Strassen-like algorithms for symmetric tensor contractions

Edgar Solomonik

University of Illinois at Urbana-Champaign

Scientfic and Statistical Computing Seminar
University of Chicago

April 13, 2017

Outline

Terminology

A tensor $\boldsymbol{T} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ has

- order d (i.e. d modes / indices)
- dimensions n_1-by-$\cdots$-by-n_d (in this talk, usually each $n_i = n$)
- elements $\boldsymbol{T}_{\boldsymbol{i}_1 \ldots \boldsymbol{i}_d} = \boldsymbol{T}_{\boldsymbol{i}}$ where $\boldsymbol{i} \in \{1, \ldots, n\}^d$

We say a tensor is symmetric if for any $j, k \in \{1, \ldots, n\}$

$$\boldsymbol{T}_{\boldsymbol{i}_1 \ldots \boldsymbol{i}_j \ldots \boldsymbol{i}_k \ldots \boldsymbol{i}_d} = \boldsymbol{T}_{\boldsymbol{i}_1 \ldots \boldsymbol{i}_k \ldots \boldsymbol{i}_j \ldots \boldsymbol{i}_d}$$

A tensor is antisymmetric (skew-symmetric) if for any $j, k \in \{1, \ldots, n\}$

$$\boldsymbol{T}_{\boldsymbol{i}_1 \ldots \boldsymbol{i}_j \ldots \boldsymbol{i}_k \ldots \boldsymbol{i}_d} = (-1) \boldsymbol{T}_{\boldsymbol{i}_1 \ldots \boldsymbol{i}_k \ldots \boldsymbol{i}_j \ldots \boldsymbol{i}_d}$$

A tensor is partially-symmetric if such index interchanges are restricted to be within subsets of $\{1, \ldots, n\}$, e.g.

$$\boldsymbol{T}^{ij}_{kl} = \boldsymbol{T}^{ji}_{kl} = \boldsymbol{T}^{ji}_{lk} = \boldsymbol{T}^{ij}_{lk}$$

Tensor contractions

We work with contractions of tensors

- $\boldsymbol{A}$ of order $s + v$, and
- $\boldsymbol{B}$ of order $v + t$ into
- $\boldsymbol{C}$ of order $s + t$, defined as

$$\boldsymbol{C}_{\boldsymbol{ij}} = \sum_{\boldsymbol{k} \in \{1,\ldots,n\}^v} \boldsymbol{A}_{\boldsymbol{ik}} \boldsymbol{B}_{\boldsymbol{kj}}$$

- requires $O(\underbrace{s + t + v}_{\omega})$ multiplications and additions
- assumes an index ordering, but does not lose generality
- works with any symmetries of $\boldsymbol{A}$ and $\boldsymbol{B}$
- is extensible to symmetries of $\boldsymbol{C}$ via symmetrization (sum all permutations of modes in $\boldsymbol{C}$, denoted $[\boldsymbol{C}]_{\boldsymbol{ij}}$)
- generalizes simple matrix operations, e.g.

$$\underbrace{(s, t, v) = (1, 0, 1)}_{\text{matrix-vector product}}, \quad \underbrace{(s, t, v) = (1, 1, 0)}_{\text{vector outer product}}, \quad \underbrace{(s, t, v) = (1, 1, 1)}_{\text{matrix-matrix product}}$$

Applications of symmetric tensor contractions

Symmetric and Hermitian matrix operations are part of the BLAS

- matrix-vector products: `symv` (`symm`), `hemv`, (`hemm`)
- symmetrized outer product: `syr2` (`syr2k`), `her2`, (`her2k`)
- these operations dominate symmetric/Hermitian diagonalization

Hankel matrices are order $2\log_2(n)$ partially-symmetric tensors

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H_{11}} & \boldsymbol{H_{21}^{\mathsf{T}}} \\ \boldsymbol{H_{21}} & \boldsymbol{H_{22}} \end{bmatrix}$$

where $\boldsymbol{H_{11}}, \boldsymbol{H_{21}}, \boldsymbol{H_{22}}$ are also Hankel.

In general, partially-symmetric tensors are nested symmetric tensors

- a nonsymmetric matrix is a vector of vectors
- $\boldsymbol{T}_{kl}^{ij} = \boldsymbol{T}_{kl}^{ji} = \boldsymbol{T}_{lk}^{ji} = \boldsymbol{T}_{lk}^{ij}$ is a symmetric matrix of symmetric matrices

Applications of partially-symmetric tensor contractions

High-accuracy methods in computational quantum chemistry

- solve the multi-electron Schrödinger equation $\boldsymbol{H}|\boldsymbol{\Psi}\rangle = E|\boldsymbol{\Psi}\rangle$, where $\boldsymbol{H}$ is a linear operator, but $\boldsymbol{\Psi}$ is a function of *all* electrons
- use wavefunction ansatze like $\boldsymbol{\Psi} \approx \boldsymbol{\Psi}^{(k)} = e^{\boldsymbol{T}^{(k)}}|\boldsymbol{\Psi}^{(k-1)}\rangle$ where $\boldsymbol{\Psi}^{(0)}$ is a determinant function and $\boldsymbol{T}^{(k)}$ is an order $2k$ tensor, acting as a multilinear excitation operator on the electrons
- are most commonly versions of coupled-cluster methods which use the above ansatze for $k \in \{2, 3, 4\}$ (CCSD, CCSDT, CCSDTQ)
- solve iteratively for $\boldsymbol{T}^{(k)}$, where each iteration has cost $O(n^{2k+2})$, dominated by contractions of partially antisymmetric tensors
- for example, a dominant contraction in CCSD ($k = 2$) is

$$\boldsymbol{Z}^{a\bar{k}}_{i\bar{c}} = \sum_{b=1}^{n}\sum_{j=1}^{n} \boldsymbol{T}^{ab}_{ij} \cdot \boldsymbol{V}^{j\bar{k}}_{b\bar{c}}$$

where $\boldsymbol{T}^{ab}_{ij} = -\boldsymbol{T}^{ba}_{ij} = \boldsymbol{T}^{ba}_{ji} = -\boldsymbol{T}^{ab}_{ji}$. We'll show an algorithm that requires n^6 rather than $2n^6$ operations.

Symmetric matrix times vector

- Let $\boldsymbol{b}$ be a vector of length n with elements in ϱ
- Let $\boldsymbol{A}$ be an n-by-n symmetric matrix with elements in ϱ

$$\boldsymbol{A}_{ij} = \boldsymbol{A}_{ji}$$

- We multiply matrix $\boldsymbol{A}$ by $\boldsymbol{b}$,

$$\boldsymbol{c} = \boldsymbol{A} \cdot \boldsymbol{b}$$

$$\boldsymbol{c}_i = \sum_{j=1}^{n} \underbrace{\boldsymbol{A}_{ij} \cdot \boldsymbol{b}_j}_{\text{nonsymmetric}}$$

- This usual form has cost (ignoring low-order terms here and later)

$$T_{\text{symv}}(\varrho, n) = \mu_\varrho \cdot n^2 + \nu_\varrho \cdot n^2$$

where μ_ϱ is the cost of multiplication and ν_ϱ of addition

Fast symmetric matrix times vector

We can perform `symv` using fewer element-wise multiplications,

$$\boldsymbol{c}_i = \sum_{j=1}^{n} \underbrace{\boldsymbol{A}_{ij} \cdot (\boldsymbol{b}_i + \boldsymbol{b}_j)}_{\text{symmetric}} - \underbrace{\left(\sum_{j=1}^{n} \boldsymbol{A}_{ij}\right) \cdot \boldsymbol{b}_i}_{\text{requires only } n \text{ mults.}}$$

- $\boldsymbol{A}_{ij} \cdot (\boldsymbol{b}_i + \boldsymbol{b}_j)$ is symmetric, requires $\binom{n+1}{2}$ multiplications
- $\left(\sum_{j=1}^{n} \boldsymbol{A}_{ij}\right) \cdot \boldsymbol{b}_i$ may be computed with n multiplications
- The total cost of the new form is

$$T'_{\text{symv}}(\varrho, n) = \mu_\varrho \cdot \frac{1}{2} n^2 + \nu_\varrho \cdot \frac{5}{2} n^2$$

- This formulation is cheaper when $\mu_\varrho > 3\nu_\varrho$
- Form `symm` the formulation is cheaper when $\mu_\varrho > \nu_\varrho$

Symmetric rank-2 update

Consider a rank-2 outer product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ of length n into symmetric matrix $\boldsymbol{C}$

$$\boldsymbol{C} = \boldsymbol{a} \cdot \boldsymbol{b}^{\mathsf{T}} + \boldsymbol{b} \cdot \boldsymbol{a}^{\mathsf{T}}$$

$$\boldsymbol{C}_{ij} = \left[\boldsymbol{a} \cdot \boldsymbol{b}^{\mathsf{T}}\right]_{ij} \equiv \underbrace{\boldsymbol{a}_i \cdot \boldsymbol{b}_j}_{\text{nonsymmetric}} + \underbrace{\boldsymbol{a}_j \cdot \boldsymbol{b}_i}_{\text{permutation}}$$

Usually computed via the n^2 multiplications $\boldsymbol{a}_i \cdot \boldsymbol{b}_j$ with the cost

$$T_{\mathrm{syr2}}(\varrho, n) = \mu_\varrho \cdot n^2 + \nu_\varrho \cdot n^2.$$

Fast symmetric rank-2 update

We may compute the rank-2 update via a symmetric intermediate quantity

$$\boldsymbol{C}_{ij} = \underbrace{(\boldsymbol{a}_i + \boldsymbol{a}_j) \cdot (\boldsymbol{b}_i + \boldsymbol{b}_j)}_{\text{symmetric}} - \underbrace{\boldsymbol{a}_i \cdot \boldsymbol{b}_i - \boldsymbol{a}_j \cdot \boldsymbol{b}_j}_{\text{requires only } n \text{ mults in total}}$$

- We can compute all $(\boldsymbol{a}_i + \boldsymbol{a}_j) \cdot (\boldsymbol{b}_i + \boldsymbol{b}_j)$ in $\binom{n+1}{2}$ multiplications
- The total cost is then given to leading order by

$$T'_{\text{syr2}}(\varrho, n) = \mu_\varrho \cdot \frac{1}{2}n^2 + \nu_\varrho \cdot \frac{5}{2}n^2.$$

- $T'_{\text{syr2}}(\varrho, n) < T_{\text{syr2}}(\varrho, n)$ when $\mu_\varrho > 3\nu_\varrho$
- $T'_{\text{syr2K}}(\varrho, n, K) < T_{\text{syr2K}}(\varrho, n, K)$ when $\mu_\varrho > \nu_\varrho$

Symmetrized product of symmetric matrices

Given symmetric matrices $\boldsymbol{A}, \boldsymbol{B}$ of dimension n on non-associative commutative ring ϱ, we seek to compute the anticommutator of $\boldsymbol{A}$ and $\boldsymbol{B}$

$$\boldsymbol{C} = \boldsymbol{A} \cdot \boldsymbol{B} + \boldsymbol{B} \cdot \boldsymbol{A}$$

$$\boldsymbol{C}_{ij} = [\boldsymbol{A} \cdot \boldsymbol{B}]_{ij} \equiv \sum_{k=1}^{n} \underbrace{\boldsymbol{A}_{ik} \cdot \boldsymbol{B}_{jk}}_{\text{nonsymmetric}} + \sum_{k=1}^{n} \underbrace{\boldsymbol{A}_{jk} \cdot \boldsymbol{B}_{ik}}_{\text{permutation}}$$

The above equations require n^3 multiplications and n^3 adds for a total cost of

$$T_{\text{syrmm}}(\varrho, n) = \mu_\varrho \cdot n^3 + \nu_\varrho \cdot n^3.$$

Note that the symmetrized product defines a non-associative commutative ring (the Jordan ring) over the set of symmetric matrices.

Fast symmetrized product of symmetric matrices

We can combine the ideas from the fast routines for `symv` and `syrk`

$$\boldsymbol{C}_{ij} = \sum_k \underbrace{(\boldsymbol{A}_{ij} + \boldsymbol{A}_{ik} + \boldsymbol{A}_{jk}) \cdot (\boldsymbol{B}_{ij} + \boldsymbol{B}_{ik} + \boldsymbol{B}_{jk})}_{\text{symmetric, requires } \binom{n+2}{3} \text{ mults}}$$

$$- \underbrace{\boldsymbol{A}_{ij} \cdot \left(\sum_k \boldsymbol{B}_{ij} + \boldsymbol{B}_{ik} + \boldsymbol{B}_{jk} \right)}_{\text{requires } \binom{n+1}{2} \text{ mults}} - \underbrace{\boldsymbol{B}_{ij} \cdot \left(\sum_k \boldsymbol{A}_{ij} + \boldsymbol{A}_{ik} + \boldsymbol{A}_{jk} \right)}_{\text{requires } \binom{n+1}{2} \text{ mults}}$$

$$- \underbrace{\sum_k \boldsymbol{A}_{ik} \cdot \boldsymbol{B}_{ik} - \sum_k \boldsymbol{A}_{jk} \cdot \boldsymbol{B}_{jk}}_{\text{requires } \binom{n+1}{2} \text{ mults}}$$

The reformulation requires $\binom{n}{3}$ multiplications to leading order,

$$T'_{\text{syrmm}}(\varrho, n) = \mu_\varrho \cdot \frac{1}{6} n^3 + \nu_\varrho \cdot \frac{5}{3} n^3,$$

which is faster than T_{syrmm} when $\mu_\varrho > (4/5)\nu_\varrho$.

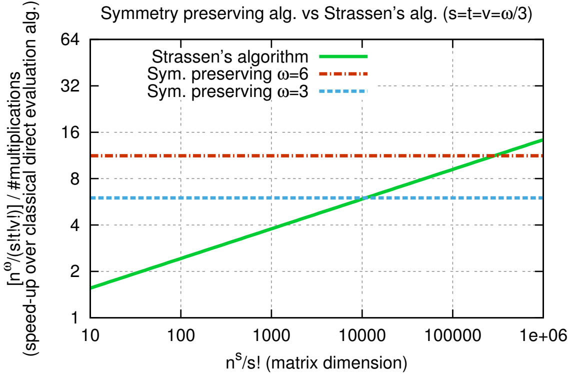
Fast symmetrized product of symmetric matrices

We can rewrite that algorithm in terms using symmetrization notation:

$$\begin{aligned}
\boldsymbol{C}_{ij} = [\boldsymbol{AB}]_{ij} = & \underbrace{\sum_k (\boldsymbol{A}_{ij} + \boldsymbol{A}_{ik} + \boldsymbol{A}_{jk}) \cdot (\boldsymbol{B}_{ij} + \boldsymbol{B}_{ik} + \boldsymbol{B}_{jk})}_{\sum_k [\boldsymbol{A}]_{ijk} \cdot [\boldsymbol{B}]_{ijk}} \\
& - \underbrace{\boldsymbol{A}_{ij} \cdot \left(\sum_k \boldsymbol{B}_{ij} + \boldsymbol{B}_{ik} + \boldsymbol{B}_{jk} \right)}_{\boldsymbol{A}_{ij} \cdot \sum_k [\boldsymbol{B}]_{ijk}} - \underbrace{\boldsymbol{B}_{ij} \cdot \left(\sum_k \boldsymbol{A}_{ij} + \boldsymbol{A}_{ik} + \boldsymbol{A}_{jk} \right)}_{\boldsymbol{B}_{ij} \cdot \sum_k [\boldsymbol{A}]_{ijk}} \\
& - \underbrace{\sum_k \boldsymbol{A}_{ik} \cdot \boldsymbol{B}_{ik} - \sum_k \boldsymbol{A}_{jk} \cdot \boldsymbol{B}_{jk}}_{[\boldsymbol{A}_{ik} \circ_1 \boldsymbol{B}]_{ij}} \\
= & \sum_k [\boldsymbol{A}]_{ijk} \cdot [\boldsymbol{B}]_{ijk} - \boldsymbol{A}_{ij} \sum_k [\boldsymbol{B}]_{ijk} - \boldsymbol{B}_{ij} \sum_k [\boldsymbol{A}]_{ijk} - [\boldsymbol{A} \circ_1 \boldsymbol{B}]_{ij}
\end{aligned}$$

where $\boldsymbol{A} \circ_1 \boldsymbol{B} = \sum_k \boldsymbol{A}_{ik} \cdot \boldsymbol{B}_{jk}$

Comparison to Strassen's algorithm



Fully symmetric tensor contractions

We can now state the general symmetric tensor contraction algorithms, given

- $\boldsymbol{A}$ of order $s+v$, and
- $\boldsymbol{B}$ of order $v+t$ into
- $\boldsymbol{C}$ of order $s+t$, defined as

we define the (nonsymmetrized) contraction as $\boldsymbol{C} = \boldsymbol{A} \odot_v \boldsymbol{B}$ where

$$\boldsymbol{C}_{\boldsymbol{ij}} = \sum_{\boldsymbol{k}\in\{1,\dots,n\}^v} \boldsymbol{A}_{\boldsymbol{ik}} \boldsymbol{B}_{\boldsymbol{kj}}$$

We can then define the symmetrized tensor contraction as

$$\boldsymbol{C}_{\boldsymbol{i}} = [\boldsymbol{A} \odot_v \boldsymbol{B}]_{\boldsymbol{i}}$$

The usual method first computes $\boldsymbol{A} \odot_v \boldsymbol{B}$ with cost

$$T'_{\text{syctr}}(\varrho, n, s, t, v) = \mu_\varrho \cdot \binom{n}{s}\binom{n}{t}\binom{n}{v} + \nu_\varrho \cdot \binom{n}{s}\binom{n}{t}\binom{n}{v}.$$

Fast fully-symmetric contraction algorithm

The fast algorithm is defined as follows (using $\omega = s + t + v$)

$$\boldsymbol{C_i} = \underbrace{\sum_{\boldsymbol{k}\in\{1,\ldots,n\}^v} [\boldsymbol{A}]_{\boldsymbol{ik}} \cdot [\boldsymbol{B}]_{\boldsymbol{ik}}}_{\text{symmetric, requires } \binom{n+\omega-1}{\omega} \text{ multiplications}}$$

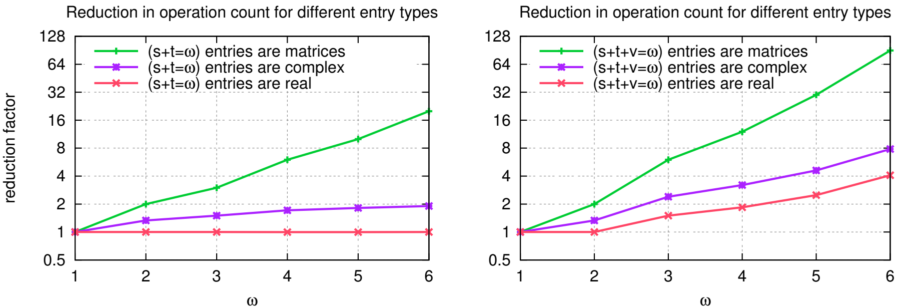
$$- \underbrace{\sum_{p+q=1}^{v} \sum_{\boldsymbol{k}\in\{1,\ldots,n\}^{v-p-q}} \left(\sum_{\boldsymbol{p}\in\{1,\ldots,n\}^p} [\boldsymbol{A}]_{\boldsymbol{ikp}} \right) \cdot \left(\sum_{\boldsymbol{q}\in\{1,\ldots,n\}^q} [\boldsymbol{B}]_{\boldsymbol{ikq}} \right)}_{\text{requires } O(n^{\omega-1}) \text{ multiplications}}$$

$$- \underbrace{\sum_{r=1}^{\min(s,t)} [\boldsymbol{A} \odot_{v+r} \boldsymbol{B}]_{\boldsymbol{i}}}_{\text{requires } O(n^{\omega-1}) \text{ multiplications}}$$

The leading order cost is

$$T'_{\text{syctr}}(\varrho, n, s, t, v) = \mu_\varrho \cdot \binom{n}{\omega} + \nu_\varrho \cdot \binom{n}{\omega} \cdot \left[\binom{\omega}{t} + \binom{\omega}{s} + \binom{\omega}{v} \right].$$

Reduction in operation count of fast algorithm with respect to standard



(s, t, v) values for left and right graph tabulated below

ω	1	2	3	4	4	6
Left graph	$(1, 0, 0)$	$(1, 1, 0)$	$(2, 1, 0)$	$(2, 2, 0)$	$(3, 2, 0)$	$(3, 3, 0)$
Right graph	$(1, 0, 0)$	$(1, 1, 0)$	$(1, 1, 1)$	$(2, 1, 1)$	$(2, 2, 1)$	$(2, 2, 2)$

Theoretical error bounds

We express error bounds in terms of $\gamma_n = \frac{n\epsilon}{1-n\epsilon}$, where ϵ is the machine precision.

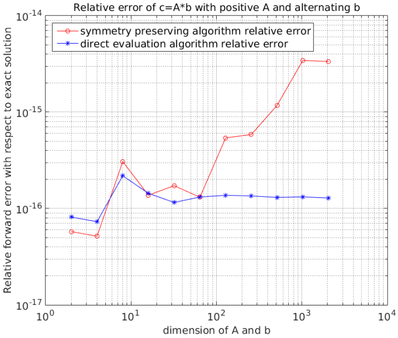
Let Ψ be the standard algorithm and Φ be the fast algorithm. The error bound for the standard algorithm arises from matrix multiplication

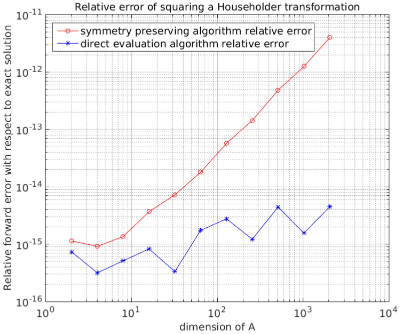
$$||fl\left(\Psi(\boldsymbol{A}, \boldsymbol{B})\right) - \boldsymbol{C}||_\infty \leq \gamma_m \cdot ||\boldsymbol{A}||_\infty \cdot ||\boldsymbol{B}||_\infty \ \text{ where } \ m = \binom{n}{v}\binom{\omega}{v}.$$

The following error bound holds for the fast algorithm

$$||fl\left(\Phi(\boldsymbol{A}, \boldsymbol{B})\right) - \boldsymbol{C}||_\infty \leq \gamma_m \cdot ||\boldsymbol{A}||_\infty \cdot ||\boldsymbol{B}||_\infty \ \text{ where } \ m = 3\binom{n}{v}\binom{\omega}{t}\binom{\omega}{s}.$$

Stability of symmetry preserving algorithms





Nesting the fast algorithm

For partially-(anti)symmetric contractions we can

- nest the new algorithm over each group of symmetric modes
- reduction in mults can translate to reduction in the number of operations
- for Hankel matrices, yields sub $O(n^2)$ algorithm, but not $O(n)$ or $O(n \log(n))$ as reduction in mults is a factor of two only in leading order
- but for coupled-cluster contractions, significant reductions in cost can be achieved
  - CCSD 1.3X on a typical system
  - CCSDT 2.1X on a typical system
  - CCSDTQ 5.7X on a typical system

Communication cost of the standard algorithm

We consider communication bandwidth cost on a sequential machine with cache size M.

The intermediate formed by the standard algorithm may be computed via matrix multiplication with communication cost,

$$W(n,s,t,v,M) = \Theta\left(\frac{\binom{n}{s}\binom{n}{t}\binom{n}{v}}{\sqrt{M}} + \binom{n}{s+v} + \binom{n}{t+v} + \binom{n}{s+t}\right).$$

The cost of symmetrizing the resulting intermediate is low-order or the same.

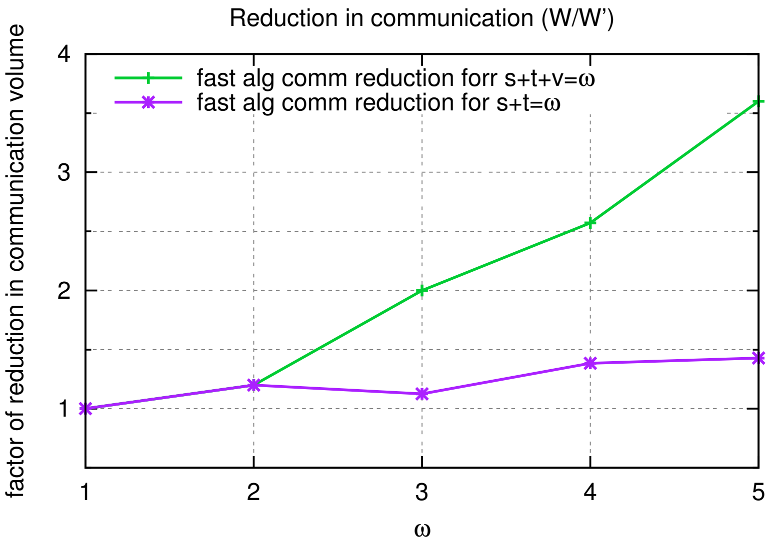
Communication cost of the fast algorithm

We can lower bound the cost of the fast algorithm using the Hölder-Brascamp-Lieb inequality.

An algorithm that blocks $\boldsymbol{Z}$ symmetrically nearly attains the cost

$$W'(n,s,t,v,M) = O\Bigg(\frac{\binom{n}{\omega}}{M^{\omega/(\omega-\min(s,t,v))}} \cdot \left[\binom{\omega}{t} + \binom{\omega}{s} + \binom{\omega}{v}\right] + \binom{n}{s+v} + \binom{n}{t+v} + \binom{n}{s+t}\Bigg).$$

which is not far from the lower bound and attains it when $s = t = v$.


Reduction in communication (W/W')
factor of reduction in communication volume
fast alg comm reduction forr s+t+v=ω
fast alg comm reduction for s+t=ω
1
2
3
4
1
2
3
4
5
ω

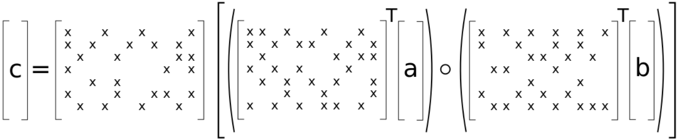
Further communication lower bounds

We can use bilinear algorithm formulations to derive comm. lower bounds

- symmetry preserving tensor contraction algorithms have arbitrary order projections from order d set
- bilinear algorithms[1] provide a more general framework
- a bilinear algorithm is defined by matrices $\boldsymbol{F}^{(\boldsymbol{A})}, \boldsymbol{F}^{(\boldsymbol{B})}, \boldsymbol{F}^{(\boldsymbol{C})}$,

$$\boldsymbol{c} = \boldsymbol{F}^{(\boldsymbol{C})}[(\boldsymbol{F}^{(\boldsymbol{A})\mathsf{T}} a) \circ (\boldsymbol{F}^{(\boldsymbol{B})\mathsf{T}} b)]$$

where $\circ$ is the Hadamard (pointwise) product



- communication lower bounds derived based on matrix rank[2]

[1] Pan, Springer, 1984

[2] S., Hoefler, Demmel, in preparation

Communication cost of symmetry preserving algorithms

For contraction of order $s + v$ tensor with order $v + t$ tensor[3]

- symmetry preserving algorithm requires $\frac{(s+v+t)!}{s!v!t!}$ fewer multiplies
- matrix-vector-like algorithms ($\min(s, v, t) = 0$)
  - vertical communication dominated by largest tensor
  - horizontal communication asymptotically greater if only unique elements are stored and $s \neq v \neq t$
- matrix-matrix-like algorithms ($\min(s, v, t) > 0$)
  - vertical and horizontal communication costs asymptotically greater for symmetry preserving algorithm when $s \neq v \neq t$

[3] S., Hoefler, Demmel; Technical Report, ETH Zurich, 2015.

Summary of results

The following table lists the leading order number of multiplications F required by the standard algorithm and F' by the fast algorithm for various cases of symmetric tensor contractions,

ω	s	t	v	F	F'	applications
2	1	1	0	n^2	$n^2/2$	syr2, syr2k, her2, her2k
2	1	0	1	n^2	$n^2/2$	symv, symm, hemv, hemm
3	1	1	1	n^3	$n^3/6$	symmetrized matmul
s+t+v	s	t	v	$\binom{n}{s}\binom{n}{t}\binom{n}{v}$	$\binom{n}{\omega}$	any symmetric tensor contraction

High-level conclusions:

- The fast symmetric contraction algorithms provide interesting potential arithmetic cost improvements for complex BLAS routines and partially symmetric tensor contractions.
- However, the new algorithms require more communication per flop, incur more numerical error, and usually unable to exploit fused-multiply-add units or blocked matrix multiplication primitives.

Acknowledgements and references

Collaborators on various parts:

- James Demmel
- Torsten Hoefler
- Devin Matthews

S., Demmel; Technical Report, ETH Zurich, 2015.

Backup slides

The fast algorithm for computing $\boldsymbol{C}$ forms the following intermediates with $\binom{n}{\omega}$ multiplications (where $\omega = s + t + v$),

$$\begin{aligned}
Z_{\boldsymbol{i}} =& \Bigg(\sum_{\boldsymbol{j}\in\chi(\boldsymbol{i})} A_{\boldsymbol{j}}\Bigg) \cdot \Bigg(\sum_{\boldsymbol{l}\in\chi(\boldsymbol{i})} B_{\boldsymbol{l}}\Bigg) \\
V_{\boldsymbol{i}} =& \Bigg(\sum_{\boldsymbol{j}\in\chi(\boldsymbol{i})} A_{\boldsymbol{j}}\Bigg) \cdot \Bigg(\sum_{k_1} \sum_{\boldsymbol{l}\in\chi(\boldsymbol{i}\cup\boldsymbol{k})} B_{\boldsymbol{l}}\Bigg) \\
&+ \Bigg(\sum_{k_1} \sum_{\boldsymbol{j}\in\chi(\boldsymbol{i}\cup\boldsymbol{k})} A_{\boldsymbol{j}}\Bigg) \cdot \Bigg(\sum_{\boldsymbol{l}\in\chi(\boldsymbol{i})} B_{\boldsymbol{l}}\Bigg) \\
W_{\boldsymbol{i}} =& \Bigg(\sum_{\boldsymbol{j}\in\chi(\boldsymbol{i})} A_{\boldsymbol{j}}\Bigg) \cdot \Bigg(\sum_{\boldsymbol{l}\in\chi(\boldsymbol{i})} B_{\boldsymbol{l}}\Bigg) \\
C_{\boldsymbol{i}} =& \sum_{\boldsymbol{k}} Z_{\boldsymbol{i}\cup\boldsymbol{k}} - \sum_{\boldsymbol{k}} V_{\boldsymbol{i}\cup\boldsymbol{k}} \\
&- \sum_{\boldsymbol{j}\in\chi(\boldsymbol{i})} \Bigg(\sum_{\boldsymbol{k}} W_{\boldsymbol{j}\cup\boldsymbol{k}}\Bigg)
\end{aligned}$$