Minimizing communication in tensor contraction algorithms

Edgar Solomonik

ETH Zurich

SIAM LA, Atlanta GA
Oct 27, 2015

Exploiting symmetry by unfolding

Let **A** and **B** be two $n \times n$ antisymmetric matrices and consider the contraction,

$$c = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} \cdot B_{ij} = 2 \sum_{i=1}^{n} \sum_{j=1}^{i-1} A_{ij} \cdot B_{ij}$$

This contraction may be unfolded into an inner product of vectors,

$$c = \langle \mathrm{vec}(\mathbf{A}), \mathrm{vec}(\mathbf{B}) \rangle = \langle \mathrm{vech}(\mathbf{A}), \mathrm{vech}(\mathbf{B}) \rangle$$

where vech (half-vectorization) takes only the unique entries.

Exploiting symmetry by unfolding

Let **A** and **B** be two $n \times n$ antisymmetric matrices and consider the contraction,

$$c = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} \cdot B_{ij} = 2 \sum_{i=1}^{n} \sum_{j=1}^{i-1} A_{ij} \cdot B_{ij}$$

This contraction may be unfolded into an inner product of vectors,

$$c = \langle \text{vec}(\mathbf{A}), \text{vec}(\mathbf{B}) \rangle = \langle \text{vech}(\mathbf{A}), \text{vech}(\mathbf{B}) \rangle$$

where vech (half-vectorization) takes only the unique entries.
This technique is 8X faster for the following CCSD contraction,

$$Z_{ij}^{ab} = \sum_{e,f} V_{ef}^{ab} \cdot T_{ij}^{ef} \quad \rightarrow \quad Z_{i<j}^{a<b} = \sum_{e<f} V_{e<f}^{a<b} \cdot T_{i<j}^{e<f}$$

as the tensors are antisymmetric in (a, b), (i, j), and (e, f).

Symmetry that does not conform to unfoldings

Consider the multiplication of an antisymmetric matrix **A** with a vector **b**,

$$c_i = \sum_j A_{ij} \cdot b_j$$

while $A_{ij} = -A_{ji}$, the quantities $A_{ij} b_j$ and $A_{ji} b_i$ are arbitrarily different.

Symmetry that does not conform to unfoldings

Consider the multiplication of an antisymmetric matrix **A** with a vector **b**,

$$c_i = \sum_j A_{ij} \cdot b_j$$

while $A_{ij} = -A_{ji}$, the quantities $A_{ij} b_j$ and $A_{ji} b_i$ are arbitrarily different. Now consider another contraction from the CCSD method,

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

where **T** is partially antisymmetric,

$$T^{ab}_{ij} = -T^{ba}_{ij} = -T^{ab}_{ji} = T^{ba}_{ji}$$

it is not possible to unfold these tensors and obtain a reduced-size matrix multiplication.

Symmetric-matrix–vector multiplication

- Consider symmetric $n \times n$ matrix $\mathbf{A}$ and vectors $\mathbf{b}, \mathbf{c}$

Symmetric-matrix–vector multiplication

- Consider symmetric $n \times n$ matrix $\mathbf{A}$ and vectors $\mathbf{b}, \mathbf{c}$
- $\mathbf{c} = \mathbf{A} \cdot \mathbf{b}$ is usually done by computing a *nonsymmetric* intermediate matrix $\mathbf{W}$,

$$W_{ij} = A_{ij} \cdot b_j \qquad\qquad c_i = \sum_{j=1}^{n} W_{ij}$$

which requires n^2 multiplications and n^2 additions.

Symmetric-matrix–vector multiplication

- Consider symmetric $n \times n$ matrix $\mathbf{A}$ and vectors $\mathbf{b}, \mathbf{c}$
- $\mathbf{c} = \mathbf{A} \cdot \mathbf{b}$ is usually done by computing a *nonsymmetric* intermediate matrix $\mathbf{W}$,

$$W_{ij} = A_{ij} \cdot b_j \qquad c_i = \sum_{j=1}^{n} W_{ij}$$

  which requires n^2 multiplications and n^2 additions.

- The *symmetry preserving algorithm* employs a *symmetric* intermediate matrix $\mathbf{Z}$,

$$Z_{ij} = A_{ij} \cdot (b_i + b_j) \qquad c_i = \sum_{j=1}^{n} Z_{ij} - \left(\sum_{j=1}^{n} A_{ij} \right) \cdot b_i$$

  which requires $\frac{n^2}{2}$ multiplications and $\frac{5n^2}{2}$ additions.

Symmetrized rank-two outer product

- Consider vectors $\mathbf{a}, \mathbf{b}$ of dimension n

Symmetrized rank-two outer product

- Consider vectors $\mathbf{a}, \mathbf{b}$ of dimension n
- Symmetric matrix $\mathbf{C} = \mathbf{a} \cdot \mathbf{b}^\mathsf{T} + \mathbf{b} \cdot \mathbf{a}^\mathsf{T}$ is usually done by computing a *nonsymmetric* intermediate matrix $\mathbf{W}$,

$$W_{ij} = a_i \cdot b_j \qquad\qquad C_{ij} = W_{ij} + W_{ji}$$

which requires n^2 multiplications and $n^2/2$ additions.

Symmetrized rank-two outer product

- Consider vectors $\mathbf{a}, \mathbf{b}$ of dimension n
- Symmetric matrix $\mathbf{C} = \mathbf{a} \cdot \mathbf{b}^\mathsf{T} + \mathbf{b} \cdot \mathbf{a}^\mathsf{T}$ is usually done by computing a *nonsymmetric* intermediate matrix $\mathbf{W}$,

$$W_{ij} = a_i \cdot b_j \qquad\qquad C_{ij} = W_{ij} + W_{ji}$$

  which requires n^2 multiplications and $n^2/2$ additions.
- The *symmetry preserving algorithm* employs a *symmetric* intermediate matrix $\mathbf{Z}$,

$$Z_{ij} = (a_i + a_j) \cdot (b_i + b_j) \qquad C_{ij} = Z_{ij} - a_i \cdot b_i - a_j \cdot b_j$$

  which requires $\frac{n^2}{2}$ multiplications and $2n^2$ additions.

Symmetrized matrix multiplication

- Consider symmetric $n \times n$ matrices **A**, **B**, and **C**

Symmetrized matrix multiplication

- Consider symmetric $n \times n$ matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$
- $\mathbf{C} = \mathbf{A} \cdot \mathbf{B} + \mathbf{B} \cdot \mathbf{A}$ is usually computed via a nonsymmetric intermediate order 3 tensor $\mathbf{W}$,

$$W_{ijk} = A_{ik} \cdot B_{kj} \qquad \bar{W}_{ij} = \sum_k W_{ijk} \qquad C_{ij} = W_{ij} + W_{ji}.$$

which requires n^3 multiplications and n^3 additions.

Symmetrized matrix multiplication

- Consider symmetric $n \times n$ matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$
- $\mathbf{C} = \mathbf{A} \cdot \mathbf{B} + \mathbf{B} \cdot \mathbf{A}$ is usually computed via a nonsymmetric intermediate order 3 tensor $\mathbf{W}$,

$$W_{ijk} = A_{ik} \cdot B_{kj} \qquad \bar{W}_{ij} = \sum_k W_{ijk} \qquad C_{ij} = W_{ij} + W_{ji}.$$

  which requires n^3 multiplications and n^3 additions.
- The *symmetry preserving algorithm* employs a *symmetric* intermediate tensor $\mathbf{Z}$ using $n^3/6$ multiplications and $7n^3/6$ additions,

$$Z_{ijk} = (A_{ij} + A_{ik} + A_{jk}) \cdot (B_{ij} + B_{ik} + B_{jk}) \qquad v_i = \sum_{k=1}^{n} A_{ik} \cdot B_{ik}$$

$$C_{ij} = \sum_{k=1}^{n} Z_{ijk} - n \cdot A_{ij} \cdot B_{ij} - v_i - v_j - \left(\sum_{k=1}^{n} A_{ik} \right) \cdot B_{ij} - A_{ij} \cdot \left(\sum_{k=1}^{n} B_{ik} \right)$$

Symmetry preserving algorithm

Consider contraction of symmetric tensors **A** of order $s + v$ and **B** of order $v + t$ that is symmetrized to produce a symmetric tensor **C** of order $s + t$

Symmetry preserving algorithm

Consider contraction of symmetric tensors **A** of order $s+v$ and **B** of order $v+t$ that is symmetrized to produce a symmetric tensor **C** of order $s+t$

- Let $\omega = s+t+v$
- the symmetry preserving algorithm computes the order ω symmetric tensor $\hat{\mathbf{Z}}$, $\forall \vec{\mathbf{i}} = (i_1, \ldots, i_\omega)$, $1 \leq i_1 \leq \cdots \leq i_\omega \leq n$,

$$\begin{aligned} \vec{\mathbf{j}} \in \chi^{s+v}(\vec{\mathbf{i}}), &\quad \hat{A}_{\vec{\mathbf{i}}} \leftarrow A_{\vec{\mathbf{j}}} \\ \vec{\mathbf{l}} \in \chi^{v+t}(\vec{\mathbf{i}}), &\quad \hat{B}_{\vec{\mathbf{i}}} \leftarrow B_{\vec{\mathbf{l}}} \\ &\quad \hat{Z}_{\vec{\mathbf{i}}} = \hat{A}_{\vec{\mathbf{i}}} \cdot \hat{B}_{\vec{\mathbf{i}}} \\ \vec{\mathbf{h}} \in \chi^{s+t}(\vec{\mathbf{i}}), &\quad Z_{\vec{\mathbf{h}}} \leftarrow \hat{Z}_{\vec{\mathbf{i}}} \end{aligned}$$

  where $\chi^k(\vec{\mathbf{i}})$ is the set of all $\binom{\omega}{k}$ combinations of k elements in $\vec{\mathbf{i}}$
- $\mathbf{C} = \mathbf{Z} - \ldots$ can then be computed with $O(n^{\omega-1})$ multiplications

Symmetry preserving algorithm costs

- Let $\Upsilon^{(s,t,v)}$ be the nonsymmetric contraction algorithm
- Let $\Psi^{(s,t,v)}$ be the direct evaluation algorithm
- Let $\Phi^{(s,t,v)}$ be the symmetry preserving algorithm

Symmetry preserving algorithm costs

- Let $\Upsilon^{(s,t,v)}$ be the nonsymmetric contraction algorithm
- Let $\Psi^{(s,t,v)}$ be the direct evaluation algorithm
- Let $\Phi^{(s,t,v)}$ be the symmetry preserving algorithm

ω	s	t	v	F_Υ	F_Ψ	F_Φ	application cases
s+t+v	s	t	v	n^ω	$\binom{n}{s}\binom{n}{t}\binom{n}{v}$	$\binom{n}{\omega}$	generally
2	0	0	2	n^2	$n^2/2$	$n^2/2$	Frobenius norm of sym. mat.
2	1	0	1	n^2	n^2	$n^2/2$	symv, hemv, (symm, hemm)
2	1	1	0	n^2	n^2	$n^2/2$	syr2, her2, (syr2k, her2k)
3	1	1	1	n^3	n^3	$n^3/6$	matrix (anti)commutator

where F_X is the number of multiplications computed by algorithm X

Antisymmetry and matrix powers

The symmetry preserving algorithm can compute

- symmetrized products of two symmetric or two antisymmetric tensors

Antisymmetry and matrix powers

The symmetry preserving algorithm can compute

- symmetrized products of two symmetric or two antisymmetric tensors
- antisymmetrized products of a symmetric and an antisymmetric tensor

Antisymmetry and matrix powers

The symmetry preserving algorithm can compute

- symmetrized products of two symmetric or two antisymmetric tensors
- antisymmetrized products of a symmetric and an antisymmetric tensor
- Hermitian tensor contractions

Antisymmetry and matrix powers

The symmetry preserving algorithm can compute

- symmetrized products of two symmetric or two antisymmetric tensors
- antisymmetrized products of a symmetric and an antisymmetric tensor
- Hermitian tensor contractions
- $\mathbf{A}^2$ for symmetric or antisymmetric $\mathbf{A}$ with $n^3/6$ multiplications

Antisymmetry and matrix powers

The symmetry preserving algorithm can compute

- symmetrized products of two symmetric or two antisymmetric tensors
- antisymmetrized products of a symmetric and an antisymmetric tensor
- Hermitian tensor contractions
- $\mathbf{A}^2$ for symmetric or antisymmetric $\mathbf{A}$ with $n^3/6$ multiplications
- $\mathbf{A}^2$ for nonsymmetric $\mathbf{A}$ (or $\mathbf{A} \cdot \mathbf{B} + \mathbf{B} \cdot \mathbf{A}$ for nonsymmetric $\mathbf{A}$, $\mathbf{B}$) with $2n^3/3$ multiplications

Antisymmetry and matrix powers

The symmetry preserving algorithm can compute

- symmetrized products of two symmetric or two antisymmetric tensors
- antisymmetrized products of a symmetric and an antisymmetric tensor
- Hermitian tensor contractions
- $\mathbf{A}^2$ for symmetric or antisymmetric $\mathbf{A}$ with $n^3/6$ multiplications
- $\mathbf{A}^2$ for nonsymmetric $\mathbf{A}$ (or $\mathbf{A} \cdot \mathbf{B} + \mathbf{B} \cdot \mathbf{A}$ for nonsymmetric $\mathbf{A}$, $\mathbf{B}$) with $2n^3/3$ multiplications
- that CCSD contraction,

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

in n^6 operations (2X fewer) via $\Phi^{(1,0,1)} \otimes \Upsilon^{(1,2,1)}$

Bilinear algorithms

A bilinear algorithm is defined by three matrices: $\mathbf{F^{(A)}}$, $\mathbf{F^{(B)}}$, $\mathbf{F^{(C)}}$

Bilinear algorithms

A bilinear algorithm is defined by three matrices: $\mathbf{F}^{(\mathbf{A})}$, $\mathbf{F}^{(\mathbf{B})}$, $\mathbf{F}^{(\mathbf{C})}$
Given input vectors **a** and **b**, it computes vector,

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

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

Bilinear algorithms

A bilinear algorithm is defined by three matrices: $\mathbf{F^{(A)}}$, $\mathbf{F^{(B)}}$, $\mathbf{F^{(C)}}$
Given input vectors **a** and **b**, it computes vector,

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

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

- the number of columns in the three matrices is equal and is the *bilinear algorithm rank*

Bilinear algorithms

A bilinear algorithm is defined by three matrices: $\mathbf{F}^{(\mathbf{A})}$, $\mathbf{F}^{(\mathbf{B})}$, $\mathbf{F}^{(\mathbf{C})}$
Given input vectors **a** and **b**, it computes vector,

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

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

- the number of columns in the three matrices is equal and is the *bilinear algorithm rank*
- the number of rows in each matrix corresponds to the number of inputs (dimensions of **a** and **b**) and outputs (dimension of **c**)

Bilinear algorithms

A bilinear algorithm is defined by three matrices: $\mathbf{F}^{(\mathbf{A})}$, $\mathbf{F}^{(\mathbf{B})}$, $\mathbf{F}^{(\mathbf{C})}$
Given input vectors **a** and **b**, it computes vector,

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

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

- the number of columns in the three matrices is equal and is the *bilinear algorithm rank*
- the number of rows in each matrix corresponds to the number of inputs (dimensions of **a** and **b**) and outputs (dimension of **c**)
- matrix multiplication and symmetric tensor contraction correspond to different bilinear algorithms (problems)

Bilinear algorithms

A bilinear algorithm is defined by three matrices: $\mathbf{F^{(A)}}$, $\mathbf{F^{(B)}}$, $\mathbf{F^{(C)}}$
Given input vectors **a** and **b**, it computes vector,

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

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

- the number of columns in the three matrices is equal and is the *bilinear algorithm rank*
- the number of rows in each matrix corresponds to the number of inputs (dimensions of **a** and **b**) and outputs (dimension of **c**)
- matrix multiplication and symmetric tensor contraction correspond to different bilinear algorithms (problems)
- the bilinear rank is the number of multiplications, for the symmetry preserving algorithm, it is $\binom{n}{\omega}$

Symmetry preserving algorithm as a bilinear algorithm

The bilinear algorithm

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

for computing $\mathbf{Z}$ (as $\mathbf{c}$) is encoded as follows

$$\begin{array}{lll}
\vec{\mathbf{j}} \in \chi^{s+v}(\vec{\mathbf{i}}), & \hat{A}_{\vec{\mathbf{i}}} \leftarrow A_{\vec{\mathbf{j}}} & \hat{\mathbf{a}} = \mathbf{F}^{(\mathbf{A})\mathsf{T}}\mathbf{a} \\
\vec{\mathbf{l}} \in \chi^{v+t}(\vec{\mathbf{i}}), & \hat{B}_{\vec{\mathbf{i}}} \leftarrow B_{\vec{\mathbf{l}}} & \hat{\mathbf{b}} = \mathbf{F}^{(\mathbf{B})\mathsf{T}}\mathbf{b} \\
 & \hat{Z}_{\vec{\mathbf{i}}} = \hat{A}_{\vec{\mathbf{i}}} \cdot \hat{B}_{\vec{\mathbf{i}}} & \hat{\mathbf{z}} = \hat{\mathbf{a}} \circ \hat{\mathbf{b}} \\
\vec{\mathbf{h}} \in \chi^{s+t}(\vec{\mathbf{i}}), & Z_{\vec{\mathbf{h}}} \leftarrow \hat{Z}_{\vec{\mathbf{i}}} & \mathbf{c} = \mathbf{F}^{(\mathbf{C})}\hat{\mathbf{z}}
\end{array}$$

Expansion in bilinear algorithms

Given $\Lambda = (\mathbf{F^{(A)}}, \mathbf{F^{(B)}}, \mathbf{F^{(C)}})$, we say $\Lambda_{\text{sub}} \subseteq \Lambda$ if there exists projection matrix $\mathbf{P}$ such that,

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

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

Expansion in bilinear algorithms

Given $\Lambda = (\mathbf{F}^{(\mathbf{A})}, \mathbf{F}^{(\mathbf{B})}, \mathbf{F}^{(\mathbf{C})})$, we say $\Lambda_{\text{sub}} \subseteq \Lambda$ if there exists projection matrix $\mathbf{P}$ such that,

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

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

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

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

we have

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

Vertical communication in bilinear algorithms

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

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

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,

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

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,

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

further, we have

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

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,

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

further, we have

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

so we obtain the expected bound,

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

Horizontal communication in bilinear algorithms

Any load balanced schedule on a parallel machine with p processes of $\Lambda = (\mathbf{F^{(A)}}, \mathbf{F^{(B)}}, \mathbf{F^{(C)}})$ 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,

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

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 $t = 0$.

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,

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

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

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

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 $\kappa := \max(s+v, v+t, s+t)$,

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

Conclusion

Summary:

- Symmetry preserving algorithms lower the number of multiplications necessary for symmetric tensor contractions
- Reducing the number of multiplications, reduces bilinear rank, and leads to overall cost improvements for nested algorithms
- However, the communication cost requirements of symmetry preserving algorithms are larger in certain cases

Conclusion

Summary:

- Symmetry preserving algorithms lower the number of multiplications necessary for symmetric tensor contractions
- Reducing the number of multiplications, reduces bilinear rank, and leads to overall cost improvements for nested algorithms
- However, the communication cost requirements of symmetry preserving algorithms are larger in certain cases

Future work:

- communication lower bounds for nested algorithms (partially symmetric contractions)
- full derivation of cost improvements for applications, in particular coupled cluster methods
- high performance implementation

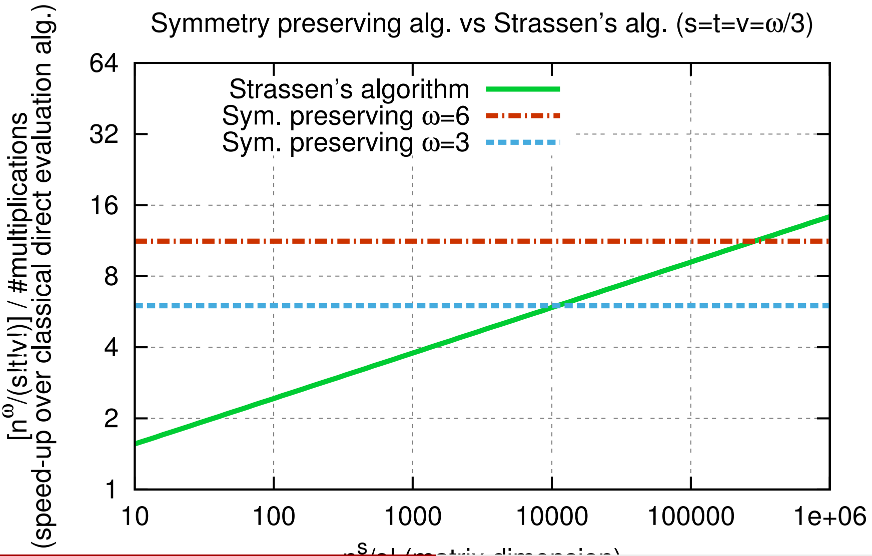
Further references

For more information see

- ES and James Demmel; Contracting symmetric tensors using fewer multiplications
- ES, James Demmel, and Torsten Hoefler; Communication lower bounds for tensor contraction algorithms

Backup slides

Symmetry preserving algorithm vs Strassen's algorithm



Nesting of bilinear algorithms

Given two bilinear algorithms:

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

We can nest them by computing their tensor product

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

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, $\mathcal{E}_{12}(c^{(A)}, c^{(B)}, c^{(C)})$

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

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 $\mathbf{A} \otimes \mathbf{B}$ has rank at least $k^{\min(\alpha,\beta)}$

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 $\mathbf{A}$ and $\mathbf{B}$, such that for some $\alpha, \beta \in [0, 1]$ and any $k \in \mathbb{N}$

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

then any subset of $k \in \mathbb{N}$ columns of $\mathbf{A} \otimes \mathbf{B}$ 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.