Matrix Multiplication, a Little Faster

Elaye Karstadt and Oded Schwartz

Department of Computer Science
Hebrew University of Jerusalem

SIAM AN17
MS on Communication-Avoiding Algorithms
July 13th, 2017

Research is supported by grants 1878/14, and 1901/14 from the Israel Science Foundation (founded by the Israel Academy of Sciences and Humanities) and grant 3-10891 from the Ministry of Science and Technology, Israel. Research is also supported by the Einstein Foundation and the Minerva Foundation. This work was supported by the Peta-Cloud industryacademia consortium. This research was supported by a grant from the United States-Israel Bi-national Science Foundation (BSF), Jerusalem, Israel. This work was supported by the HUJI Cyber Security Research Center in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office. We acknowledge PRACE for awarding us access to Hazel Hen at GCS@HLRS, Germany

Outline

Outline

Goals

- Improve the performance of fast matrix multiplication algorithms:
  1. Reduce the arithmetic complexity by a constant factor.
  2. Reduce the communication costs (within memory hierarchy) by a constant factor.

Strassen's Algorithm

- Strassen's algorithm (1969) was the first sub-cubic matrix multiplication algorithm. Achieving arithmetic complexity of $O\left(n^{\log_2 7}\right)$.
- Performs 2×2 matrix multiplication using 7 scalar multiplications and 18 additions instead of 8 scalar multiplications and 4 additions.
- Winograd (1971) improved the leading coefficient of its complexity from 7 to 6 by reducing the number of additions from 18 to 15.

Lower Bounds on 2×2 matrix multiplication

Theorem (Probert, 1976)

15 additions are necessary to compute 2×2 matrix multiplication using 7 scalar multiplications.

- Thus Strassen-Winograd's algorithm is optimal for 2×2 base case.

Theorem (Karstadt and Schwartz, 2017)

There exists a Strassen-like algorithm with 2x2 base and 7 multiplications, that uses only 12 additions

- Contradiction?

Lower Bounds on 2×2 matrix multiplication

Theorem (Probert, 1976)

15 additions are necessary to compute 2×2 matrix multiplication using 7 scalar multiplications.

- Thus Strassen-Winograd's algorithm is optimal for 2×2 base case.

Theorem (Karstadt and Schwartz, 2017)

There exists a Strassen-like algorithm with 2x2 base and 7 multiplications, that uses only 12 additions

- Contradiction?

Alternative Basis Strassen

Theorem (Karstadt and Schwartz, 2017)

There exists a Strassen-like algorithm with 2x2 base and 7 multiplications, that uses only 12 additions

- Our algorithm seems to contradict Probert's lower bound. However, his bound implicitly assumes that input and output are represented in the standard basis. We utilize this. Namely, we reduce the number of additions by changing the basis of the input and output matrices.
- Bodrato (2010) used a similar method of intermediate representation of 2×2 matrices for repeated squaring and for chain matrix multiplication.

Basis invariant lower bounds

Theorem (Karstadt and Schwartz, 2017)

There exists a Strassen-like algorithm with 2x2 base and 7 multiplications, that uses only 12 additions

- We extend Probert's lower bound to account for alternative bases:

Theorem (Karstadt and Schwartz, 2017)

Irrespective of input/output bases, a Strassen-like algorithm with 2x2 base case and 7 multiplications requires at least 12 additions

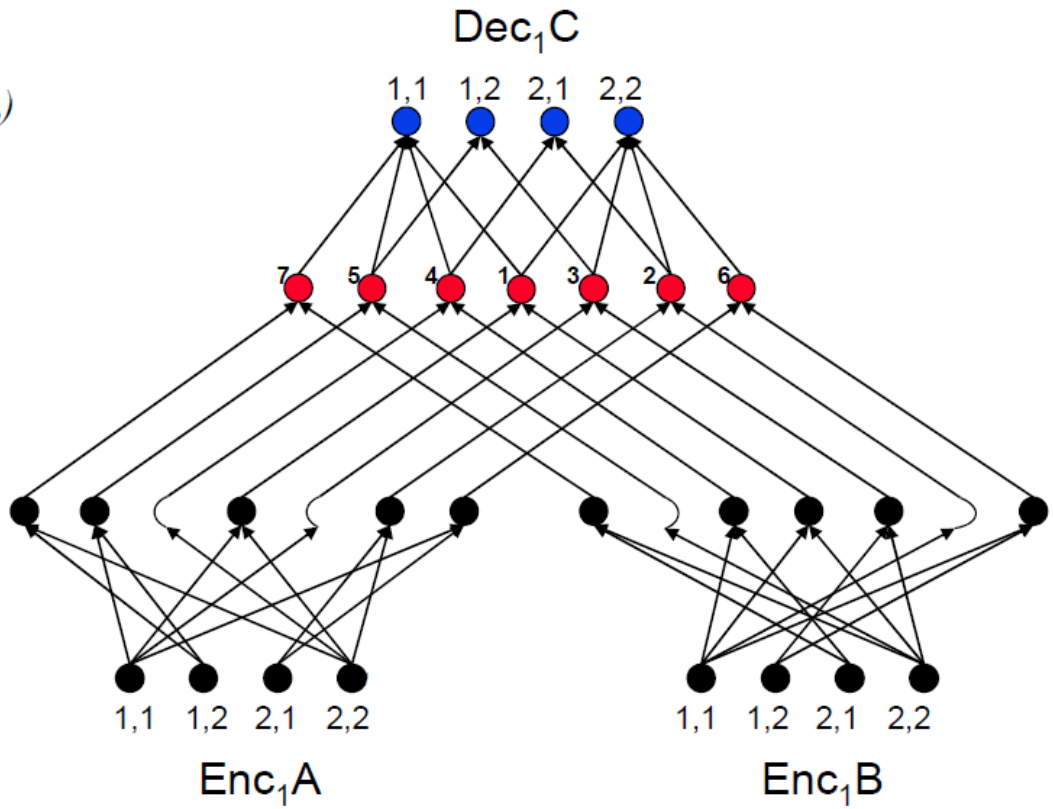
- This proves that our new algorithm is optimal.

Outline

Strassen's Algorithm

$$M_1 = (A_{11} + A_{22}) \cdot (B_{11} + B_{22})$$
$$M_2 = (A_{21} + A_{22}) \cdot B_{11}$$
$$M_3 = A_{11} \cdot (B_{12} - B_{22})$$
$$M_4 = A_{22} \cdot (B_{21} - B_{11})$$
$$M_5 = (A_{11} + A_{12}) \cdot B_{22}$$
$$M_6 = (A_{21} - A_{11}) \cdot (B_{11} + B_{12})$$
$$M_7 = (A_{12} - A_{22}) \cdot (B_{21} + B_{22})$$

$$C_{11} = M_1 + M_4 - M_5 + M_7$$
$$C_{12} = M_3 + M_5$$
$$C_{21} = M_2 + M_4$$
$$C_{22} = M_1 - M_2 + M_3 + M_6$$



We refer to U, V as the encoding matrices (of A and B respectively) and W as the decoding matrix.

Encoding and Decoding

- A matrix multiplication algorithm can be described by three matrices $\langle U, V, W \rangle$ such that:

$$\overrightarrow{A \cdot B} = W^T \left(U \cdot \vec{A} \odot V \cdot \vec{B} \right)$$

Where $\vec{A}$ is a row-order vectorization of A, $\odot$ is element-wise multiplication of vectors and $\cdot$ is a matrix-vector multiplication.

Strassen's Algorithm

$$\langle U, V, W \rangle =$$
$$\left\langle \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \right\rangle$$

Fact

The number of multiplications performed by a bi-linear algorithm is the number of rows of the encoding/decoding matrices.

Fact

The number of additions performed by a bi-linear algorithm is

$$nnz(U) - rows(U) + nnz(V) - rows(V) + nnz(W) - cols(W)$$

Encoding/Decoding of Matrix Multiplication

Fact (Triple Product Condition)

Let U, V, W be matrices of sizes $t \times m \cdot n$, $t \times n \cdot k$, $t \times m \cdot k$ respectively. Then $\langle U, V, W \rangle$ are encoding/decoding matrices of an $\langle m, n, k; t \rangle$-algorithm iff:

$$\forall i_1, k_1 \in [m], i_2, j_1 \in [n], j_2, k_2 \in [k]$$

$$\tau_{(i_1,i_2),(j_1,j_2),(k_1,k_2)} = \sum_{r=1}^{t} U_{r,(i_1,i_2)} V_{r,(j_1,j_2)} W_{r,(k_1,k_2)} = \delta_{i_1,k_1} \delta_{i_2,j_1} \delta_{j_2,k_2}$$

Where $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise.

Matrix Multiplication Algorithms

Definition (Matrix Multiplication algorithms)

We denote $n_0 \times m_0$ by $m_0 \times k_0$ matrix multiplication using t_0 products by $\langle m_0, n_0, k_0; t_0 \rangle$-algorithm.

- Thus Strassen's and Winograd's are $\langle 2, 2, 2; 7 \rangle$-algorithms.
- When applied recursively, an $\langle n_0, n_0, n_0; t_0 \rangle$-algorithm has arithmetic complexity of $O\left(n^{\log_{n_0} t_0}\right)$.
- Given an $\langle m_0, n_0, k_0; t_0 \rangle$-algorithm, it is easy to derive an $\left\langle m_0 n_0 k_0, m_0 n_0 k_0, m_0 n_0 k_0; t_0^3 \right\rangle$-algorithm with arithmetic complexity $O(n^{\omega_0})$ for $\omega_0 = \log_{m_0 n_0 k_0} t_0^3$.

Leading Coefficient

- How does the number of additions affect the leading coefficient?

Lemma

Suppose an $\langle n_0, n_0, n_0; t_0 \rangle$ – algorithm performs ℓ linear operations (addition/subtraction/scalar multiplication) at the base case. Its arithmetic complexity is:

$$F(n) = \left(1 + \frac{\ell}{t_0 - n_0^2}\right) n^{\log_{n_0} t_0} - \frac{\ell}{t_0 - n_0^2} n^2$$

Outline

Outline

Alternative-Basis Matrix Multiplication Algorithms

Input: $A \in R^{n\times m}$, $B \in R^{m\times k}$
Output: $C \in R^{n\times k}$ such that $C = A \cdot B$

1: **function** $ABS(A, B)$
2: $\quad \tilde{A} = \phi(A)$ ▷ Basis transformation
3: $\quad \tilde{B} = \psi(B)$ ▷ Basis transformation
4: $\quad \tilde{C} = ALG(\tilde{A}, \tilde{B})$ ▷ an $\langle n, m, k; t\rangle_{\phi,\psi,\upsilon}$-algorithm
5: $\quad C = \upsilon^{-1}(\tilde{C})$ ▷ Basis transformation
6: $\quad$ **return** C

Definition (Alternative Basis Matrix Multiplication algorithms)

We denote $n \times m$ by $m \times k$ matrix multiplication algorithm that uses t products and basis transformations ϕ, ψ, υ by $\langle n, m, k; t\rangle_{\phi,\psi,\upsilon}$-algorithm.
When $n = m = k$ and $\phi = \psi = \upsilon$, we write $\langle n, n, n; t\rangle_{\phi}$-algorithm.

Matrix Multiplication with Basis Transformation

Lemma (Karstadt and Schwartz, 2017)

Let ϕ, ψ, υ be invertible linear transformations. The following are equivalent:

1. *The matrices $\langle U, V, W\rangle$ are encoding/decoding matrices of an $\langle m, n, k; t\rangle$-algorithm.*
2. *$\left\langle U\phi^{-1}, V\psi^{-1}, W\upsilon^{T}\right\rangle$ are encoding/decoding matrices of an $\langle m, n, k; t\rangle_{\phi,\psi,\upsilon}$-algorithm.*

Proof.

$$\overrightarrow{A\cdot B} = W^{T}\left(U\cdot\vec{A}\odot V\cdot\vec{B}\right)$$

$$\Updownarrow$$

$$\upsilon\left(\overrightarrow{A\cdot B}\right) = \left(W\upsilon^{T}\right)^{T}\left(\left(U\phi^{-1}\right)\cdot\left(\phi\vec{A}\right)\odot\left(V\psi^{-1}\right)\cdot\left(\psi\vec{B}\right)\right)$$

□

Improving the Strassen-Winograd Algorithm

To Improve on Strassen-Winograd's algorithm's 15 additions, we need an alternative basis $\langle 2,2,2;7\rangle_{\phi,\psi,\upsilon}$-algorithm, namely three matrices $\langle U,V,W\rangle$ and basis transformation ϕ,ψ,υ such that:

1. U,V,W have fewer non-zeros than those of Strassen-Winograd.
2. The basis transformations ϕ,ψ, and υ^{-1} can be computed fast.

Outline

Alternative Basis

Define an invertible linear transformation $\psi : R^{n^2} \to R^{n^2}$ (where $n = 2^m$) recursively. Let $\psi_1 : R^4 \to R^4$ (with inverse ψ_1^{-1}):

$$\psi_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} \qquad \psi_1^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & -1 & 1 & 1 \end{pmatrix}$$

For convenience, when applying ψ_1 to matrices, we omit the vectorization and write:

$$\psi_1(A) = \psi_1 \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} = \begin{pmatrix} A_{1,1} & A_{1,2} - A_{2,1} + A_{2,2} \\ A_{21} - A_{2,2} & A_{1,2} + A_{2,2} \end{pmatrix}$$

Where $A_{i,j}$ can be ring elements or submatrices.

Alternative Basis Transformation

Let for $n = 2^{k+1}$ we define $\psi_{k+1} : R^{n^2} \to R^{n^2}$ be:

$$\psi_{k+1}(A) = \psi_1 \begin{pmatrix} \psi_k(A_{1,1}) & \psi_k(A_{1,2}) \\ \psi_k(A_{2,1}) & \psi_k(A_{2,2}) \end{pmatrix} \qquad \psi_{k+1}^{-1}(A) = \psi_1^{-1} \begin{pmatrix} \psi_k^{-1}(A_{1,1}) & \psi_k^{-1}(A_{1,2}) \\ \psi_k^{-1}(A_{2,1}) & \psi_k^{-1}(A_{2,2}) \end{pmatrix}$$

For convenience, we omit the subscript of ψ when obvious from context.

Lemma

The arithmetic complexity of computing $\psi(A)$ is

$$F_\psi(n) = n^2 \log_2 n$$

The same holds for computing $\psi^{-1}(A)$.

Recursive Alternative-Basis Strassen

- Given $\psi^{-1}(A)$, $\psi^{-1}(B)$ we use the following bi-linear recursive algorithm to compute $\psi(A \cdot B)$:

$$\langle U_{opt}, V_{opt}, W_{opt} \rangle =$$

$$\left\langle \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 1 & -1 & 0 \\ -1 & 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & -1 & 0 & -1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \right\rangle$$

- Using the formula from before we know the number of additions is:

$$\overbrace{nnz(U)}^{10} - \overbrace{rows(U)}^{7} + \overbrace{nnz(V)}^{10} - \overbrace{rows(V)}^{7} + \overbrace{nnz(W)}^{10} - \overbrace{cols(W)}^{4} = 12$$

Correctness of Encoding/Decoding Matrices

$$\left\langle U_{opt} \cdot \psi_{opt}, V_{opt} \cdot \psi_{opt}, W_{opt} \cdot \psi_{opt}^{-T} \right\rangle =$$

$$\left\langle \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 1 & -1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 1 & -1 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 1 & -1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 1 & -1 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 1 & -1 & -1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & -1 & -1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix} \right\rangle$$

It is easy to verify that $\left\langle U_{ABS} \cdot \psi_1, V_{ABS} \cdot \psi_1, W_{ABS} \cdot \psi_1^{-T} \right\rangle$ satisfies the triple product condition.

Computing Multiplication in Alternative Basis

Lemma (Main Lemma)

The arithmetic complexity of our $\langle 2,2,2;7\rangle_{\psi_{opt}}$-algorithm is:

$$F(n) = 5n^{\log_2 7} - 4n^2 + 3n^2 \log_2 n$$

- Recently, Cenk and Hasan (2017) showed a clever way to apply Strassen-Winograd's algorithm directly to $n \times n$ matrices by forsaking the uniform divide-and-conquer pattern of Strassen-like algorithms. Their arithmetic complexity is $5n^{\log_2 7} + 0.5 \cdot n^{\log_2 6} + 2n^{\log_2 5} - 6.5n^2$. However, this comes at the cost of increased communication costs and memory footprint.

Computing Multiplication in Alternative Basis

Lemma (Main Lemma)

The arithmetic complexity of our $\langle 2,2,2;7\rangle_{\psi_{opt}}$-algorithm is:

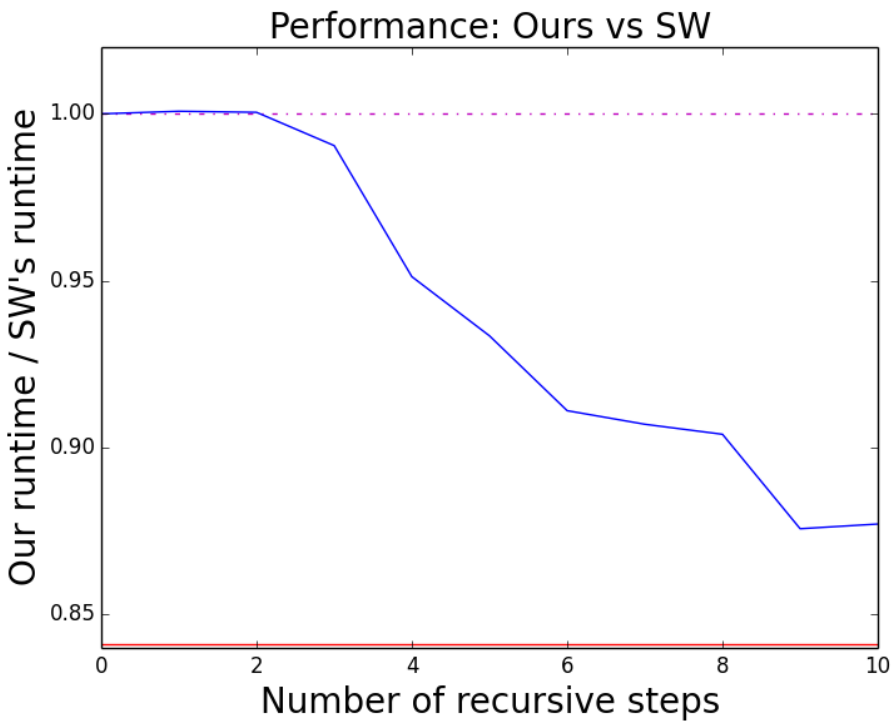
$$F(n) = 5n^{\log_2 7} - 4n^2 + 3n^2 \log_2 n$$

- Recently, Cenk and Hasan (2017) showed a clever way to apply Strassen-Winograd's algorithm directly to $n \times n$ matrices by forsaking the uniform divide-and-conquer pattern of Strassen-like algorithms. Their arithmetic complexity is $5n^{\log_2 7} + 0.5 \cdot n^{\log_2 6} + 2n^{\log_2 5} - 6.5n^2$. However, this comes at the cost of increased communication costs and memory footprint.

$\langle 2,2,2;7\rangle$ – *Algorithms*

Algorithm	Additions	Arithmetic cost	I/O-complexity
Strassen (1969)	18	$7n^{\log_2 7} - 6n^2$	$6 \cdot \left(\frac{\sqrt{3} \cdot n}{\sqrt{M}}\right)^{\log_2 7} \cdot M - 18n^2 + 3M$
Winograd (1971)	15	$6n^{\log_2 7} - 5n^2$	$5 \cdot \left(\frac{\sqrt{3} \cdot n}{\sqrt{M}}\right)^{\log_2 7} \cdot M - 15n^2 + 3M$
Ours	12	$5n^{\log_2 7} - 4n^2 + 3n^2 \log_2 n$	$4 \cdot \left(\frac{\sqrt{3} \cdot n}{\sqrt{M}}\right)^{\log_2 7} \cdot M - 12n^2 + 5M + 3n^2 \cdot \log_2 \left(\cdot \frac{\sqrt{2} \cdot n}{\sqrt{M}}\right)$

Strassen-Winograd vs. Our $\langle 2,2,2;7\rangle_\psi$-algorithm



$N = 32768$

All experiments were conducted on a single compute node of HLRS's Hazel Hen, with two 12-core (24 threads) Intel Xeon CPU E5-2680 v3 and 128GB of memory.

Outline

Improved Alternative-Basis Variants

Algorithm	Linear Operations	Leading Coefficient	Alternative-Basis Linear Operations	Improved leading coefficient	Reduction Factor
$\langle 3,2,3;15\rangle$ [Ballard and Benson, 2015]	64	15.06	52	7.94	47.3%
$\langle 2,3,4;20\rangle$ [Ballard and Benson, 2015]	78	9.96	58	7.46	25.6%
$\langle 3,3,3;23\rangle$ [Laderman, 1976]	87	8.91	75	6.57	26.3%
$\langle 6,3,3;40\rangle$ [Smirnov, 2013]	1246	55.63	202	9.39	83.2%

Finding Optimal Alternative-Basis Variants

Problem

Matrix Sparsification (MS):
Let U be an $m \times n$ matrix of full rank, find an invertible matrix A s.t.

$$A = \underset{A \in GL_n}{\texttt{argmin}}(nnz(UA))$$

- Finding basis transformations for a Strassen-like algorithm consists of three independent MS problems. Unfortunately, MS is not only NP-Hard [McCormick, 1983] to solve, but also NP-Hard to approximate[1] [Gottlieb and Neylon, 2010]
- However, since we are interested in small base cases, sparsifying basis transformations can be found manually, or by computer aided search heuristics.

[1] Over $\mathbb{Q}$, assuming NP does not admit quasi-polynomial time deterministic algorithms} to within a factor of $2^{\log^{.5-o(1)} n}$

Outline

Conclusion

We obtained a new Strassen-like algorithm, with 2×2 base case, improving on Strassen-Winograd by factors of $5/6$ (arithmetic complexity) and $4/5$ (I/O-Complexity).

- This was believed to be impossible due to a lower bound of Probert.
- We extend Probert's lower bound to be applicable to our new alternative-basis method, proving our algorithm to be optimal.
- Our new algorithm is faster in practice even on reasonably small matrices.
- We applied our new alternative-basis method to improve other Strassen-like algorithm.

Thank You

Thank You

Questions?

Permutations of Encoding/Decoding Matrices

Definition

Denote the permutation matrix which swaps row-order for column-order of vectorization of an $I \times J$ matrix by $P_{I \times J}$.

Lemma

Let $\langle U, V, W \rangle$ be the encoding/decoding matrices of a $\langle m, k, n; t \rangle$-algorithm:

- *$\langle VP_{n \times k}, UP_{m \times k}, WP_{m \times n} \rangle$ are encoding/decoding matrices of a $\langle n, k, m; t \rangle$-algorithm.*
- *$\langle WP_{m \times n}, U, VP_{n \times k} \rangle$ are encoding/decoding matrices of a $\langle n, m, k; t \rangle$-algorithm.*
- *$\langle W, VP_{n \times k}, U \rangle$ are encoding/decoding matrices of a $\langle m, n, k; t \rangle$-algorithm.*
- *$\langle V, WP_{m \times n}, UP_{m \times k} \rangle$ are encoding/decoding matrices of a $\langle k, n, m; t \rangle$-algorithm.*
- *$\langle UP_{m \times k}, W, V \rangle$ are encoding/decoding matrices of a $\langle k, m, n; t \rangle$-algorithm.*

Basis Transformation Complexity

Lemma

The IO-complexity of computing $\psi(A)$ is

$$IO_{\psi}(n, M) \leq 3n^2 \log_{n_0}\left(\sqrt{2}\frac{n}{\sqrt{M}}\right) + 2M$$

Proof.

The basis transformation has 4 recursive calls and requires 4 additions at each step. With base case occuring when the problem fits entirely in memory, namely $2n^2 \leq M$. Each addition requires at most 3 data transfers (one of each input and one for writing the output). Yielding the recurrence.

$$IO_{ALG}(n, M) \leq \begin{cases} 4 \cdot IO_{\psi}\left(\frac{n}{n_0}, M\right) + 3 \cdot 4 \cdot \left(\frac{n}{n_0}\right)^2 & 2n^2 > M \\ 2M & \text{otherwise} \end{cases}$$

□

The same holds for computing $\psi^{-1}(A)$.

Computing Multiplication in Alternative Basis

Lemma

The I/O-complexity of our $\langle 2,2,2;7\rangle_{\psi_{OPT}}$*-algorithm is:*

$$IO(n,M) \leq \frac{4}{3}\left(M\left(\sqrt{3}\cdot\frac{n}{\sqrt{M}}\right)^{\log_2 7} - 3n^2\right) + 3M$$

Proof.

The recursive bi-linear part has 7 *recursive calls and requires* 12 *additions at each step. With base base occuring when the problem fits entirely in memory, namely* $3n^2 \leq M$*. Each addition requires at most 3 data transfers (one of each input and one for writing the output). Yielding the recurrence*

$$IO_{ALG}(n,M) \leq \begin{cases} 7\cdot IO_{\psi}\left(\frac{n}{n_0},M\right) + 3\cdot 12\cdot\left(\frac{n}{n_0}\right)^2 & 3n^2 > M \\ 3M & \text{otherwise} \end{cases}$$

□

Constrictions on Multiplicands

Lemma

[Hopcroft and Kerr 1971]If an algorithm for 2×2 matrix multiplication has k left (right) hand multiplicands of forms $A_{1,1}$, $(A_{1,2} + A_{2,1})$ and $(A_{1,1} + A_{1,2} + A_{2,1})$ (where additions are done mod 2) then it requires at least $6 + k$ multiplications.

Lemma

[Hopcroft and Kerr 1971]If an algorithm for 2×2 matrix multiplication has k left (right) hand multiplicands of one of the following groups (where additions are done mod 2) then it requires at least $\lceil 6 + \frac{k}{2} \rceil$ multiplications.

1. $A_{1,1}, A_{1,2}, (A_{1,1} + A_{1,2})$
2. $A_{2,1}, A_{2,2}, (A_{2,1} + A_{2,2})$

Constrictions on Multiplicands Cont.

Lemma

[Hopcroft and Kerr 1971] If an algorithm for 2×2 matrix multiplication has k left (right) hand multiplicands of one of the following groups (where additions are done mod 2) then it requires at least $6 + k$ multiplications.

1. $(A_{1,1} + A_{2,1}), (A_{1,2} + A_{2,1} + A_{2,2}), (A_{1,1} + A_{1,2} + A_{2,2})$
2. $(A_{1,1} + A_{1,2}), (A_{1,2} + A_{2,1} + A_{2,2}), (A_{1,1} + A_{2,2} + A_{2,2})$
3. $(A_{1,1} + A_{1,2} + A_{2,1} + A_{2,2}), (A_{1,2} + A_{2,1}), (A_{1,1} + A_{2,2})$
4. $A_{2,1}, (A_{1,1} + A_{2,2}), (A_{1,1} + A_{2,1} + A_{2,2})$
5. $(A_{2,1} + A_{2,2}), (A_{1,1} + A_{1,2} + A_{2,2}), (A_{1,1} + A_{1,2} + A_{2,1})$
6. $A_{1,2}, (A_{1,1} + A_{2,2}), (A_{1,1} + A_{1,2} + A_{2,2})$
7. $(A_{1,1} + A_{2,2}), (A_{1,1} + A_{2,1} + A_{2,2}), (A_{1,1} + A_{1,2} + A_{2,1})$
8. $A_{2,2}, (A_{1,2} + A_{2,1}), (A_{1,2} + A_{2,1} + A_{2,2})$

Lower Bound on Additive Complexity

- Using the previous results, Proberts used a counting argument to show that any $\langle 2,2,2;7\rangle$ algorithm requires at least 15 additions.

Lemma

[Probert, 1976] Each encoding of a $\langle 2,2,2;7\rangle$-algorithm requires at least 4 additions.

Lemma

[Probert, 1976] The decoding of a $\langle 2,2,2;7\rangle$-algorithm requires at least 7 additions.

- From this, Probert's theorem follows.

Basis Invariant Lower Bound on Additive Complexity

Corollary

Any 2×2 matrix multiplication algorithm where a left hand (or right hand) multiplicand appears at least twice (modulu 2) requires 8 or more multiplications.

Fact

A simple counting argument shows that any 7×4 binary matrix with less than 10 non-zero entries has a duplicate row (modulu 2) or a zeroed out row.

Lemma

Irrespective of basis transformations ϕ, ψ, υ, the encoding matrices U, V, of a $\langle 2,2,2;7 \rangle$-algorithm have no duplicate rows or zeroed out rows.

Basis Invariant Lower Bound on Additive Complexity

Lemma

Irrespective of basis transformations ϕ, ψ, υ, the encoding matrices U, V, of a $\langle 2,2,2;7\rangle_{\phi,\psi,\upsilon}$-algorithm have at least 10 non-zero entries.

Lemma

Irrespective of basis transformations ϕ, ψ, υ, the decoding matrix W of an $\langle 2,2,2;7\rangle_{\phi,\psi,\upsilon}$-algorithm has at least 10 non-zero entries.

- The basis invariant lower bound is an immediate Corollary of these Lemmas.

Austin R Benson and Grey Ballard. A framework for practical parallel fast matrix multiplication. *ACM SIGPLAN Notices*, 50(8):42–53, 2015.

Marco Bodrato. A strassen-like matrix multiplication suited for squaring and higher power computation. In *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*, pages 273–280. ACM, 2010.

Lee-Ad Gottlieb and Tyler Neylon. Matrix sparsification and the sparse null space problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 205–218. Springer, 2010.

John E Hopcroft and Leslie R Kerr. On minimizing the number of multiplications necessary for matrix multiplication. *SIAM Journal on Applied Mathematics*, 20(1):30–36, 1971.

Julian D Laderman. A noncommutative algorithm for multiplying 3×3 matrices using 23 multiplications. *Bulletin of the American Mathematical Society*, 82(1):126–128, 1976.

S Thomas McCormick. A combinatorial approach to some sparse matrix problems. Technical report, DTIC Document, 1983.

Robert L Probert. On the additive complexity of matrix multiplication. *SIAM Journal on Computing*, 5(2):187–203, 1976.

AV Smirnov. The bilinear complexity and practical algorithms for matrix multiplication. *Computational Mathematics and Mathematical Physics*, 53(12):1781–1795, 2013.

Volker Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.

Shmuel Winograd. On multiplication of 2$\times$ 2 matrices. *Linear algebra and its applications*, 4(4):381–388, 1971.