrix} 1 & 0 \cdots 0 & 0 \\ 0 & 1 \cdots 1 & 1 \\ 0 & \mp 1 \cdots 1 & -1 \\ \cdots & \cdots & \cdots \\ 0 & 0 \quad 0 & 1 \end{pmatrix} \vdots \Rightarrow \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & c_3 & \cdots & c_r \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \vdots \Rightarrow I_r$$

where $c_3 = \pm 2$ and c_j can be only $0, 2$ or -2 for $j > 3$. This means that (a shift for the third line and) no more than $r-4$ divisions are needed to invert the (remaining part of the) matrix. □

CONJECTURE 1. *We conjecture that $\Delta_n = r - 4 = 2n - 5$.*

5. OPTIMALITY CRITERIA

In the following, for a square matrix M we indicate with $M[i,j]$ its entry in position (i,j), with $M^{(i)}$ its i^{th} line and with $M^{[j]}$ its j^{th} column.

DEFINITION 1. *The* **support** *of $M^{(i)}$ is the set $s(M^{(i)})$ of column indexes $j \in \mathbb{N}$ such that $M[i,j] \neq 0$, and similarly (dually) for $M^{[i]}$. The* **support** *of M is the set $s(M)$ of pairs $(i,j) \in \mathbb{N} \times \mathbb{N}$ such that $M[i,j] \neq 0$. The cardinality of a line support $s(M^{(i)})$ is indicated with $^{\#}M^{(i)}$, of a column support with $^{\#}M^{[i]}$ and of $s(M)$ with $^{\#}M$.*

DEFINITION 2. *We note with $gcd_i(M) = gcd(M^{(i)})$ the greatest common divisor of all coefficients in $M^{(i)}$.*

Inversion sequences are sequences of basic operations modifying the initial matrix and producing as last matrix the identity matrix. Gauss' method is e.g. a classical algorithm to produce effective IS, reducing at each step the *maximum* possible cardinality of still-to-be-analysed lines support. However, it uses as basic operations, apart from sums and subtractions, multiplications and division *at each step*, and from a computer point of view this is not quite optimal when looking for efficiency.

Gauss' method is quite efficient to invert matrices, but we deal with a single matrix, fixed at the beginning, and search

IS which are, in terms of quantity and/or quality of used basic operations, the best possible.

We propose here some optimality criteria, that guarantee termination of our exhaustive algorithm. For A_n matrices as specified in (1), we have

$$\begin{aligned} {}^{\#}A_n &= 1 + (2n-1-2)(2n-1) + 1 \\ &= ((2n-1)^2 - 2(2n-1) + 1) + 1 \\ &= (2n-1-1)^2 + 1 = 4(n-1)^2 + 1 \end{aligned}$$

We now list the used criteria: we suppose that the basic row operations involve lines $M^{(i_1)}, M^{(i_2)}$, and that $M^{(i_1)}$ is overwritten with the result. We indicate with $\widetilde{M}$ the matrix *after* the execution of the operation.

(A) Support reduction: ${}^{\#}\widetilde{M}^{(i)} < {}^{\#}M^{(i)}$. The resulting line must have at least one more entry with value 0. "Old" 0 entries are not modified.

(B) Regularization: $\widetilde{M}[i_1, j_1]/M[i_2, j_1] = \widetilde{M}[i_1, j_2]/M[i_2, j_2]$. The resulting line must have more entries differing from the corresponding ones in another line by a common multiplicative factor than before.

Criterion **(A)** is sufficient to guarantee termination of the exhaustive research algorithm that we'll describe in the following section. The solution implemented by GMP-4.2.1 applies once criterion **(B)**.

6. THE TOOM GRAPH

Let a set of interpolation values $\{v_i\}$ be given, that is, let A_n be known from the beginning. We perform an exhaustive search on all the possible IS, according to the criteria of section 5. This can be modeled by an oriented and weighted graph $G = (N, E)$ (which we call Toom graph), in which each node $\nu \in N$ is represented by the matrix M_ν. We call α the distinguished "initial" node, with $M_\alpha = A_n$, and similarly ω the corresponding "final" node for I. Edges are described below.

In order to have a sufficiently easy theoretical modeling of the graph procedure analysis, we will consider as basic operations the following ones (i, i_1, i_2 are row indexes and $c, c_1, c_2 \in \mathbb{Z}$):

1. $M^{(i_1)}_{\nu'} = \widetilde{M}^{(i_1)}_{\nu} = c_1 M^{(i_1)}_{\nu} + c_2 M^{(i_2)}_{\nu}$
2. $M^{(i)}_{\nu'} = \widetilde{M}^{(i)}_{\nu} = \dfrac{M^{(i_1)}_{\nu}}{c} = \left(\dfrac{M_\nu[i,1]}{c}, \cdots, \dfrac{M_\nu[i,2n-1]}{c} \right)$

where each of the divisions by c is exact. The edges $\varepsilon = (\nu_1, \nu_2) \in E$ are identified by the quadruple (i_1, i_2, c_1, c_2) or by the pair (i, c), respectively. With this in mind, IS can simply be considered as edges sequences, or, in graph terminology, as paths joining α to ω.

To consider the different computational cost of row elementary operations depending on c, c_1, c_2 values in graph analysis, we introduce some constant weights, named in table 1 (possibly swapping $|c_1|$ and $|c_2|$ values). `STEP` is practically the basic cost of an addition/subtraction of two long integers. An appropriated tuning of these constants will result in choosing a particular criterion of graph visit.

Definition 3. *The* **weight** *$w(\varepsilon)$ of an edge $\varepsilon \in E$ ($w(c)$ of a coefficient $c \in \mathbb{Z}$) is the constant defined by table 1*

$\lvert c_1 \rvert$	$\lvert c_2 \rvert$	**Operation**	**Weight**
1	1	$+$ or $-$	`STEP`
1	$= 2^k$	$\pm$, shift	`''` + `_1_2`
1	$\neq 2^k$	$\pm$, multiply	`''` + `_1_X`
$= 2^k$	$\neq 2^h$	$\pm$, shift and multiply	`''` + `_2_X`
$\neq 2^k$	$\neq 2^h$	general combination	`''` + `_X_Y`

$\lvert c \rvert$	**Operation**	**Weight**
1	(may be unitary $-$)	~ 0
$= 2^k$	shifting	`SHIFT`
$\neq 2^k$	/	`DIV`

Table 1: Constants for elementary row operations

according to the values of c_1, c_2 or c. The weight w_{IS} of a IS $= (\varepsilon_1, \ldots, \varepsilon_l)$, where $\varepsilon_i = (\nu_i, \nu_{i+1})$ and $M_{\nu_{l+1}} = I$, is $w_{IS} = w(\varepsilon_1) + \cdots + w(\varepsilon_l)$. An IS is **minimal** *if its weight is minimal among the weights of all possible IS connecting ν_1 to ν_{l+1}. The weight $w(M)$ of a matrix M is w_{IS} of a minimal IS for M. The weight $w(\nu)$ of a node ν is $w(\nu) = w(M_\nu)$.*

Obviously, $w(I) = 0$.

Example (Karatsuba graph): Let $(v_0 = \infty, v_1 = 1, v_2 = 0)$ identify Karatsuba matrix A_2. We have

$$\begin{array}{ccc} A_2 & \xrightarrow{\varepsilon_1} & M_1 \\ {\scriptstyle\varepsilon_2}\downarrow & & \downarrow{\scriptstyle\varepsilon_3} \\ M_2 & \xrightarrow{\varepsilon_4} & I \end{array} \quad \text{with} \quad \begin{array}{l} A_2 = \begin{pmatrix} 1&0&0 \\ 1&1&1 \\ 0&0&1 \end{pmatrix} \\ M_1 = \begin{pmatrix} 1&0&0 \\ 0&1&1 \\ 0&0&1 \end{pmatrix},\ M_2 = \begin{pmatrix} 1&0&0 \\ 1&1&0 \\ 0&0&1 \end{pmatrix} \end{array}$$

Example (Knuth graph): Let $(v_0 = \infty, v_1 = -1, v_2 = 0)$ identify the Knuth matrix A'_2. We have

$$\begin{array}{ccc} A'_2 & \xrightarrow{\varepsilon_1} & M'_1 \\ {\scriptstyle\varepsilon_2}\downarrow & & \downarrow{\scriptstyle\varepsilon_3} \\ M'_2 & \xrightarrow{\varepsilon_4} & I \end{array} \quad \text{with} \quad \begin{array}{l} A'_2 = \begin{pmatrix} 1&0&0 \\ 1&-1&1 \\ 0&0&1 \end{pmatrix} \\ M'_1 = \begin{pmatrix} 1&0&0 \\ 0&1&-1 \\ 0&0&1 \end{pmatrix},\ M'_2 = \begin{pmatrix} 1&0&0 \\ 1&-1&0 \\ 0&0&1 \end{pmatrix} \end{array}$$

These almost trivial graphs show that in the Toom graph generated by a fixed matrix A_n there are many minimal IS, and, that different matrices may have equivalent minimal IS. We call A_n-graph the graph generated by a matrix A_n. In the following, we consider Toom graphs generated by A_n (or A_{n_1,n_2}) matrices.

Our model consider weights satisfying these relationships:

$$\texttt{\_1\_2} < \texttt{\_1\_X} < \texttt{\_2\_X} < \texttt{\_X\_Y}$$
$$\texttt{SHIFT} < \texttt{DIV} \quad ; \quad \texttt{SHIFT} < \texttt{STEP}$$

We can also reasonably consider that

$$\texttt{\_1\_2} \leqslant \texttt{SHIFT} \quad ; \quad \texttt{STEP} < \texttt{DIV}$$

For the first one, this is justifiable from the fact that for the two following equivalent processes (A) and (B)

$$(A)\ \begin{array}{l} X \leftarrow 2X; \\ X \leftarrow X + Y; \end{array} \quad ; \quad (B)\ X \leftarrow 2X + Y;$$

it's possible to have a function performing (B) process, which reads X and Y just once. The second relation is satisfied for all the software libraries we have access to.

7. A TRAVEL THROUGH TOOM GRAPHS

What we're really trying to do is practically to solve a minimum weight path problem between α and ω.

Toom graphs are very big. Beginning with Toom-3, it is easy to work with dozens of thousands nodes, and, in order to cope both with memory limitation and time reduction, we developed some strategies not to consider/add some nodes to the graphs. Note that a node ν (and therefore all of its neighbours) might be visited many times, since in general there are many different sequences ("paths") of elementary row operations joining ν with another node ν'.

In general, two opposite approaches are possible to automatically visit the graph by means of a computer program: the *purely functional* one, which does not need the graph to be really constructed, and a second one for which the graph is actually built.

It is clear that a recursive function f visiting a graph G would have no computer memory occupation problems (Gauss' method gives immediately a reasonable upper bound l for IS lengths), but this approach would be extremely time consuming. Infact, in this case G would be implicitly represented by the stack of f nested callings, with length $\leqslant K \times l$ for a fixed constant K, and the needed memory can be precisely estimated from the beginning (avoiding a computational "explosion" because of space lacking). But there is a "time" disadvantage: when an instance of f finishes analysing a node ν, every information about the subgraph G' with ν as initial node is lost. If ν appears again, reached by another path, G' must be completely analysed again.

On the other hand, f could effectively memorise the nodes of a graph while analysing it, in order to recognise if a node has already been inserted, avoiding time loosing. Unfortunately, as said, these graphs are huge, and without a tuned analysis management they grow so much as to rapidly fill all the available memory.

We stay someway in the middle, keeping only *some* nodes for *some* time, so as to avoid until possible – possibly before filling up all the available memory – to repeat analysing nodes, but we also apply the below explained considerations in order to skip "trivial" graph analysis, when the path(s) from the current node to the identity node are trivial or can be sufficiently easily estimated.

7.1 A guided tour of the graph

Let $f(\alpha)$ be, as indicated before, the program visiting the graph having α as its initial node. We introduce a second argument for f, the DESIRED_WEIGHT (DW). Its meaning is related to the fact that we're interested only in IS with $w_{IS} \leqslant DW$, marking as "not interesting" the nodes ν with $w(M_\nu) > DW$.[1]

We implement this marking not as a boolean variable updating, but as two weights (to be compared) we associate to every node ν. They are a lower and upper bound of $w(\nu)$, respectively, and are indicated with $m_w(\nu)$ and $\mathcal{M}_w(\nu)$ (for min and max). The function f returns as result the pair $(m_w(\nu), \mathcal{M}_w(\nu))$. The idea is that making use of DW, it is not always necessary to know precisely the exact weight of a node to classify a matrix as interesting or not.

In the analysis made by $f(\nu, t)$ – where t is the desired weight – for a node ν the following may happen:

$\boxed{m_w(\nu) = \mathcal{M}_w(\nu)}$: this means that ν has already been completely analysed, and $w(\nu) = m_w(\nu) = \mathcal{M}_w(\nu)$. Then f immediately returns the pair $(m_w(\nu), \mathcal{M}_w(\nu))$, with no further computations.

$\boxed{t < m_w(\nu)}$: then f does not need to recurse over and over again, because, so to say, a sufficient number of operations to invert M_ν is not permitted. Then f immediately returns the pair $(m_w(\nu), \mathcal{M}_w(\nu))$. Note that this may happen even if $w(M_\nu)$ has still not been determined exactly, in particular even if ν has never been found before in the graph.

In the above cases, no further analysis is carried beyond ν, and this makes graph analysis faster. Suppose $f(\nu, t)$ is executing, and let ν' be a neighbour of ν, joined to ν by the edge ε, and $t' = t - w(\varepsilon)$. The recursive call $f(\nu', t')$ on ν' will give $w = (m_w(\nu'), \mathcal{M}_w(\nu'))$ as returned value.

At this point, the "parent" instance of f receiving w must try the other neighbours of ν, but before doing this, if $m_w(\nu') = \mathcal{M}_w(\nu')$, then t is updated: $t = \min\{t, m_w(\nu') + w(\varepsilon)\}$. This is a very important step, because if f succeeded in inverting $M_{\nu'}$ with a IS having weight $m_w(\nu') = \mathcal{M}_w(\nu') < t'$, the new value t for DW will be strictly smaller than the precedent one, and all the paths that will be analysed from now will have a stronger restriction, so that pruning is likely to become more and more effective.

7.2 Heuristics for weight estimates

It is clear that the better estimate one is able to give for a matrix weight $w(M)$, the shorter and more efficient the graph analysis will be, both in space and time requirements. We present here some of the ideas we implemented in order to give as best as possible estimates.

A first trivial estimate for $m_w(\nu)$ and $\mathcal{M}_w(\nu)$ can be obtained from the support of $M_\nu \in GL(m, \mathbb{Z})$ and of the single lines and columns, considering that $^{\#}I_m = m$.

DEFINITION 4. *A matrix row $M^{(i)}$ is a* **singleton** *line if all its entries but one are zero, and the non zero entry has value $v = 1$ or $v = -1$. Similarly for columns. A singleton line is indicated with SL_j, where j is the position of v.*

Obviously, the more accurate the estimates are, the less nodes are considered/build in the graph, because the function f has more probabilities to obtain sufficient information from the estimates. By just using columns support cardinality we can define a less than trivial lower bound.

LEMMA 2. *For each column $M_\nu^{[j]}$, $^{\#}M_\nu^{[j]}$ is reduced at most by 1 by a reduction step.*

PROOF. At most one entry is changed in each column. □

LEMMA 3. *A step can produce at most one (more) singleton column in M_ν.*

PROOF. Every column that needs just one operation to become singleton must have exactly two non zero entries. By contradiction, there should be 2 columns as follows, where a, b, c, d are the only non zero entries:

$$\begin{pmatrix} \vdots \\ a \\ \vdots \\ c \\ \vdots \end{pmatrix} \begin{pmatrix} \vdots \\ b \\ \vdots \\ d \\ \vdots \end{pmatrix}$$

To become both singleton, a and b, say, must vanish at the same time, but this means $(a, b) \leftarrow \alpha(a, b) + \beta(c, d) = (0, 0)$. That is, (a, b) and (c, d) are linearly dependent, and this contradicts the fact that M_ν is invertible for every ν. □

[1] In a certain sense, DW works as a threshold, useful to prune Toom graphs analysis.

Let $c_1, \dots c_s$ be all the not zero entries in all the singleton columns of M_ν. We set $w_s(M_\nu) = \sum_{i=1}^{s} w(c_i)$.

THEOREM 3. *Let* $cs[i] = {}^{\#}\{j \mid {}^{\#}M_\nu^{[j]} = i\}$. *Then the following procedure returns a valid* $m_w(\nu)$.

```
result = w_s(M_ν); steps = 0;
for ( i = 2; i < m; i ← i + 1)
  if ( cs[i] ≠ 0 )
    steps = max(steps, i − 2) + cs[i];
result ← result + steps · STEP;
return result;
```

PROOF. At the beginning, `result` contains the weight given by the divisions by the single not zero entries of the singleton columns that must be performed. Let the following be a sparse representation of cs vector, where $i_1 \geqslant 2$:

i	i_1	i_2	$\cdots$	k
$cs[i]$	cs_1	cs_2	$\cdots$	cs_k

One needs at least $(i_h - 2) + cs_h$ steps to transform cs_h columns, each with i_h not zero entries, into singleton ones. First reduce each column support cardinality to 2 (you need at least $i_h - 2$ steps, lemma 2), then (lemma 3) definitely transform them into singleton ones.

The cs_h contribute is necessary: the max takes care of the possibility that more than $i_h - 2$ steps were already done. □

Weight estimation can also be performed line by line, just summing up the corresponding values. There are some interesting particular cases, whose proofs can be found in [2]:

PROPOSITION 2. *Let* $M_\nu^{(i)} = (a, 0, \dots 0, b, 0, \dots, 0, c)$, *with* b *in* j^{th} *position, different from the first and the last one. Then this line-weight estimate can be computed exactly.*

PROPOSITION 3. *If* ${}^{\#}\{i \mid M_\nu^{(i)} \text{ is not singleton}\} = 1$ *then the weight estimate can be computed exactly.*

Similar considerations may be done, *mutatis mutandis*, reasoning with columns.

8. IMPLEMENTATION AND RESULTS

The authors developed `C++` code using STL library to make experiments, and some of the obtained results are presented here. The operations are indicated between matrices using the following notation – i is the index of the modified line:

$i \pm= j$	j^{th} line is added to or subtracted from i^{th} line
$i -= (c)j$	j^{th} line times c is subtracted from i^{th} line
$(c)i \pm= j$	j^{th} line is added (subtracted) to i^{th} line times c
$i\ /= (c)$	i^{th} line entries are divided by c
$i \gg (c)$	i^{th} line entries are divided by 2^c

8.1 Toom-2.5

This case could be treated by hand. We present it as a checking test. Starting from the matrix defined by the reasonably good values $\{\infty, 1, -1, 0\}$ we obtained the (expected) IS indicated below. Note that $\det(A_{3,2}) = 2$: at least a division by 2 (shifting) is necessary. A Toom-graph with 17 nodes was built. The weight is

$$4 \cdot \texttt{STEP} + \texttt{SHIFT}$$

$$A_{3,2} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \overset{2-=3}{\Longrightarrow} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 2 \\ 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \underset{3-=1}{\overset{2\gg(1)}{\Longrightarrow}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \underset{2-=4}{\overset{3+=2}{\Longrightarrow}} I_4$$

There are $n_{IS} = 16$ minimal equivalent IS, involving 19 intermediate matrices.

8.2 Toom-3

GMP solution : We report the IS implemented by Paul Zimmermann for the GMP library [8], starting from the matrix obtained by $\{\infty, 2, -1, 1, 0\}$, and compute its weight w_G according to our definitions.

$$A_3 = \begin{pmatrix} 1 & 0\,0 & 0\,0 \\ 16 & 8\,4 & 2\,1 \\ 1 & -1\,1 & -1\,1 \\ 1 & 1\,1 & 1\,1 \\ 0 & 0\,0 & 0\,1 \end{pmatrix} \overset{2+=(2)3}{\Longrightarrow} \begin{pmatrix} 1 & 0\,0 & 0\,0 \\ 18 & 6\,6 & 0\,3 \\ 1 & -1\,1 & -1\,1 \\ 1 & 1\,1 & 1\,1 \\ 0 & 0\,0 & 0\,1 \end{pmatrix} \underset{3+=4}{\overset{2/=(3)}{\Longrightarrow}} \begin{pmatrix} 1\,0\,0\,0\,0 \\ 6\,2\,2\,0\,1 \\ 2\,0\,2\,0\,2 \\ 1\,1\,1\,1\,1 \\ 0\,0\,0\,0\,1 \end{pmatrix}$$

$$\underset{2\gg(1)}{\overset{2+=5}{\Longrightarrow}} \begin{pmatrix} 1\,0\,0\,0\,0 \\ 3\,1\,1\,0\,1 \\ 2\,0\,2\,0\,2 \\ 1\,1\,1\,1\,1 \\ 0\,0\,0\,0\,1 \end{pmatrix} \overset{2-=(2)1}{\Longrightarrow} \begin{pmatrix} 1\,0\,0\,0\,0 \\ 1\,1\,1\,0\,1 \\ 1\,0\,1\,0\,1 \\ 1\,1\,1\,1\,1 \\ 0\,0\,0\,0\,1 \end{pmatrix} \underset{2-=3}{\overset{4-=2}{\Longrightarrow}} \begin{pmatrix} 1\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0 \\ 1\,0\,1\,0\,1 \\ 0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,1 \end{pmatrix} \underset{3-=5}{\overset{3-=1}{\Longrightarrow}} I_5$$

$$w_G = 8 \cdot \texttt{STEP} + \texttt{DIV} + 2 \cdot \texttt{SHIFT} + 2 \cdot (\texttt{\_1\_2})$$

Surprisingly enough, just by applying criterion **(A)** our program was able to compute solutions with a lesser weight than GMP's. We report two of them, both obtained from the matrix defined by $\{\infty, 2, 1, -1, 0\}$. Their common weight is

$$w_{BZ} = 8 \cdot \texttt{STEP} + \texttt{DIV} + \texttt{SHIFT} + \min(\texttt{\_1\_X}, \texttt{SHIFT}) + \texttt{\_1\_2}$$

and depending on `SHIFT` and `_1_X` values, the below indicated solutions are obtained.

1^{st} solution : `SHIFT` > `_1_X` : Note that $\det(A_3) = 12$, and the division is "split" into a shifting and a division by 6. By using estimates, the nodes really inserted in the Toom-graph are 9982. Considering also criterion **(B)** – applied in the IS used by GMP – the graph gets bigger (49133 nodes), but the minimal found $\overline{IS}$ is equivalent.

Moreover, *independently from weight values*, $w(\overline{IS}) < w_G$.

$$A_3 = \begin{pmatrix} 1 & 0\,0 & 0\,0 \\ 16 & 8\,4 & 2\,1 \\ 1 & 1\,1 & 1\,1 \\ 1 & -1\,1 & -1\,1 \\ 0 & 0\,0 & 0\,1 \end{pmatrix} \overset{2-=4}{\Longrightarrow} \begin{pmatrix} 1 & 0\,0 & 0\,0 \\ 15 & 9\,3 & 3\,0 \\ 1 & 1\,1 & 1\,1 \\ 0 & -2\,0 & -2\,0 \\ 0 & 0\,0 & 0\,1 \end{pmatrix} \underset{4/=(-1)}{\overset{3-=5}{\Longrightarrow}} \begin{pmatrix} 1\,0\,0\,0\,0 \\ 15\,9\,3\,3\,0 \\ 1\,1\,1\,1\,0 \\ 0\,2\,0\,2\,0 \\ 0\,0\,0\,0\,1 \end{pmatrix} \overset{4\gg(1)}{\Longrightarrow}$$

$$\widetilde{A}_3 = \begin{pmatrix} 1\,0\,0\,0\,0 \\ 15\,9\,3\,3\,0 \\ 1\,1\,1\,1\,0 \\ 0\,1\,0\,1\,0 \\ 0\,0\,0\,0\,1 \end{pmatrix} \overset{2-=(3)3}{\Longrightarrow} \begin{pmatrix} 1\,0\,0\,0\,0 \\ 12\,6\,0\,0\,0 \\ 1\,1\,1\,1\,0 \\ 0\,1\,0\,1\,0 \\ 0\,0\,0\,0\,1 \end{pmatrix} \underset{3-=4}{\overset{2/=(6)}{\Longrightarrow}} \begin{pmatrix} 1\,0\,0\,0\,0 \\ 2\,1\,0\,0\,0 \\ 1\,0\,1\,0\,0 \\ 0\,1\,0\,1\,0 \\ 0\,0\,0\,0\,1 \end{pmatrix} \underset{4-=2}{\overset{3-=1}{\Longrightarrow}} \overset{2-=(2)1}{\Longrightarrow} I_5$$

Here $n_{IS} = 78$, with 35 intermediate matrices involved.

2^{nd} solution : `SHIFT` < `_1_X` : The initial operations are the same as above: the different behaviour begins from $\widetilde{A}_3$:

$$\widetilde{A}_3 = \begin{pmatrix} 1\,0\,0\,0\,0 \\ 15\,9\,3\,3\,0 \\ 1\,1\,1\,1\,0 \\ 0\,1\,0\,1\,0 \\ 0\,0\,0\,0\,1 \end{pmatrix} \overset{2/=(3)}{\Longrightarrow} \begin{pmatrix} 1\,0\,0\,0\,0 \\ 5\,3\,1\,1\,0 \\ 1\,1\,1\,1\,0 \\ 0\,1\,0\,1\,0 \\ 0\,0\,0\,0\,1 \end{pmatrix} \overset{2-=3}{\Longrightarrow} \begin{pmatrix} 1\,0\,0\,0\,0 \\ 4\,2\,0\,0\,0 \\ 1\,1\,1\,1\,0 \\ 0\,1\,0\,1\,0 \\ 0\,0\,0\,0\,1 \end{pmatrix}$$

$$\underset{3-=4}{\overset{2\gg(1)}{\Longrightarrow}} \begin{pmatrix} 1\,0\,0\,0\,0 \\ 2\,1\,0\,0\,0 \\ 1\,0\,1\,0\,0 \\ 0\,1\,0\,1\,0 \\ 0\,0\,0\,0\,1 \end{pmatrix} \overset{2-=(2)1}{\Longrightarrow} \begin{pmatrix} 1\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0 \\ 0\,0\,1\,0\,0 \\ 0\,1\,0\,1\,0 \\ 0\,0\,0\,0\,1 \end{pmatrix} \overset{4-=2}{\Longrightarrow} I_5$$

Here $n_{IS} = 456$, considering just criterion (**A**), with 35 intermediate matrices involved, and 706 by considering both criteria, with 40 intermediate matrices.

We tested all matrices obtained from $\{\infty, a, b, c, 0\}$, where $a, b, c \in \left\{\pm 1, \pm 2, \dots, \pm 16, \pm\frac{1}{2}, \pm\frac{1}{3}, \dots, \pm\frac{1}{16}\right\}$. No IS with

weight smaller than the two presented above was found. Paths with the same weight are substantially equivalent to these ones and were found using as interpolating points $\{\infty, 1, -1, \pm 2, 0\}$ and $\{\infty, 1, -1, \pm\frac{1}{2}, 0\}$.

In all our tests, for many different weight values, we always obtained w_{BZ} as result. We think that it is the case for *all* possible weights.

The authors developed GMP code implementing the second Toom-3 IS. Figure (1) shows the relative performance with respect to the current – version 4.2.1 – implementation. This improvement and the further results in [1] concerning evaluation could contribute to modify the threshold values selecting Karatsuba, Toom-3 and FFT method use in GMP multiplication.

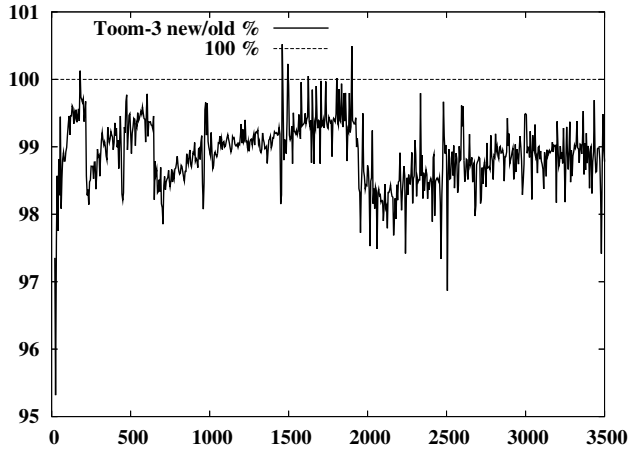

Figure 1: Comparison: Toom-3 new/old (GMP)

8.3 Toom-3.5

Starting from the matrix defined by $\{\infty, 2, -2, 1, -1, 0\}$ we obtained the IS indicated below, with weight

$$12 \cdot \texttt{STEP} + 2 \cdot \texttt{DIV} + 2 \cdot \texttt{SHIFT} + 2 \cdot (\texttt{\_1\_2})$$

We note that in this case a regularization step (criterion **B**) is necessary (step n. 12) to obtain a minimal IS.

$$A_{4,3} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 32 & 16 & 8 & 4 & 2 & 1 \\ -32 & 16 & -8 & 4 & -2 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

1) $3 - = 2$
2) $5 - = 4$
3) $2 - = 6$
4) $3 \gg (1)$
5) $2 + = 3$
6) $4 - = 6$
7) $5/ = (-1)$
8) $3 + = 5$
9) $5 \gg (1)$
10) $3/ = (-6)$
11) $4 - = 5$
12) $3 - = (4)1$
13) $2 - = (4)4$
14) $5 - = 3$
15) $2/ = (12)$
16) $3 - = 1$
17) $4 - = 2$

9. CONCLUSIONS

We presented a method to determine optimal (with respect to the model given by **(A)** and **(B)** criteria) sequences of basic operations inverting matrices appearing in Toom-Cook methods through an optimised research on graphs. Weights for each basic operation are introduced, measuring the inversion cost for the matrices appearing in intermediate steps. An interval-like estimate analysis permits to know in advance that a path is not optimal, saving thus time and space.

Intermediate version of Toom methods are also presented. As an interesting result, we were able to determine two new inversion sequences for Toom-3 matrix, which seem to be good alternatives to the currently implemented ones.

10. ACKNOWLEDGEMENTS

This work was partially supported by FILAS. The authors want to thank Torbjörn Granlund and Paul Zimmermann for helpful discussions, and the anonymous reviewers for their valuable suggestions and corrections.

11. REFERENCES

[1] M. Bodrato. Towards optimal Toom-Cook multiplication for univariate and multivariate polynomials in characteristic 2 and 0. In C. Carlet and B. Sunar, editors, *WAIFI 2007 proceedings*, volume 4547 of *LNCS*, pages 116–133. Springer, June 2007. `http://bodrato.it/papers/#WAIFI2007`.

[2] M. Bodrato and A. Zanoni. What about Toom-Cook matrices optimality ? Technical Report 605, Centro "Vito Volterra", Università di Roma "Tor Vergata", October 2006. `http://bodrato.it/papers/#CIVV2006`.

[3] J. Chung and M. A. Hasan. Asymmetric squaring formulae. Technical Report 24, University of Waterloo, August 2006.

[4] J. Chung and M. A. Hasan. Asymmetric squaring formulae. In *Proceedings of the ARITH-18 conference*. IEEE, June 2007.

[5] S. A. Cook. *On the minimum computation time of functions*. PhD thesis, Dept. of Mathematics, Harvard University, 1966.

[6] T. S. Denis, M. Rasmussen, and G. Rose. Multi-precision math (tommath library documentation). `http://math.libtomcrypt.com/files/tommath.pdf`.

[7] P. Gaudry, A. Kruppa, and P. Zimmermann. A GMP-based implementation of Schönhage-Strassen's large integer multiplication algorithm. In C. W. Brown, editor, *Proceedings of the ISSAC 2007 conference*. ACM press, July 2007.

[8] GNU MP: The GNU multiple precision arithmetic library, v4.2.1, 2006. `http://gmplib.org/#DOC`.

[9] A. A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7(7):595–596, 1963.

[10] D. E. Knuth. *The Art of Computer Programming. Volume 2: Seminumerical Algorithms*. Addison-Wesley, 1981.

[11] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7(3–4):281–292, 1971.

[12] A. L. Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. *Soviet Mathematics Doklady*, 3:714–716, 1963.

[13] D. Zuras. More on squaring and multiplying large integers. *IEEE Transactions on Computers*, 43(8):899–908, August 1994.

APPENDIX

A. TOOM-4, TOOM-4.5 AND TOOM-5

We present some IS for the indicated methods, that were found not in a completely automatic way. Because of limited computing resources, some initial steps were done manually and the graph analysis starts from the so reached matrix.

A.1 Toom-4

Starting from the matrix defined by $\{\infty, 2, 1, -1, \frac{1}{2}, -\frac{1}{2}, 0\}$ we obtained the below IS, with weight

$$\texttt{18}\cdot\texttt{STEP} + \texttt{3}\cdot\texttt{DIV} + \texttt{SHIFT} + \min(\texttt{\_1\_X}\ ,\ \texttt{SHIFT}) + \texttt{2}\cdot(\texttt{\_1\_X}) + \texttt{4}\cdot(\texttt{\_1\_2})$$

$\boxed{1^{st}\ \textbf{solution}}$: $\texttt{SHIFT} > \texttt{\_1\_X}$

$$A_4 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 64 & 32 & 16 & 8 & 4 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 2 & 4 & 8 & 16 & 32 & 64 \\ 1 & -2 & 4 & -8 & 16 & -32 & 64 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

1) $2+=5$
2) $4-=3$
3) $6-=5$
4) $4 \gg 1$
5) $5-=1$
6) $5-=(64)7$
7) $3+=4$
8) $(2)5+=6$
9) $2-=(65)3$
10) $3-=1$
11) $3-=7$
12) $4/=(-1)$
13) $6/=(-1)$
14) $2+=(45)3$
15) $5-=(8)3$
16) $5/=(24)$
17) $6-=2$
18) $2-=(16)4$
19) $2/=(18)$
20) $3-=5$
21) $4-=2$
22) $6+=(30)2$
23) $6/=(60)$
24) $2-=6$

$\boxed{2^{nd}\ \textbf{solution}}$: $\texttt{SHIFT} < \texttt{\_1\_X}$

The first 21 operations, leading to the below matrix $\widetilde{A}_4$, are the same as above: then the IS changes as follows:

$$\widetilde{A}_4 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -30 & 0 & 0 & 0 & 30 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

22) $6/=(30)$ 23) $6+=2$ 24) $6 \gg (1)$ 25) $2-=6$

A.2 Toom-4.5

Starting from the matrix defined by $\{\infty, -1, -2, \frac{1}{2}, 1, 2, -\frac{1}{2}, 0\}$ we obtained the below IS, with weight

$$\texttt{22}\cdot\texttt{STEP} + \texttt{4}\cdot\texttt{DIV} + \texttt{SHIFT} + \texttt{3}\cdot(\texttt{\_1\_X}) + \texttt{6}\cdot(\texttt{\_1\_2})$$

$$A_{5,4} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -128 & 64 & -32 & 16 & -8 & 4 & -2 & 1 \\ 1 & 2 & 4 & 8 & 16 & 32 & 64 & 128 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \\ 1 & -2 & 4 & -8 & 16 & -32 & 64 & -128 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

1) $2+=5$
2) $3-=6$
3) $7+=4$
4) $4-=2$
5) $(2)6+=3$
6) $2 \gg (1)$
7) $5-=2$
8) $6-=(128)2$
9) $(2)3+=7$
10) $4+=6$
11) $2-=8$
12) $(2)4-=7$
13) $6+=(126)8$
14) $7-=(32)5$
15) $(5)7-=3$
16) $4/=(-180)$
17) $6/=(-24)$
18) $3+=(510)1$
19) $3/=(-120)$
20) $7/=(360)$
21) $6-=(4)4$
22) $2-=4$
23) $5-=7$
24) $4-=6$
25) $7-=1$
26) $3+=7$
27) $5-=3$

A.3 Toom-5

Starting from the matrix defined by $\{\infty, -2, \frac{1}{2}, 4, 2, -1, 1, -\frac{1}{2}, 0\}$ we obtained the below IS, with weight

$$\texttt{32}\cdot\texttt{STEP} + \texttt{5}\cdot\texttt{DIV} + \texttt{2}\cdot\texttt{SHIFT} + \texttt{6}\cdot(\texttt{\_1\_X}) + \texttt{8}\cdot(\texttt{\_1\_2})$$

$$A_5 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 256 & -128 & 64 & -32 & 16 & -8 & 4 & -2 & 1 \\ 1 & 2 & 4 & 8 & 16 & 32 & 64 & 128 & 256 \\ 4^8 & 4^7 & 4^6 & 4^5 & 256 & 64 & 16 & 4 & 1 \\ 256 & 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -2 & 4 & -8 & 16 & -32 & 64 & -128 & 256 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

1) $6-=7$
2) $2-=5$
3) $4-=9$
4) $4-=(2^{16})1$
5) $8-=3$
6) $6 \gg (1)$
7) $(2)5+=2$
8) $2/=(-1)$
9) $8/=(-1)$
10) $7+=6$
11) $6/=(-1)$
12) $3-=7$
13) $5-=(512)7$
14) $(2)3-=8$
15) $7-=1$
16) $7-=9$
17) $8+=2$
18) $5+=3$
19) $8-=(80)6$
20) $3-=(510)9$
21) $4-=2$
22) $(3)3+=5$
23) $8/=(180)$
24) $5+=(378)7$
25) $2 \gg (2)$
26) $6-=2$
27) $5/=(-72)$
28) $3/=(-360)$
29) $2-=8$
30) $7-=3$
31) $4-=(256)5$
32) $3-=5$
33) $4-=(4096)3$
34) $4-=(16)7$
35) $4+=(256)6$
36) $6+=2$
37) $(180)2+=4$
38) $2/=(11340)$
39) $4+=(720)6$
40) $4/=(-2160)$
41) $6-=4$
42) $8-=2$

B. ASYMMETRICAL SQUARING

Chung and Anwar Hasan [3] showed different linear systems which can be built, but only for squaring. Their report proposed an inversion algorithm for the 5-way squaring method using temporary variables with a cost of $\texttt{18}\cdot\texttt{STEP} + \texttt{7}\cdot\texttt{SHIFT}$. Using the same matrix, we found the following IS, with no temporary variables and smaller weight:

$$\texttt{16}\cdot\texttt{STEP} + \texttt{3}\cdot\texttt{SHIFT}$$

$$A_5^s = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 1 & 1 & 0 & -1 & -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

1) $3+=4$
2) $7-=1$
3) $7-=2$
4) $7-=8$
5) $3 \gg (1)$
6) $5+=3$
7) $4-=3$
8) $5 \gg (1)$
9) $6+=4$
10) $4-=8$
11) $6 \gg (1)$
12) $3-=5$
13) $4-=6$
14) $5-=1$
15) $5-=9$
16) $6-=2$
17) $7+=5$
18) $7+=4$
19) $3-=7$

In [4] the authors report the complete algorithm with the above improvement. Further refinements can be found in [1].