Low Rank Bilinear Algorithms for Symmetric Tensor Contractions

Edgar Solomonik

ETH Zurich

SIAM PP, Paris, France
April 14, 2016

Tensor contractions

For some $s, t, v \geq 0$, a tensor contraction of tensors A and B is

$$C_{\vec{i}\vec{j}} = \sum_{\vec{k}} A_{\vec{i}\vec{k}} \cdot B_{\vec{k}\vec{j}}, \quad \text{alternatively written,} \quad C^{\vec{i}}_{\vec{j}} = \sum_{\vec{k}} A^{\vec{i}}_{\vec{k}} \cdot B^{\vec{k}}_{\vec{j}},$$

where $\vec{i} = \{i_1, \ldots, i_s\}$, $\vec{j} = \{j_1, \ldots, j_t\}$, and $\vec{k} = \{k_1, \ldots, k_v\}$.

Matrix/vector examples:

- $(s, t, v) = (0, 0, 1)$ vector inner product
- $(s, t, v) = (1, 0, 1)$ matrix-vector multiplication
- $(s, t, v) = (1, 1, 0)$ vector outer product
- $(s, t, v) = (1, 1, 1)$ matrix-matrix multiplication
- $(s, t, v) = (s, 1, 1)$ tensor-times-matrix

Applications of higher-order tensor contractions

Some applications of contractions of tensors of order at least three:

- tensor factorization algorithms, e.g. alternating least squares
- deep learning convolutional neural networks
- higher-order analysis of probabilistic correlation
- post-Hartree-Fock electronic structure, e.g. coupled cluster
- density matrix renormalization group (DMRG)

Contractions in Coupled Cluster (CCSD method)

$$
\begin{aligned}
W^{mn}_{ei} &= V^{mn}_{ei} + \sum_{f} V^{mn}_{ef} t^{f}_{i}, \\
X^{mn}_{ij} &= V^{mn}_{ij} + P^{i}_{j} \sum_{e} V^{mn}_{ie} t^{e}_{j} + \frac{1}{2} \sum_{ef} V^{mn}_{ef} \tau^{ef}_{ij}, \\
U^{am}_{ie} &= V^{am}_{ie} - \sum_{n} W^{mn}_{ei} t^{a}_{n} + \sum_{f} V^{ma}_{ef} t^{f}_{i} + \frac{1}{2} \sum_{nf} V^{mn}_{ef} T^{af}_{in}, \\
Q^{am}_{ij} &= V^{am}_{ij} + P^{i}_{j} \sum_{e} V^{am}_{ie} t^{e}_{j} + \frac{1}{2} \sum_{ef} V^{am}_{ef} \tau^{ef}_{ij}, \\
z^{a}_{i} &= f^{a}_{i} - \sum_{m} F^{m}_{i} t^{a}_{m} + \sum_{e} f^{a}_{e} t^{e}_{i} + \sum_{em} V^{ma}_{ei} t^{e}_{m} + \sum_{em} V^{ae}_{im} F^{m}_{e} + \frac{1}{2} \sum_{efm} V^{am}_{ef} \tau^{ef}_{im} \\
&\quad - \frac{1}{2} \sum_{emn} W^{mn}_{ei} T^{ea}_{mn}, \\
Z^{ab}_{ij} &= V^{ab}_{ij} + P^{i}_{j} \sum_{e} V^{ab}_{ie} t^{e}_{j} + P^{a}_{b} P^{i}_{j} \sum_{me} U^{am}_{ie} T^{eb}_{mj} - P^{a}_{b} \sum_{m} Q^{am}_{ij} t^{b}_{m} \\
&\quad + P^{a}_{b} \sum_{e} F^{a}_{e} T^{eb}_{ij} - P^{i}_{j} \sum_{m} F^{m}_{i} T^{ab}_{mj} + \frac{1}{2} \sum_{ef} V^{ab}_{ef} \tau^{ef}_{ij} + \frac{1}{2} \sum_{mn} X^{mn}_{ij} \tau^{ab}_{mn},
\end{aligned}
$$

where $P^{x}_{y} f(x, y) := f(x, y) - f(y, x)$

Exploiting symmetry in tensor contractions

Tensor symmetry (e.g. $A_{ij} = A_{ji}$) reduces memory and cost

- for order d tensor, $d!$ less memory
- dot product $\sum_{i,j} A_{ij} B_{ij} = 2 \sum_{i<j} A_{ij} B_{ij} + \sum_i A_{ii} B_{ii}$
- matrix-vector multiplication ($A_{ij} = A_{ji}$)

$$c_i = \sum_j A_{ij} b_j \quad = \quad \sum_j A_{ij}(b_i + b_j) - \left(\sum_j A_{ij} \right) b_i$$

- $A_{ij} b_j \neq A_{ji} b_i$ but $A_{ij}(b_i + b_j) = A_{ji}(b_j + b_i) \rightarrow (1/2)n^2$ multiplies
- partially-symmetric case: $T_{ij}^{ab} = -T_{ji}^{ab}$

$$W_{ic}^{ak} = \sum_j \sum_b T_{ij}^{ab} V_{bc}^{jk}$$

$$= \sum_j \left(\sum_b T_{ij}^{ab} (V_{bc}^{ik} + V_{bc}^{jk}) \right) - \sum_b \left(\sum_j T_{ij}^{ab} \right) V_{bc}^{ik}$$

- $Z_{ijc}^{ak} = \sum_b T_{ij}^{ab} (V_{bc}^{ik} + V_{bc}^{jk}) = -Z_{jic}^{ak} \rightarrow$ **2x fewer operations**

Symmetry preserving algorithms

By exploiting symmetry, reduce multiplies (but increase adds)

- rank-2 vector outer product

$$C_{ij} = a_i b_j + a_j b_i \quad = \quad (a_i + a_j)(b_i + b_j) - a_i b_i - a_j b_j$$

- squaring a symmetric matrix A (or $AB + BA$)

$$C_{ij} = \sum_k A_{ik} A_{kj} \quad = \quad \sum_k (A_{ik} + A_{kj} + A_{ij})^2 - \dots$$

- fully symmetric contraction of order $s + v$ and $v + t$ tensors

$$\frac{(s+t+v)!}{s!t!v!} \quad \text{fewer multiplies}$$

e.g. cases above are

- $(s, t, v) = (1, 1, 0) \rightarrow$ reduction by 2X
- $(s, t, v) = (1, 1, 1) \rightarrow$ reduction by 6X

Applications of symmetry preserving algorithms

Extensions and applications:

- numerically stable by forward error bounds and experiments
- for Hermitian tensors, multiplies cost 3X more than adds
  - Hermitian matrix multiplication and tridiagonal reduction (BLAS and LAPACK routines) with 25% fewer operations
- cost reductions in partially-symmetric coupled cluster contractions: 2X-9X for select contractions, 1.3X, 2.1X for CCSD, CCSDT
- $(2/3)n^3$ multiplies for squaring a *nonsymmetric* matrix

$$X_{\mathrm{SY}} := \frac{1}{2}(X + X^{\mathrm{T}}), \quad X_{\mathrm{AS}} := \frac{1}{2}(X - X^{\mathrm{T}}),$$

$$\begin{aligned} C &= AB + (A^{\mathrm{T}}B^{\mathrm{T}})^{\mathrm{T}} = AB + BA \\ &= (A_{\mathrm{SY}}B_{\mathrm{SY}})_{\mathrm{SY}} + (A_{\mathrm{SY}}B_{\mathrm{AS}})_{\mathrm{AS}} + (A_{\mathrm{AS}}B_{\mathrm{SY}})_{\mathrm{AS}} + (A_{\mathrm{AS}}B_{\mathrm{AS}})_{\mathrm{SY}} \end{aligned}$$

four invocations of $(s, t, v) = (1, 1, 1)$, squaring when $A = B$

Symmetry preserving blocking (sketch)

Multiplication of a symmetric matrix A and a nonsymmetric matrix B:

- classical approach, two choices:
  1. treat A as nonsymmetric (unpack if stored as symmetric)
  2. multiply by lower-triangle of A then by its transpose
- proposed new approach
  - fold $n \times n$ matrix A into $\sqrt{p} \times \frac{n}{\sqrt{p}} \times \sqrt{p} \times \frac{n}{\sqrt{p}}$ tensor T
  - note that $T^{ij}_{kl} = T^{ji}_{lk}$, define partially-symmetric $Y^{ij}_{kl} = T^{ij}_{kl} + T^{ij}_{lk}$ and partially-antisymmetric $S^{ij}_{kl} = T^{ij}_{kl} - T^{ij}_{lk}$
  - use symmetry preserving alg. over indices of dims $\sqrt{p} \times \sqrt{p}$, results in p subproblems with symmetric matrices with dims $\frac{n}{\sqrt{p}} \times \frac{n}{\sqrt{p}}$
- food for thought: keep folding/symmetrizing to $2 \times \cdots \times 2$ tensors $\rightarrow$ Hankel matrices (modulo sign interchanges)

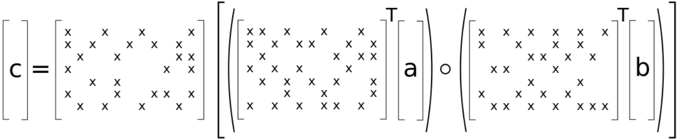
Bilinear algorithms

Bilinear algorithms[1] for symmetric contractions

- a bilinear algorithm is defined by matrices $F^{(A)}, F^{(B)}, F^{(C)}$,

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

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



- the number of rows in each matrix corresponds to the number of inputs (dimensions of a and b) and outputs (dimension of c)
- for clasiscal $n \times n$ matrix multiplication $F^{(A)}, F^{(B)}, F^{(C)}$ are $n^2 \times n^3$ and have one unit entry per column
- number of columns in $F^{(A)}, F^{(B)}, F^{(C)}$ is the *bilinear algorithm rank*

[1] Pan, How to Multiply Matrices Faster, Springer, 1984

Bilinear algorithms as tensor factorizations

A bilinear algorithm corresponds to a CP tensor decomposition

$$c_i = \sum_{r=1}^{R} F_{ir}^{(C)} \left(\sum_j F_{jr}^{(A)} a_j \right) \left(\sum_k F_{kr}^{(B)} b_k \right)$$

$$= \sum_j \sum_k \left(\sum_{r=1}^{R} F_{ir}^{(C)} F_{jr}^{(A)} F_{kr}^{(B)} \right) a_j b_k$$

$$= \sum_j \sum_k T_{ijk} a_j b_k \quad \text{where} \quad T_{ijk} = \sum_{r=1}^{R} F_{ir}^{(C)} F_{jr}^{(A)} F_{kr}^{(B)}$$

For multiplication of $n \times n$ matrices,

- T is $n^2 \times n^2 \times n^2$
- classical algorithm has rank $R = n^3$
- Strassen's algorithm has rank $R \approx n^{\log_2(7)}$

Bilinear algorithms as tensor factorizations

A bilinear algorithm corresponds to a CP tensor decomposition

$$c_i = \sum_{r=1}^{R} F^{(C)}_{ir} \Big(\sum_j F^{(A)}_{jr} a_j \Big) \Big(\sum_k F^{(B)}_{kr} b_k \Big)$$

$$= \sum_j \sum_k \Big(\sum_{r=1}^{R} F^{(C)}_{ir} F^{(A)}_{jr} F^{(B)}_{kr} \Big) a_j b_k$$

$$= \sum_j \sum_k T_{ijk} a_j b_k \quad \text{where} \quad T_{ijk} = \sum_{r=1}^{R} F^{(C)}_{ir} F^{(A)}_{jr} F^{(B)}_{kr}$$

For symmetric tensor contractions (not counting diagonals)

- T is $\binom{n}{s+t} \times \binom{n}{s+v} \times \binom{n}{v+t}$
- classical algorithm has rank $R = \binom{n}{s}\binom{n}{t}\binom{n}{v}$
- symmetry preserving $\to R \approx \binom{n}{s+t+v}$, that is $\frac{(s+t+v)!}{s!t!v!}$ less

Expansion in bilinear algorithms

Given $\Lambda = (F^{(A)}, F^{(B)}, F^{(C)})$, $\Lambda_{\text{sub}} \subseteq \Lambda$ if $\exists$ projection matrix P, so

$$\Lambda_{\text{sub}} = (F^{(A)}P, F^{(B)}P, F^{(C)}P),$$

the projection matrix extracts $\#\text{cols}(P)$ columns of each matrix.

A bilinear algorithm Λ has expansion bound $\mathcal{E}_\Lambda : \mathbb{N}^3 \to \mathbb{N}$, if for all

$$\Lambda_{\text{sub}} := (F^{(A)}_{\text{sub}}, F^{(B)}_{\text{sub}}, F^{(C)}_{\text{sub}}) \subseteq \Lambda$$

we have

$$\text{rank}(\Lambda_{\text{sub}}) \leq \mathcal{E}_\Lambda \left(\text{rank}(F^{(A)}_{\text{sub}}), \text{rank}(F^{(B)}_{\text{sub}}), \text{rank}(F^{(C)}_{\text{sub}})\right)$$

For matrix mult., Loomis-Whitney inequality $\to \mathcal{E}_{\text{MM}}(x, y, z) = \sqrt{xyz}$
For sym. pres. $\mathcal{E}_{\text{SP}}^{(s,t,v)}(x, y, z) = O\Big(\min\big(x^{\frac{s+t+v}{s+v}}, y^{\frac{s+t+v}{t+v}}, z^{\frac{s+t+v}{s+t}}\big)\Big)$

Communication in symmetry preserving algorithms

Communication lower bounds based on bilinear algorithm expansion

- horizontal comm. – max data sent or received
- vertical comm. – max data moved between memory and cache

For contraction of order $s + v$ tensor with order $v + t$ tensor

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

Conclusion

Summary:

- symmetry preserving algorithms reduce cost of contractions
- they have been tested using Cyclops Tensor Framework
  `https://github.com/solomonik/ctf`
- rank structure of bilinear algorithms yields communication bounds

Future work:

- communication lower bounds for partially-symmetric cases
- high performance implementation

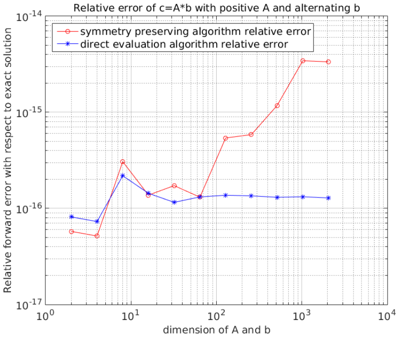
Related work: *J. Noga and P. Valiron, Improved algorithm for triple-excitation contributions within the coupled cluster approach, Molecular Physics, 103 (2005).*

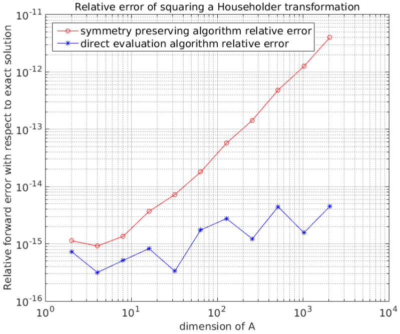
References (for more, email `solomonik@inf.ethz.ch`):

- *E.S. and J. Demmel; Contracting symmetric tensors using fewer multiplications; ETH Zurich, 2015*
- *E.S., J. Demmel, and T. Hoefler; Communication lower bounds for tensor contraction algorithms; ETH Zurich, 2015*

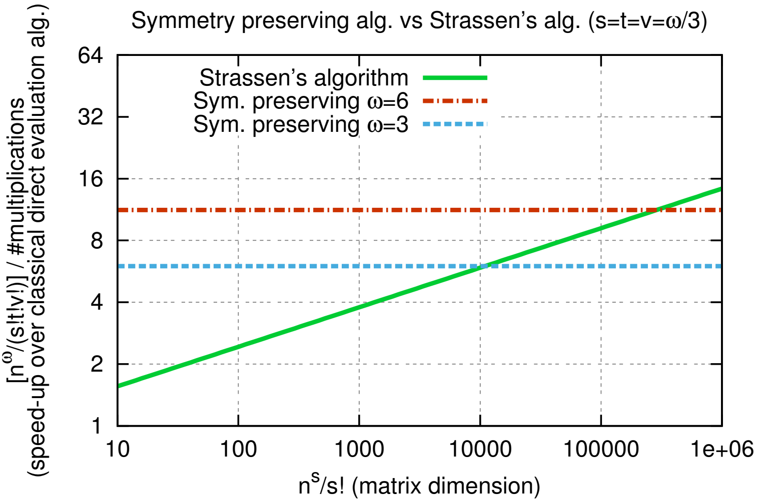
Backup slides

Stability of symmetry preserving algorithms





Symmetry preserving algorithm vs Strassen's algorithm



A library for tensor computations

Cyclops Tensor Framework

- implicit for loops based on index notation (Einstein summation)
- matrix sums, multiplication, Hadamard product (tensor contractions)
- distributed symmetric-packed/sparse storage via cyclic layout

Jacobi iteration (solves $Ax = b$ iteratively) example code snippet

```
Vector<> Jacobi(Matrix<> A, Vector<> b, int n){
  ... // split A = R + diag(1./d)
  do {
    x["i"] = d["i"]*(b["i"]-R["ij"]*x["j"]);
    r["i"] = b["i"]-A["ij"]*x["j"]; // compute residual
  } while (r.norm2() > 1.E-6); // check for convergence
  return x;
}
```

Coupled cluster using CTF

Extracted from Aquarius (Devin Matthews' code, `https://github.com/devinamatthews/aquarius`)

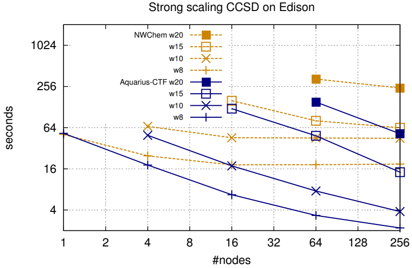
```
FMI["mi"]      += 0.5*WMNEF["mnef"]*T2["efin"];
WMNIJ["mnij"] += 0.5*WMNEF["mnef"]*T2["efij"];
FAE["ae"]      -= 0.5*WMNEF["mnef"]*T2["afmn"];
WAMEI["amei"] -= 0.5*WMNEF["mnef"]*T2["afin"];

Z2["abij"]  = WMNEF["ijab"];
Z2["abij"] += FAE["af"]*T2["fbij"];
Z2["abij"] -= FMI["ni"]*T2["abnj"];
Z2["abij"] += 0.5*WABEF["abef"]*T2["efij"];
Z2["abij"] += 0.5*WMNIJ["mnij"]*T2["abmn"];
Z2["abij"] -= WAMEI["amei"]*T2["ebmj"];
```

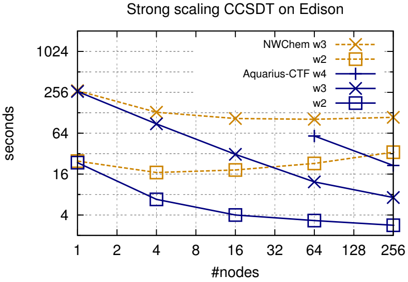
CTF is used within **Aquarius, QChem, VASP, and Psi4**

Comparison with NWChem

NWChem is the most commonly-used distributed-memory quantum chemistry method suite

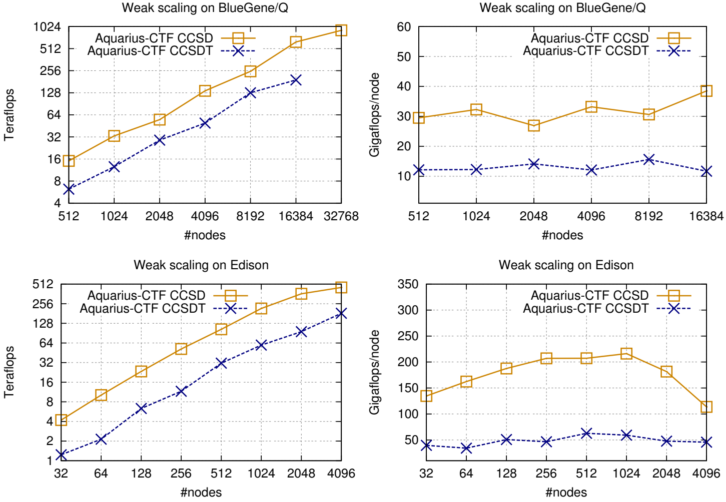
- provides Coupled Cluster methods: CCSD and CCSDT
- derives equations via Tensor Contraction Engine (TCE)
- generates contractions as blocked loops leveraging Global Arrays





Coupled cluster on IBM BlueGene/Q and Cray XC30

CCSD up to 55 (50) water molecules with cc-pVDZ
CCSDT up to 10 water molecules with cc-pVDZ[a]



[a] S., Matthews, Hammond, Demmel, JPDC, 2014

Coupled cluster methods

Coupled cluster provides a systematically improvable approximation to the manybody time-independent Schrödinger equation $H|\Psi\rangle = E|\Psi\rangle$

- the Hamiltonian has one- and two- electron components $H = F + V$
- Hartree-Fock (SCF) computes mean-field Hamiltonian: F, V
- Coupled-cluster methods (CCSD, CCSDT, CCSDTQ) consider transitions of (doubles, triples, and quadruples) of electrons to unoccupied orbitals, encoded by tensor operator, $T = T_1 + T_2 + T_3 + T_4$
- they use an exponential ansatz for the wavefunction, $\Psi = e^T \phi$ where ϕ is a Slater determinant
- expanding $0 = \langle \phi' | H | \Psi \rangle$ yields nonlinear equations for $\{T_i\}$ in F, V

$$0 = V_{ij}^{ab} + P(a,b) \sum_e T_{ij}^{ae} F_e^b - \frac{1}{2} P(i,j) \sum_{mnef} T_{im}^{ab} V_{ef}^{mn} T_{jn}^{ef} + \ldots$$

where P is an antisymmetrization operator

Vertical communication in bilinear algorithms

Any schedule on a sequential machine with a cache of size H for $\Lambda = (F^{(A)}, F^{(B)}, F^{(C)})$ with expansion bound $\mathcal{E}_\Lambda$ has vertical communication cost,

$$Q_\Lambda \geq \max\left[\frac{2\,\text{rank}(\Lambda)H}{\mathcal{E}_\Lambda^{\max}(H)}, \#\text{rows}(F^{(A)}) + \#\text{rows}(F^{(B)}) + \#\text{rows}(F^{(C)})\right]$$

where $\mathcal{E}_\Lambda^{\max}(H) := \max_{c^{(A)},c^{(B)},c^{(C)}\in\mathbb{N},c^{(A)}+c^{(B)}+c^{(C)}=3H} \mathcal{E}_\Lambda(c^{(A)}, c^{(B)}, c^{(C)})$

Vertical communication in matrix multiplication

For the classical (non-Strassen-like) matrix multiplication algorithm of m-by-k matrix A with k-by-n matrix B into m-by-n matrix C,

$$\mathcal{E}_{\text{MM}}(c^{(A)}, c^{(B)}, c^{(C)}) = (c^{(A)}c^{(B)}c^{(C)})^{1/2}$$

further, we have

$$\mathcal{E}_{\text{MM}}^{\max}(H) = \max_{c^{(A)}, c^{(B)}, c^{(C)} \in \mathbb{N}, c^{(A)}+c^{(B)}+c^{(C)} \leq 3H} (c^{(A)}c^{(B)}c^{(C)})^{1/2} = H^{3/2}$$

so we obtain the expected bound,

$$\begin{aligned} Q_{\text{MM}} &\geq \max\left[\frac{2\,\text{rank}(\text{MM})H}{\mathcal{E}_{\text{MM}}^{\max}(H)}, \#\text{rows}(F^{(A)}) + \#\text{rows}(F^{(B)}) + \#\text{rows}(F^{(C)})\right] \\ &= \max\left[\frac{2mnk}{\sqrt{H}}, mk + kn + mn\right] \end{aligned}$$

Horizontal communication in bilinear algorithms

Any load balanced schedule on a parallel machine with p processes of $\Lambda = (F^{(A)}, F^{(B)}, F^{(C)})$ with expansion bound $\mathcal{E}_\Lambda$ has horizontal communication cost,

$$W_\Lambda \geq c^{(A)} + c^{(B)} + c^{(C)}$$

for some (communicated amounts) $c^{(A)}, c^{(B)}, c^{(C)} \in \mathbb{N}$ such that,

$$\begin{aligned}\text{rank}(\Lambda)/p \leq \mathcal{E}_\Lambda(&c^{(A)} + \#\text{rows}(F^{(A)})/p, \\ &c^{(B)} + \#\text{rows}(F^{(B)})/p, \\ &c^{(C)} + \#\text{rows}(F^{(C)})/p)\end{aligned}$$

Horizontal communication in matrix multiplication

For the classical (non-Strassen-like) matrix multiplication algorithm of m-by-k matrix A with k-by-n matrix B into m-by-n matrix C on a parallel machine of p processors,

$$W_{\mathrm{MM}} = \Omega\left(W_{\mathrm{O}}(\min(m,n,k), \mathrm{median}(m,n,k), \max(m,n,k), p)\right)$$

where

$$W_{\mathrm{O}}(x,y,z,p) = \begin{cases} \left(\frac{xyz}{p}\right)^{2/3} & : p > yz/x^2 \\ x\left(\frac{yz}{p}\right)^{1/2} & : yz/x^2 \geq p > z/y \\ xy & : z/y \geq p \end{cases}$$

Communication lower bounds for direct evaluation of symmetric contractions

An expansion bound on $\Psi^{(s,t,v)}$ is

$$\mathcal{E}_{\Psi}^{(s,t,v)}(c^{(A)}, c^{(B)}, c^{(C)}) = q\left(c^{(A)}c^{(B)}c^{(C)}\right)^{1/2},$$

where $q = \left[\binom{s+v}{s}\binom{v+t}{v}\binom{s+t}{s}\right]^{1/2}$.

Therefore, the same (asymptotically) horizontal and vertical communication lower bounds apply for $\Psi^{(s,t,v)}$ as for a matrix multiplication with dimensions $n^s \times n^t \times n^v$.

Communication lower bounds for direct evaluation of symmetric contractions

Another expansion bound on $\Psi^{(s,t,0)}$ (when $v = 0$) is

$$\mathcal{E}_{\Psi}^{(s,t,0)}(c^{(A)}, c^{(B)}, c^{(C)}) = \left(\binom{\omega}{s} - 1\right) c^{(C)} + \min\left((c^{(A)})^{\omega/s}, (c^{(B)})^{\omega/t}, c^{(C)}\right)$$

There are also symmetric bounds when $s = 0$ or $t = 0$.
When exactly one of s, t, v is zero, any load balanced schedule of $\Psi^{(s,t,v)}$ on a parallel machine with p processors has horizontal communication cost,

$$W_{\Psi} = \Omega\left((n^{\omega}/p)^{\max(s,t,v)/\omega}\right)$$

This can be greater than the corresponding nonsymmetric bound,

$$W_{\Psi} = \Omega\left((n^{\omega}/p)^{1/2}\right)$$

Communication lower bounds for the symmetry preserving algorithm

An expansion bound on $\Phi^{(s,t,v)}$ is

$$\mathcal{E}_{\Phi}^{(s,t,v)}(c^{(A)}, c^{(B)}, c^{(C)}) = \min\left(\left(\binom{\omega}{t} c^{(A)}\right)^{\frac{\omega}{s+v}}, \left(\binom{\omega}{s} c^{(B)}\right)^{\frac{\omega}{v+t}}, \left(\binom{\omega}{v} c^{(C)}\right)^{\frac{\omega}{s+t}} \right)$$

This yields communication bounds with $\kappa := \max(s+v, v+t, s+t)$,

$$Q_{\Phi} = \Omega\left(\frac{n^{\omega} H}{H^{\omega/\kappa}} + n^{\kappa}\right) \qquad W_{\Phi} = \begin{cases} \Omega\left((n^{\omega}/p)^{\kappa/\omega}\right) & : s,t,v > 0 \\ \Omega\left((n^{\omega}/p)^{\max(s,t,v)/\omega}\right) & : \kappa = \omega \end{cases}$$

Nesting of bilinear algorithms

Given two bilinear algorithms:

$$\Lambda_1 =(F_1^{(A)}, F_1^{(B)}, F_1^{(C)})$$
$$\Lambda_2 =(F_2^{(A)}, F_2^{(B)}, F_2^{(C)})$$

We can nest them by computing their tensor product

$$\Lambda_1 \otimes \Lambda_2 :=(F_1^{(A)} \otimes F_2^{(A)}, F_1^{(B)} \otimes F_2^{(B)}, F_1^{(C)} \otimes F_2^{(C)})$$
$$\text{rank}(\Lambda_1 \otimes \Lambda_2) = \text{rank}(\Lambda_1) \cdot \text{rank}(\Lambda_2)$$

Communication lower bounds for nested algorithms

Conjecture: if bilinear algorithms λ_1 and λ_2 have expansion bounds $\mathcal{E}_1$ and $\mathcal{E}_2$, then $\lambda_1 \otimes \lambda_2$ has expansion bound, $\mathcal{E}_{12}(c^{(A)}, c^{(B)}, c^{(C)})$

$$= \max_{\substack{c_1^{(A)}, c_1^{(B)}, c_1^{(C)}, c_2^{(A)}, c_2^{(B)}, c_2^{(C)} \in \mathbb{N} \\ c_1^{(A)} c_2^{(A)} = c^{(A)}, c_1^{(B)} c_2^{(B)} = c^{(B)}, c_1^{(C)} c_2^{(C)} = c^{(C)}}} \left[\mathcal{E}_1(c_1^{(A)}, c_1^{(B)}, c_1^{(C)}) \mathcal{E}_2(c_2^{(A)}, c_2^{(B)}, c_2^{(C)}) \right]$$

Simplified conjecture: consider matrices A and B, such that for some $\alpha, \beta \in [0, 1]$ and any $k \in \mathbb{N}$

- any subset of k columns of A has rank at least k^{α}
- any subset of k columns of B has rank at least k^{β}

then any subset of $k \in \mathbb{N}$ columns of $A \otimes B$ has rank at least $k^{\min(\alpha,\beta)}$

The first conjecture would provide lower bounds for the nested algorithms we wish to use for partially-symmetric coupled-cluster contractions.