Fast and stable matrix multiplication

Olga Holtz

Department of Mathematics

University of California-Berkeley

holtz@math.berkeley.edu

joint work James Demmel, Ioana Dumitriu and Robert Kleinberg

The quest for speed

How fast can one multiply two $n \times n$ matrices?

- Standard multiplication: $O(n^3)$ operations.

The quest for speed

How fast can one multiply two $n \times n$ matrices?

- Standard multiplication: $O(n^3)$ operations.
- Strassen's algorithm: $O(n^{2.81})$ operations.

The quest for speed

How fast can one multiply two $n \times n$ matrices?

- Standard multiplication: $O(n^3)$ operations.
- Strassen's algorithm: $O(n^{2.81})$ operations.
- ...

The quest for speed

How fast can one multiply two $n \times n$ matrices?

- Standard multiplication: $O(n^3)$ operations.
- Strassen's algorithm: $O(n^{2.81})$ operations.
- ...
- Coppersmith and Winograd's algorithm: $O(n^{2.38})$ operations.

The quest for speed

How fast can one multiply two $n \times n$ matrices?

- Standard multiplication: $O(n^3)$ operations.
- Strassen's algorithm: $O(n^{2.81})$ operations.
- ...
- Coppersmith and Winograd's algorithm: $O(n^{2.38})$ operations.
- Is $O(n^2)$ achievable?

Why should we care?

Complexity of matrix multiplication = complexity of "almost all" matrix problems:

- solving linear systems,
- evaluating determinants,
- LU factorization,
- many more.

Why should we care?

Complexity of matrix multiplication = complexity of "almost all" matrix problems:

- solving linear systems,
- evaluating determinants,
- LU factorization,
- many more.

See P. Bürgisser, M. Clausen, M. A. Shokrollahi Algebraic complexity theory.

Strassen's algorithm

Volker Strassen

Gaussian elimination is not optimal. **Numer. Mathematik [1969].**

Strassen's algorithm

Volker Strassen

Gaussian elimination is not optimal. **Numer. Mathematik [1969].**

Main idea:

- Multiplication by recursively partitioning into smaller blocks.
- To be faster than $O(n^3)$, this needs a method to multiply small matrices (order k) using $o(k^3)$ multiplications.

Strassen

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

requires only 7 multiplications:

$$\begin{aligned}
M_1 &:= (A_{11} + A_{22})(B_{11} + B_{22}) \\
M_2 &:= (A_{21} + A_{22})B_{11} \\
M_3 &:= A_{11}(B_{12} - B_{22}) \\
M_4 &:= A_{22}(B_{21} - B_{11}) \\
M_5 &:= (A_{11} + A_{12})B_{22} \\
M_6 &:= (A_{21} - A_{11})(B_{11} + B_{12}) \\
M_7 &:= (A_{12} - A_{22})(B_{21} + B_{22}).
\end{aligned}$$

Strassen

Then $$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix},$$

where $$\begin{aligned} C_{11} &:= M_1 + M_4 - M_5 + M_7 \\ C_{12} &:= M_3 + M_5 \\ C_{21} &:= M_2 + M_4 \\ C_{22} &:= M_1 - M_2 + M_3 + M_6. \end{aligned}$$

Strassen

Then $\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$,

where
$$\begin{aligned} C_{11} &:= M_1 + M_4 - M_5 + M_7 \\ C_{12} &:= M_3 + M_5 \\ C_{21} &:= M_2 + M_4 \\ C_{22} &:= M_1 - M_2 + M_3 + M_6. \end{aligned}$$

Applied recursively, this yields running time $O(n^{\log_2 7}) \approx O(n^{2.8})$.

Coppersmith and Winograd

Don Coppersmith Shmuel Winograd

Matrix multiplication via arithmetic progressions. **Journal of Symbolic Computation [1990].**

Coppersmith and Winograd

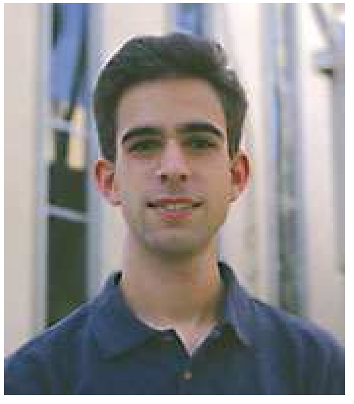
Don Coppersmith

Shmuel Winograd

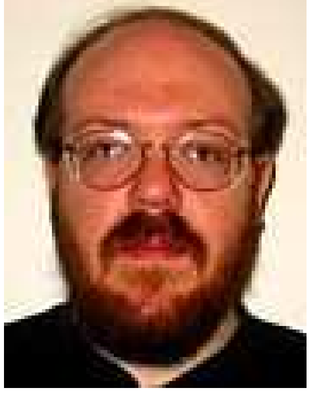
Matrix multiplication via arithmetic progressions. **Journal of Symbolic Computation [1990].**

Used a thm on dense sets of integers containing no three terms in arithmetic progression (R. Salem & D. C. Spencer [1942]) to get an algorithm with running time $\approx O(n^{2.376})$.

Group-theoretic approach



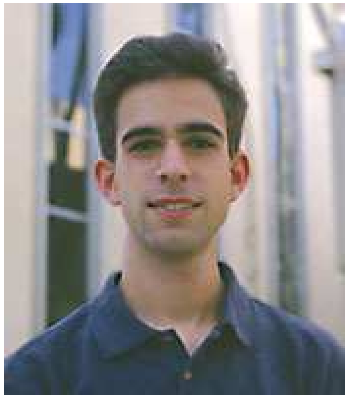
Chris Umans



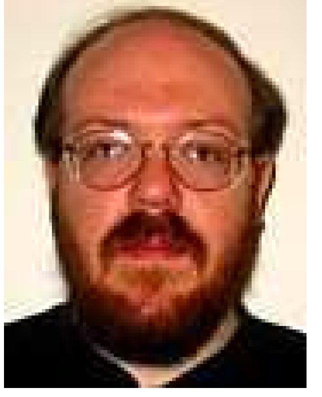
Henry Cohn

A group-theoretic approach to matrix multiplication, **FOCS Proceedings [2003].**

Group-theoretic approach



Chris Umans



Henry Cohn

A group-theoretic approach to matrix multiplication, **FOCS Proceedings [2003].**

Proposed embedding into group algebra to be combined with recursive partitioning.

Why group algebras?

Multiplying two polynomials has complexity $O(n \log n)$ instead of $O(n^2)$ by embedding coefficients into $\mathbb{C}[G]$ where G is a finite cyclic group of order $N \geq 2n$, via the map

$$p \mapsto \{p(w) : w = \exp(2\pi k \mathrm{i}/N)\}_{k=0,\ldots,N-1}.$$

$$\boxed{\begin{array}{l}\text{embed}\\ \text{into } \mathbb{C}[G]\end{array}} \rightarrow \boxed{\begin{array}{l}\text{convolve}\\ \text{using FFT}\end{array}} \rightarrow \boxed{\begin{array}{l}\text{extract}\\ \text{from } \mathbb{C}[G]\end{array}}.$$

Why group algebras?

Multiplying two polynomials has complexity $O(n \log n)$ instead of $O(n^2)$ by embedding coefficients into $\mathbb{C}[G]$ where G is a finite cyclic group of order $N \geq 2n$, via the map

$$p \mapsto \{p(w) : w = \exp(2\pi k\mathrm{i}/N)\}_{k=0,\ldots,N-1}.$$

$$\boxed{\begin{array}{l}\text{embed}\\ \text{into } \mathbb{C}[G]\end{array}} \longrightarrow \boxed{\begin{array}{l}\text{convolve}\\ \text{using FFT}\end{array}} \longrightarrow \boxed{\begin{array}{l}\text{extract}\\ \text{from } \mathbb{C}[G]\end{array}}.$$

The same can be done with matrix products, via the map $A \mapsto \sum_{x,y} A(x,y)x^{-1}y$. The group G must have special properties.

The algorithm

- Embed A, B in group algebra

The algorithm

- Embed A, B in group algebra
- Perform FFT

The algorithm

- Embed A, B in group algebra
- Perform FFT
- Reorganize results into new matrices

The algorithm

- Embed A, B in group algebra
- Perform FFT
- Reorganize results into new matrices
- Multiply new matrices recursively

The algorithm

- Embed A, B in group algebra
- Perform FFT
- Reorganize results into new matrices
- Multiply new matrices recursively
- Reorganize results into new matrices

The algorithm

- Embed A, B in group algebra
- Perform FFT
- Reorganize results into new matrices
- Multiply new matrices recursively
- Reorganize results into new matrices
- Perform Inverse FFT

The algorithm

- Embed A, B in group algebra
- Perform FFT
- Reorganize results into new matrices
- Multiply new matrices recursively
- Reorganize results into new matrices
- Perform Inverse FFT
- Extract $C = AB$ from group algebra

Properties required

- For unambiguous embedding into $\mathbb{C}[G]$ there must be three subgroups H_1, H_2, H_3 with the triple product property :

  $$h_1 h_2 h_3 = 1, h_i \in H_i \implies h_1 = h_2 = h_3 = 1$$

  (can be generalized to other subsets of G).

Properties required

- For unambiguous embedding into $\mathbb{C}[G]$ there must be three subgroups H_1, H_2, H_3 with the triple product property :

$$h_1h_2h_3 = 1, h_i \in H_i \implies h_1 = h_2 = h_3 = 1$$

(can be generalized to other subsets of G).

- For the resulting algorithm to be faster than $O(n^3)$, we must beat the sum of the cubes:

$$|H_1|\,|H_2|\,|H_3| > \sum_j d_j^3$$

(d_j are the character degrees of G).

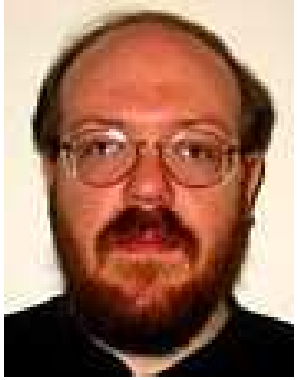
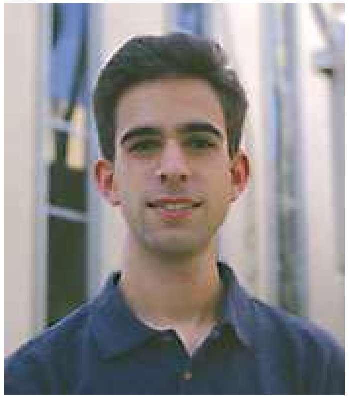
Wedderburn's theorem

Theorem. The group algebra of a finite group G decomposes as the direct product

$$\mathbb{C}[G] \cong \mathbb{C}^{d_1 \times d_1} \times \cdots \times \mathbb{C}^{d_k \times d_k}$$

of matrix algebras of orders $d_1, \ldots, d_k$. These orders are the character degrees of G, or the dimensions of its irreducible representations.

Beating the sum of the cubes

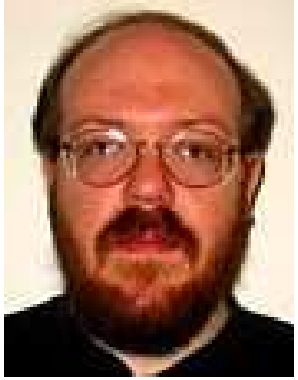
Balázs Szegedy   Henry Cohn   Chris Umans   Bobby Kleinberg

Group-theoretic algorithms for matrix multiplication,

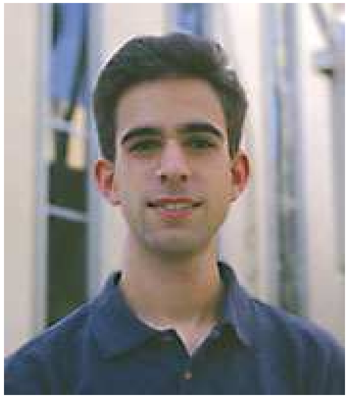
FOCS Proceedings [2005].

Beating the sum of the cubes

Balázs Szegedy



Henry Cohn



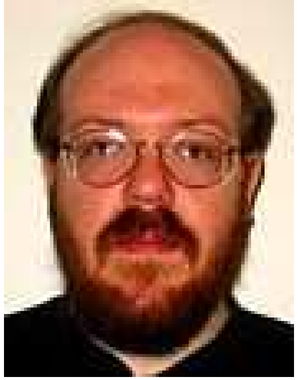
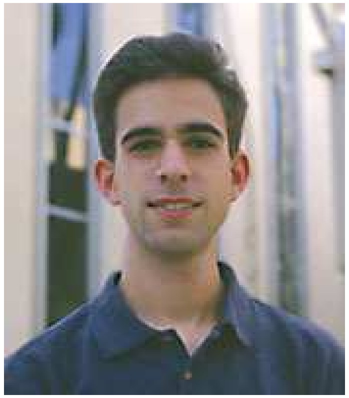
Chris Umans

Bobby Kleinberg

Group-theoretic algorithms for matrix multiplication,
FOCS Proceedings [2005].

Found groups with subsets beating the sum of the cubes and satisfying the triple product property.

Beating the sum of the cubes

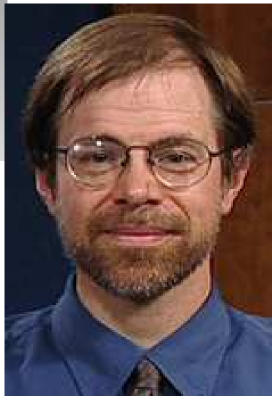

Balázs Szegedy   Henry Cohn   Chris Umans   Bobby Kleinberg

Group-theoretic algorithms for matrix multiplication,
FOCS Proceedings [2005].

Found groups with subsets beating the sum of the cubes and satisfying the triple product property.

Press coverage: *SIAM News [Nov 2005]* by Sara Robinson.

Error analysis



Jim Demmel

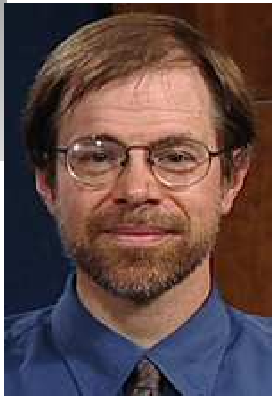
Ioana Dumitriu

Olga Holtz

Bobby Kleinberg

Fast matrix multiplication is stable, **ArXiv Math.NA/0603207 [2006].**

Error analysis



Jim Demmel

Ioana Dumitriu

Olga Holtz

Bobby Kleinberg

Fast matrix multiplication is stable, **ArXiv Math.NA/0603207 [2006].**

Main question: Do you get the right answer in the presence of roundoff? To answer, need error analysis for a large class of recursive matrix multiplication algorithms.

Method

Forward error analysis in the spirit of

D. Bini and G. Lotti ***Stability of fast algorithms for matrix multiplication,*** **Numer. Mathematik [1980/81]**.

Method

Forward error analysis in the spirit of

D. Bini and G. Lotti ***Stability of fast algorithms for matrix multiplication,*** **Numer. Mathematik [1980/81]**.

What was missing:

- More general roundoff assumptions
- Wider scope:

  nonstationary algorithms

  algorithms with pre- and post- processing

Recursive matmul algorithms

aka Bilinear noncommutative algorithms

- *Stationary partitioning* algorithms: at each step, split matrices into the same number k^2 of square blocks.

Recursive matmul algorithms

aka Bilinear noncommutative algorithms

- *Stationary partitioning* algorithms: at each step, split matrices into the same number k^2 of square blocks.
- *Non-stationary partitioning* algorithms: the number of blocks may vary at each step.

Recursive matmul algorithms

aka Bilinear noncommutative algorithms

- *Stationary partitioning* algorithms: at each step, split matrices into the same number k^2 of square blocks.
- *Non-stationary partitioning* algorithms: the number of blocks may vary at each step.
- Partitioning may be combined with *pre- and post-processing*, both linear maps that introduce roundoff errors.

In all cases, the error bounds have the form

$$\|C_{comp} - C\| \le c n^d \varepsilon \|A\| \cdot \|B\| + O(\varepsilon^2),$$

where	c, d	are	modest constants,
	ε		machine precision,
	n		order of A, B, $C = AB$,
	C_{comp}		computed value of C.

Cf. with error bound for n^3-algorithm:

$$|C_{comp} - C| \le c n \varepsilon |A| \cdot |B| + O(\varepsilon^2).$$

Group-theoretic algorithms

- Embed A, B in group algebra (exact)

Group-theoretic algorithms

- Embed A, B in group algebra (exact)
- Perform FFT (roundoff)

Group-theoretic algorithms

- Embed A, B in group algebra (exact)
- Perform FFT (roundoff)
- Reorganize results into new matrices (exact)

Group-theoretic algorithms

- Embed A, B in group algebra (exact)
- Perform FFT (roundoff)
- Reorganize results into new matrices (exact)
- Multiply new matrices recursively (roundoff)

Group-theoretic algorithms

- Embed A, B in group algebra (exact)
- Perform FFT (roundoff)
- Reorganize results into new matrices (exact)
- Multiply new matrices recursively (roundoff)
- Reorganize results into new matrices (exact)

Group-theoretic algorithms

- Embed A, B in group algebra (exact)
- Perform FFT (roundoff)
- Reorganize results into new matrices (exact)
- Multiply new matrices recursively (roundoff)
- Reorganize results into new matrices (exact)
- Perform Inverse FFT (roundoff)

Group-theoretic algorithms

- Embed A, B in group algebra (exact)
- Perform FFT (roundoff)
- Reorganize results into new matrices (exact)
- Multiply new matrices recursively (roundoff)
- Reorganize results into new matrices (exact)
- Perform Inverse FFT (roundoff)
- Extract $C = AB$ from group algebra (exact)

Semi-direct product, wreath product

If H is any group and Q is a group which acts (on the left) by automorphisms of H, with $q \cdot h$ denoting the action of $q \in Q$ on $h \in H$, then the semidirect product $H \rtimes Q$ is the set of ordered pairs (h, q) with the multiplication law

$$(h_1, q_1)(h_2, q_2) = (h_1(q_1 \cdot h_2), q_1 q_2).$$

Semi-direct product, wreath product

If H is any group and Q is a group which acts (on the left) by automorphisms of H, with $q \cdot h$ denoting the action of $q \in Q$ on $h \in H$, then the semidirect product $H \rtimes Q$ is the set of ordered pairs (h, q) with the multiplication law

$$(h_1, q_1)(h_2, q_2) = (h_1(q_1 \cdot h_2), q_1 q_2).$$

If H is any group, S is any finite set, and Q is a group with a left action on S, the wreath product $H \wr Q$ is the semidirect product $(H^S) \rtimes Q$ where Q acts on the direct product of $|S|$ copies of H by permuting the coordinates according to the action of Q on S.

Running example, I

Consider the set $S = \{0, 1\}$ and a two-element group Q whose non-identity element acts on S by swapping 0 and 1. Let H be the group $(\mathbb{Z}/16)^3$. An element of H^S is an ordered pair of elements of H:

$$\begin{pmatrix} x_{00} & x_{01} & x_{02} \\ x_{10} & x_{11} & x_{12} \end{pmatrix}.$$

An element of $H \wr Q$ is an ordered pair (X, q) where X is a matrix as above, and $q = \pm 1$. Example:

$$\begin{array}{c} (X, -1) \\ \cdot (Y, -1) \end{array} = \left(\begin{pmatrix} x_{00} + y_{10} & x_{01} + y_{11} & x_{02} + y_{12} \\ x_{10} + y_{00} & x_{11} + y_{01} & x_{12} + y_{02} \end{pmatrix}, 1 \right)$$

Triple product property

If S, T are subsets of a group G, let $Q(S, T)$ denote their right quotient set:

$$\begin{aligned} Q(S,T) &:= \{st^{-1} : s \in S, t \in T\}, \\ Q(S) &:= Q(S,S). \end{aligned}$$

Triple product property

If S, T are subsets of a group G, let $Q(S,T)$ denote their right quotient set:

$$\begin{aligned} Q(S,T) &:= \{st^{-1} : s \in S, t \in T\}, \\ Q(S) &:= Q(S,S). \end{aligned}$$

Definition. If H is a group and X, Y, Z are three subsets, we say X, Y, Z satisfy the triple product property if, for all $q_x \in Q(X)$, $q_y \in Q(Y)$, $q_z \in Q(Z)$, the condition $q_x q_y q_z = 1$ implies $q_x = q_y = q_z = 1$.

Simultaneous triple product property

If $\{(X_i, Y_i, Z_i) : i \in I\}$ is a collection of ordered triples of subsets of H, we say that this collection satisfies the simultaneous triple product property (STPP) if, for all $i, j, k \in I$ and all $q_x \in Q(X_i, X_j)$, $q_y \in Q(Y_j, Y_k)$, $q_z \in Q(Z_k, Z_i)$, the condition $q_x q_y q_z = 1$ implies $q_x{=}q_y{=}q_z{=}1$ and $i{=}j{=}k$.

Simultaneous triple product property

If $\{(X_i, Y_i, Z_i)\ :\ i \in I\}$ is a collection of ordered triples of subsets of H, we say that this collection satisfies the simultaneous triple product property (STPP) if, for all $i, j, k \in I$ and all $q_x \in Q(X_i, X_j)$, $q_y \in Q(Y_j, Y_k)$, $q_z \in Q(Z_k, Z_i)$, the condition $q_x q_y q_z = 1$ implies $q_x{=}q_y{=}q_z{=}1$ and $i{=}j{=}k$.

Lemma If a group H has subsets $\{X_i, Y_i, Z_i\}_{1 \le i \le n}$ satisfying the simultaneous triple product property, then for every element $h\pi$ in $H \wr \mathrm{Sym}_n$ there is at most one way to represent $h\pi$ as a quotient $(x\sigma)^{-1} y\tau$ such that $x \in \prod_{i=1}^{n} X_i$, $y \in \prod_{i=1}^{n} Y_i$, $\sigma, \tau \in \mathrm{Sym}_n$.

Running example, II

In our running example, the group H is $(\mathbb{Z}/16\mathbb{Z})^3$. Consider the following three subgroups of H.

$$\begin{aligned} X &:= (\mathbb{Z}/16\mathbb{Z}) \times \{0\} \times \{0\} \\ Y &:= \{0\} \times (\mathbb{Z}/16\mathbb{Z}) \times \{0\} \\ Z &:= \{0\} \times \{0\} \times (\mathbb{Z}/16\mathbb{Z}) \end{aligned}$$

Then X, Y, Z satisfy the triple product property: if $q_x \in Q(X), q_y \in Q(Y), q_z \in Q(Z)$, and $q_x + q_y + q_z = 0$, then $q_x = q_y = q_z = 0$.

Running example, IIa

Now consider the following six subsets of H:

$$\begin{aligned}
\overline{X}_0 &:= \{1,2,\ldots,15\} \times \{0\} \times \{0\} \\
\overline{Y}_0 &:= \{0\} \times \{1,2,\ldots,15\} \times \{0\} \\
\overline{Z}_0 &:= \{0\} \times \{0\} \times \{1,2,\ldots,15\} \\
\overline{X}_1 &:= \{0\} \times \{1,2,\ldots,15\} \times \{0\} \\
\overline{Y}_1 &:= \{0\} \times \{0\} \times \{1,2,\ldots,15\} \\
\overline{Z}_1 &:= \{1,2,\ldots,15\} \times \{0\} \times \{0\}.
\end{aligned}$$

Then $(\overline{X}_0, \overline{Y}_0, \overline{Z}_0)$ and $(\overline{X}_1, \overline{Y}_1, \overline{Z}_1)$ satisfy the simultaneous triple product property.

Discrete Fourier transform

If H is an abelian group, let $\widehat{H}$ denote the set of all homomorphisms from H to S^1 aka characters. Canonical bijection $(\chi_1, \chi_2) \mapsto \chi$:

$$\chi(h_1, h_2) = \chi_1(h_1)\chi_2(h_2).$$

There is a left action of Sym_n on the set $\widehat{H}^n$:

$$\sigma \cdot (\chi_1, \chi_2, \ldots, \chi_n) := (\chi_{\sigma^{-1}(1)}, \chi_{\sigma^{-1}(2)}, \ldots, \chi_{\sigma^{-1}(n)}).$$

Denote by $\Xi(H^n)$ a subset of $\widehat{H}^n$ containing exactly one representative of each orbit of the Sym_n action on $\widehat{H}^n$. Note $|\Xi(H^n)| = \binom{|H|+n-1}{n}$.

Running example, III

A character χ of the group $H = (\mathbb{Z}/16\mathbb{Z})^3$ is uniquely determined by a triple (a_1, a_2, a_3) of integers modulo 16. For an element $h = (b_1, b_2, b_3) \in H$,

$$\chi(h) = e^{2\pi i(a_1 b_1 + a_2 b_2 + a_3 b_3)/16}.$$

A character of the group H^2 may be represented as

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}$$

as before. The group $\mathrm{Sym}_2 = \{\pm 1\}$ acts on $\widehat{H^2}$ by exchanging the two rows of such a matrix. The set $\Xi(H^2)$ has cardinality $\binom{4096}{2} + 4096 = 8{,}390{,}656.$

Abelian STP families

An abelian STP family with growth parameters (α, β) is a collection of ordered triples (H_N, Υ_N, k_N), satisfying

1. H_N is an abelian group.
2. $\Upsilon_N = \{(X_i, Y_i, Z_i) \,:\, i = 1, 2, \ldots, N\}$ is a collection of N ordered triples of subsets of H_N satisfying the simultaneous triple product property.
3. $|H_N| = N^{\alpha + o(1)}$.
4. $k_N = \prod_{i=1}^{N} |X_i| = \prod_{i=1}^{N} |Y_i| = \prod_{i=1}^{N} |Z_i| = N^{\beta N + o(N)}$.

Remark

If $\{(H_N, \Upsilon_N, k_N)\}$ is an abelian STP family, then Lemma above ensures that there is a 1-1 mapping

$$\left(\prod_{i=1}^{N} X_i\right) \times \left(\prod_{i=1}^{N} Y_i\right) \times (\mathrm{Sym}_N)^2 \longrightarrow H_N \wr \mathrm{Sym}_N$$

given by $(x, y, \sigma, \tau) \mapsto (x\sigma)^{-1} y\tau$.

Remark

If $\{(H_N, \Upsilon_N, k_N)\}$ is an abelian STP family, then Lemma above ensures that there is a 1-1 mapping

$$\left(\prod_{i=1}^{N} X_i\right) \times \left(\prod_{i=1}^{N} Y_i\right) \times (\mathrm{Sym}_N)^2 \longrightarrow H_N \wr \mathrm{Sym}_N$$

given by $(x, y, \sigma, \tau) \mapsto (x\sigma)^{-1} y\tau$. This implies also

$$\begin{aligned} |H_N|^N N! &\geq (k_N N!)^2 \\ N^{\alpha N + o(N)} N^{N + o(N)} &\geq N^{2\beta N + o(N)} N^{2N + o(N)} \\ \alpha + 1 &\geq 2\beta + 2 \\ \frac{\alpha - 1}{\beta} &\geq \frac{\alpha + 1}{\beta + 1} \geq 2. \end{aligned}$$

Running example, IV

Extend our example to an abelian STP family. For $N \geq 1$ let $\ell = \lceil \log_2(N) \rceil$ and let $H_N = H^\ell$. For $1 \leq i \leq N$ let $i_1, i_2, \ldots, i_\ell$ denote the binary digits of the number $i - 1$ (padded with initial 0's so that it has exactly ℓ digits) and let

$$X_i = \prod_{m=1}^{\ell} \bar{X}_{i_m}, \quad Y_i = \prod_{m=1}^{\ell} \bar{Y}_{i_m}, \quad Z_i = \prod_{m=1}^{\ell} \bar{Z}_{i_m}.$$

The triples (X_i, Y_i, Z_i) satisfy the simultaneous triple product property.

Running example, IV

Growth parameters of this abelian STP family.

$|H_N| = |H|^\ell = (16^3)^{1+\lfloor \log_2(N) \rfloor} = N^{3\log_2(16) + O(1/\log N)}$,
hence $\alpha = 3\log_2(16) = 12$. Also,

$$\begin{aligned} k_N &= \prod_{i=1}^{N} |X_i| = \prod_{i=1}^{N} \prod_{m=1}^{\ell} |\bar{X}_{i_m}| = 15^{N\ell} \\ &= 15^{N\log_2(N) + O(N)} = N^{N\log_2(15) + O(N/\log N)}, \end{aligned}$$

hence $\beta = \log_2(15)$.

Abelian STP algorithms

- The non-abelian group used in the algorithm is a wreath product of H_N with the symmetric group S_N.
- The mapping from $\mathbb{C}[G]$ to a product of matrix algebras, in the Wedderburn thm, is computed by applying $N!$ copies of FFT of H_N, in parallel.
- The three subsets satisfying the triple product property are defined using the sets X_i, Y_i, Z_i.
- The resulting algorithm has running time $O(n^{(\alpha-1)/\beta+o(1)})$.

Algorithm

- **Embedding** (NO ARITHMETIC): Compute the following pair of vectors in $\mathbb{C}[H \wr \mathrm{Sym}_N]$.

$$
\begin{aligned}
a &= \sum_{x \in X} \sum_{y \in Y} A_{xy} \mathrm{e}_{x^{-1}y} \\
b &= \sum_{y \in Y} \sum_{z \in Z} B_{yz} \mathrm{e}_{y^{-1}z}.
\end{aligned}
$$

- **Fourier transform** (ARITHMETIC): Compute the following pair of vectors in $\mathbb{C}[\widehat{H}^N \rtimes \mathrm{Sym}_N]$.

$$\hat{a} = \sum_{\chi \in \widehat{H}^N} \sum_{\sigma \in \mathrm{Sym}_N} \left(\sum_{h \in H^N} \chi(h) a_{\sigma h} \right) \mathrm{e}_{\chi,\sigma}.$$

$$\hat{b} = \sum_{\chi \in \widehat{H}^N} \sum_{\sigma \in \mathrm{Sym}_N} \left(\sum_{h \in H^N} \chi(h) b_{\sigma h} \right) \mathrm{e}_{\chi,\sigma}.$$

Algorithm

- **Assemble matrices** (NO ARITHMETIC): For every $\chi \in \Xi(H^N)$, compute the following pair of matrices A^χ, B^χ, whose rows and columns are indexed by elements of Sym_N.

$$
\begin{aligned}
A^\chi_{\rho\sigma} &= \hat{a}_{\rho\cdot\chi,\sigma\rho^{-1}} \\
B^\chi_{\sigma\tau} &= \hat{b}_{\sigma\cdot\chi,\tau\sigma^{-1}}
\end{aligned}
$$

Algorithm

- **Multiply matrices** (ARITHMETIC): For every $\chi \in \Xi(H^N)$, compute the matrix product $C^\chi = A^\chi B^\chi$ by recursively applying the abelian STP algorithm.

- **Disassemble matrices** (NO ARITHMETIC):
Compute a vector
$\hat{c} = \sum_{\chi,\sigma} \hat{c}_{\chi,\sigma} \mathrm{e}_{\chi,\sigma} \in \mathbb{C}[\widehat{H}^N \rtimes \mathrm{Sym}_N]$ whose components $\hat{c}_{\chi,\sigma}$ are defined as follows.
Given χ, σ, let $\chi_0 \in \Xi(H^N)$ and $\tau \in \mathrm{Sym}_N$ be such that $\chi = \tau \cdot \chi_0$. Let

$$\hat{c}_{\chi,\sigma} := C^{\chi_0}_{\tau,\sigma\tau}.$$

- **Inverse Fourier transform** (ARITHMETIC):
Compute the following vector $c \in \mathbb{C}[H \wr \mathrm{Sym}_N]$.

$$c = \sum_{h \in H^N} \sum_{\sigma \in \mathrm{Sym}_N} \left(\frac{1}{|H|^N} \sum_{\chi \in \widehat{H}^N} \chi(-h) \hat{c}_{\chi,\sigma} \right) \mathrm{e}_{\sigma h}.$$

- **Inverse Fourier transform** (ARITHMETIC): Compute the following vector $c \in \mathbb{C}[H \wr \mathrm{Sym}_N]$.

$$c = \sum_{h \in H^N} \sum_{\sigma \in \mathrm{Sym}_N} \left(\frac{1}{|H|^N} \sum_{\chi \in \widehat{H}^N} \chi(-h) \hat{c}_{\chi,\sigma} \right) \mathrm{e}_{\sigma h}.$$

- **Output** (NO ARITHMETIC): Output the matrix $C = (C_{xz})$ whose entries are given by the formula

$$C_{xz} = c_{x^{-1}z}.$$

Running example, V

In our example with $H = (\mathbb{Z}/16\mathbb{Z})^3$ and $N = 2$, we have $k_N N! = (15^2)(2!) = 450$, so the seven steps above constitute a reduction from 450-by-450 matrix multiplication to $|\Xi(H^2)|$ 2-by-2 matrix multiplication problems. Recall that $|\Xi(H^2)| = 8{,}390{,}656$.

By comparison, the naive reduction from 450-by-450 to 2-by-2 matrix multiplication — by partitioning each matrix into $(225)^2$ square blocks of size 2-by-2 — would require the algorithm to compute $(225)^3 = 11{,}390{,}625$ smaller matrix products.

Using this for recurrence gives running time $O(n^{2.95})$.

Running example, V

Instead, if we use the $N = 2, H = (\mathbb{Z}/16\mathbb{Z})^3$ construction as the basis of an abelian STP family, we may apply the abelian STP algorithm which uses a more sophisticated recursion as the size of the matrices grows to infinity. For example, when $N = 2^\ell$, we have $n = k_N N! = 15^{N\ell}(2^\ell)!$. matrix multiplications. As $N! = O(n^{0.21})$, the resulting running time can be shown to be $O(n^{2.81})$.

Highlight I

Theorem. If $\{(H_N, \Upsilon_N, k_N)\}$ is an abelian STP family with growth parameters (α, β), then the corresponding abelian STP algorithm is stable. It satisfies the error bound

$$\|C_{comp} - C\|_F \le \mu(n)\varepsilon\|A\|_F \cdot \|B\|_F + O(\varepsilon^2),$$

with the Frobenius norm $\|\cdot\|_F$ and the function μ of order

$$\mu(n) = n^{\frac{\alpha+2}{2\beta} + o(1)}.$$

Highlight II

Let ω be the exponent of matrix multiplication.

Theorem. For every $\alpha > 0$ there exists an algorithm for multiplying $n \times n$ matrices that performs $O(n^{\omega+\alpha})$ operations and satisfies the bound

$$\|C_{comp} - C\| \leq \mu(n)\varepsilon\|A\| \cdot \|B\| + O(\varepsilon^2),$$

with $\mu(n) = O(n^c)$ for some constant c that depends on α but not on n.

Highlight II

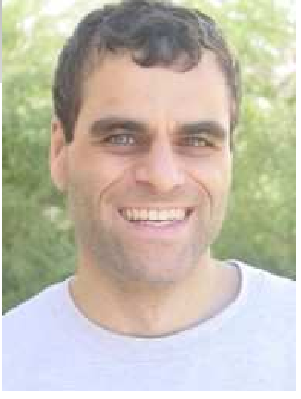
Let ω be the exponent of matrix multiplication.

Theorem. For every $\alpha > 0$ there exists an algorithm for multiplying $n \times n$ matrices that performs $O(n^{\omega+\alpha})$ operations and satisfies the bound

$$\|C_{comp} - C\| \leq \mu(n)\varepsilon\|A\| \cdot \|B\| + O(\varepsilon^2),$$

with $\mu(n) = O(n^c)$ for some constant c that depends on α but not on n.

Remark: It is an open question whether a group-theoretic algorithm can achieve $O(n^{\omega+\alpha})$ for arbitrarily small α.



Ran Raz

On the complexity of matrix product

SIAM J. Computing [2003].

Theorem. The exponent of matrix multiplication is achievable by bilinear noncommutative algorithms. More precisely, for every arithmetic circuit of size S which computes the product of two matrices A, B over a field with characteristic zero, there is a bilinear circuit of size $O(S)$ that also computes the product of A and B.

Tradeoff between n and ε

All our bounds are of the form

$$\|C_{comp} - C\| \le \mu(n)\varepsilon\|A\| \cdot \|B\| + O(\varepsilon^2), \quad (*)$$

$\mu(n) = O(n^c)$ for some constant $c \ge 1$.

Tradeoff between n and ε

All our bounds are of the form

$$\|C_{comp} - C\| \le \mu(n)\varepsilon\|A\| \cdot \|B\| + O(\varepsilon^2), \quad (*)$$

$\mu(n) = O(n^c)$ for some constant $c \ge 1$.

As $\mu(n)\varepsilon \ll 1$, the number b of bits to represent fractional part of floating point numbers satisfies $b = \log_2(1/\varepsilon) \ge \log_2 n$.

Tradeoff between n and ε

All our bounds are of the form

$$\|C_{comp} - C\| \le \mu(n)\varepsilon\|A\| \cdot \|B\| + O(\varepsilon^2), \quad (*)$$

$\mu(n) = O(n^c)$ for some constant $c \ge 1$.

As $\mu(n)\varepsilon \ll 1$, the number b of bits to represent fractional part of floating point numbers satisfies $b = \log_2(1/\varepsilon) \ge \log_2 n$.

Multiplying the number of bits by a factor f raises the cost of an algorithm by $O(f^{1+o(1)})$ using Schönhage-Strassen [1971].

Tradeoff between n and ε

All our bounds are of the form

$$\|C_{comp} - C\| \le \mu(n)\varepsilon\|A\| \cdot \|B\| + O(\varepsilon^2), \quad (*)$$

$\mu(n) = O(n^c)$ for some constant $c \ge 1$.

As $\mu(n)\varepsilon \ll 1$, the number b of bits to represent fractional part of floating point numbers satisfies $b = \log_2(1/\varepsilon) \ge \log_2 n$.

Multiplying the number of bits by a factor f raises the cost of an algorithm by $O(f^{1+o(1)})$ using Schönhage-Strassen [1971].

All algorithms satisfying $(*)$ are in fact $O(\cdot)$-equivalent.

Advertisement

[math/0603207] Fast matrix multiplication is stable http://arxiv.org/abs/math.NA/0603207

arXiv.org > math > math/0603207

Search for
(Help | Advanced search)
All papers Go!

Mathematics, abstract
math.NA/0603207

From: Olga Holtz [view email]
Date: Thu, 9 Mar 2006 04:34:36 GMT (18kb)

Fast matrix multiplication is stable

Authors: James Demmel, Ioana Dumitriu, Olga Holtz, Robert Kleinberg
Comments: 14 pages
Subj-class: Numerical Analysis; Group Theory; Computational Complexity; Data Structures and Algorithms
MSC-class: 65Y20, 65F30, 65G50, 68Q17, 68W40, 20C05, 20K01, 16S34, 43A30, 65T50

We perform forward error analysis for a large class of recursive matrix multiplication algorithms in the spirit of [D. Bini and G. Lotti, Stability of fast algorithms for matrix multiplication, Numer. Math. 36 (1980), 63--72]. As a consequence of our analysis, we show that the exponent of matrix multiplication can be achieved by numerically stable algorithms. We also show that new group-theoretic algorithms proposed in [H. Cohn, and C. Umans, A group-theoretic approach to fast matrix multiplication, FOCS 2003, 438--449] and [H. Cohn, R. Kleinberg, B. Szegedy and C. Umans, Group-theoretic algorithms for matrix multiplication, FOCS 2005, 379--388] are all included in the class of algorithms to which our analysis applies, and are therefore numerically stable. We perform detailed error analysis for three specific fast group-theoretic algorithms.

Full-text: PostScript, PDF, or Other formats

References and citations for this submission:
CiteBase (autonomous citation navigation and analysis)

Which authors of this paper are endorsers?

Links to: arXiv, form interface, /find, math, /0603, /abs, previous, next, help (Access key information)
