Communication Lower Bounds for Bilinear Algorithms

Edgar Solomonik

L P N A @CS@Illinois

Department of Computer Science
University of Illinois at Urbana-Champaign

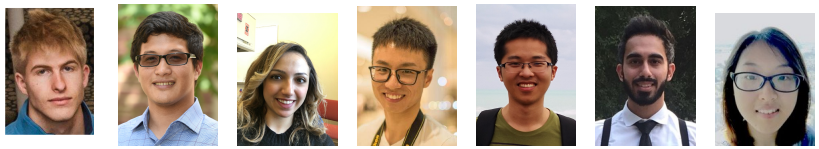
SIAM Annual Meeting

Laboratory for Parallel Numerical Algorithms

Group research foci

- tensor computations
- numerical linear algebra
- numerical optimization
- communication cost analysis*
- paralallel algorithms
- quantum simulation
- graph algorithms

L P N A @CS@Illinois



DOE CSGF

NSF

http://lpna.cs.illinois.edu

Overview

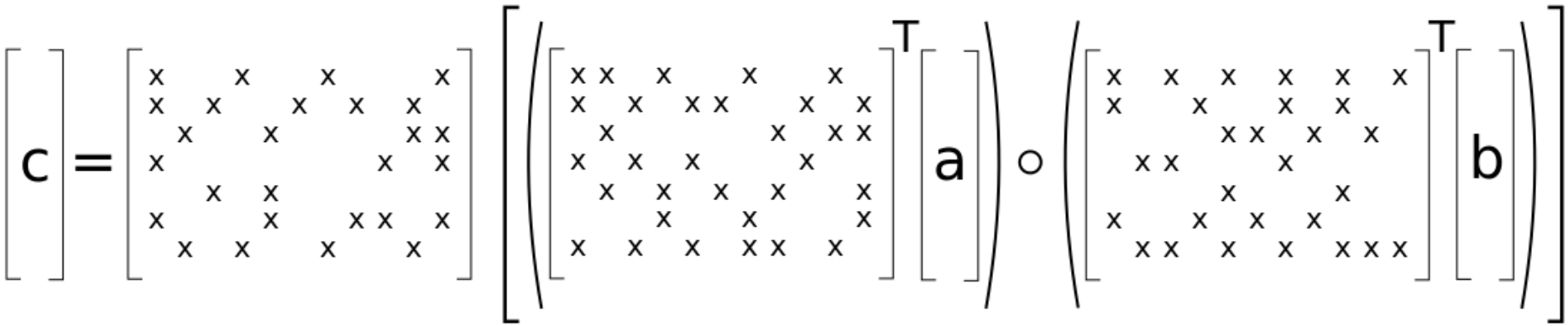
- General framework for communication lower bounds based on rank structure of bilinear algorithm
  - *Communication lower bounds of bilinear algorithms for symmetric tensor contractions*
    E.S., James Demmel, and Torsten Hoefler (SISC 2021, arXiv:1707.04618)
- Communication lower bounds for nested bilinear algorithms
  - *Communication lower bounds for nested bilinear algorithms*
    Caleb Ju, Yifan Zhang, and E.S. (arXiv:2107.09834)


Bilinear Algorithms

A bilinear algorithm[1] $\Lambda = (\boldsymbol{F}^{(A)}, \boldsymbol{F}^{(B)}, \boldsymbol{F}^{(C)})$ computes

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

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are inputs and $\odot$ is the Hadamard (pointwise) product



[1] Victor Pan, How to multiply matrices faster. 1984.

Example: Convolution Bilinear Algorithm

Given $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$, compute

$$c_i = \sum_k a_k b_{i-k}$$

- Equivalent to product of polynomials, compute via interpolation

$$\boldsymbol{c} = \boldsymbol{V}^{-1}[(\boldsymbol{V}^T\boldsymbol{a}) \odot (\boldsymbol{V}^T\boldsymbol{b})]$$

  where $\boldsymbol{V}$ is a Vandermonde matrix
- Known as Toom-Cook algorithm; with complex roots of unity as nodes, $\boldsymbol{V}$ is DFT matrix, enabling use of FFT algorithm
- Other bilinear algorithms and problem variants possible[1]

[1] Caleb Ju and E.S., Derivation and analysis of fast bilinear algorithms for convolution. SIAM Review 2020.

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 example, the convolution problem is defined by

$$t_{ijk} = 1 \text{ iff } i = j + k$$

so that

$$c_i = \sum_{j,k} t_{ijk} a_j b_k = \sum_k a_k b_{i-k}$$

Bilinear Problems

- convolution of n-dimensional vectors
  - optimal rank[1]: $R = \Theta(n)$
- matrix multiplication of $n \times n$ matrices
  - best currently known algorithm[2]: $R = O(n^{2.37286})$
- symmetric tensor contraction with $n \times \cdots \times n$ tensors
  - for symmetry preserving algorithms[3], e.g., $\boldsymbol{AB} + \boldsymbol{BA}$ via

$$c_{ij} = \sum_k (a_{ij} + a_{ik} + a_{jk})(b_{ij} + b_{ik} + b_{jk}) - \ldots$$

$R = \frac{n^d}{d!} + O(n^{d-1})$, where d is total number of indices in contraction

[1] Ke Ye and Lek-Heng Lim, FoCM 2018.

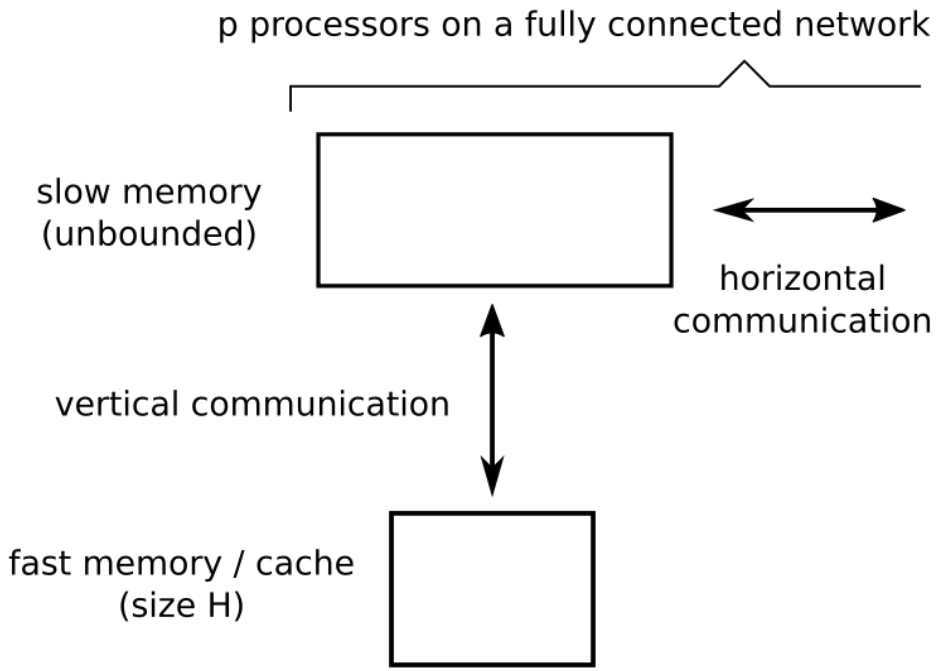
[2] Josh Alman and V. V. Williams, A refined laser method and faster matrix multiplication. SODA 2020.

[3] E.S. and James Demmel, Fast bilinear algorithms for symmetric tensor contractions. CMAM 2021.

Communication Lower Bounds for Bilinear Algorithms

How much communication does a bilinear algorithm require?

- bilinear algorithm specifies only products, not order of additions, need to consider various partial sums
- measure two types of communication:
  - vertical – words communicated between memory and cache of size H
  - horizontal – words communicated by some processor of p
- assume load-balanced input/output and no replication of work



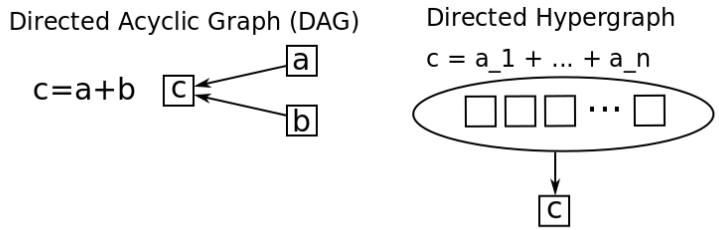
Execution DAG

Model computation by dependency graph (execution DAG)

- vertex – input/output/computed value
- edge – input to operation computing a value
- parallelization – graph partitioning
- cache-blocking – pebbling game on DAG

Communication lower bounds characterize minimal comm. cost of one or a family of execution DAG

- various orderings of a sum permitted by expressing it as a hyperedge[1]
- columns of bilinear algorithm matrices encode such hyperedges



[1] E.S., E. Carson, N. Knight, Trade-offs between synchronization, communication, and computation in parallel linear algebra computations. TOPC 2016.

Some Known Lower Bounds

Matrix multiplication of $n \times n$ matrices

- classical $O(n^3)$ algorithm requires $Q = \Theta(n^3/\sqrt{H})$ vertical[1] and $W = \Theta(n/p^{2/3})$ horizontal communication[2]
- Strassen's $O(n^{\log_2 7})$ algorithm requires $Q = \Theta(n^{\log_2 7} H/H^{\log_4 7})$ vertical and $W = \Theta(n^2/p^{1/\log_4 7})$ horizontal communication[3]
- bounds for many numerical linear algebra algorithms (LU, QR, SVD, eigensolvers) depend on above[4]

Convolution of n-dimensional vectors

- FFT requires $Q = \Theta(n \log n/\log H)$ vertical[1] and $W = \Theta(n \log n/\log(n/p))$ horizontal comm.[5]

Given computational complexity $f(n)$, these bounds share a common form

$$Q = \Theta(f(n) \cdot H/\tilde{f}(H)) \qquad W = \Theta(\tilde{f}^{-1}(f(n)/p))$$

---

[1] Jia-Wei Hong and H.T. Kung, I/O complexity: The red-blue pebble game. STOC 1981

[2] D. Irony, S. Toledo, A. Tiskin, Communication lower bounds for distributed-memory matrix multiplication. JPDC 2004.

[3] G. Ballard, J. Demmel, O. Holtz, O. Schwartz, Graph expansion and comm. costs of fast mat. mult., JACM 2013.

[4] G. Ballard, J. Demmel, O. Holtz, O. Schwartz. Minimizing communication in numerical linear algebra. SIMAX 2011.

[5] A. Aggarwal, A.K. Chandra, M. Snir, Communication complexity of PRAMs, 1990.

Lower Bounds for Bilinear Algorithms

Communication lower bounds based on the rank R of the bilinear algorithm

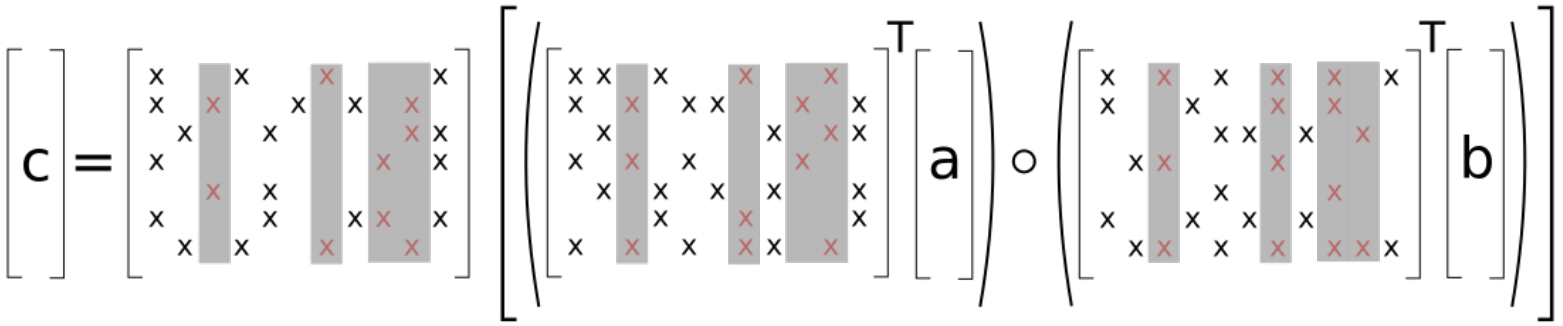
- lower bound information needed to compute any subset of the R products
  - consider inputs and outputs symmetrically
  - products are columns in bilinear matrices, so seek lower bound on rank of submatrices
- above expansion bound $\tilde{f}$ yields horiz. and vertical comm. lower bounds
  - lower bound on amount of useful work with $O(H)$ input/output words yields vertical communication lower bound

$$Q = \Omega(R \cdot H / \tilde{f}(H))$$

  - some processor computes at least R/p products, use $\tilde{f}$ to bound information it must send/receive

$$W = \Omega(\tilde{f}^{-1}(R/p) - (\#\text{inputs} + \#\text{outputs})/p)$$

Bilinear Algorithm Expansion Bound



- generally, for bilinear algorithm Λ seek function $\mathcal{E}_\Lambda(x, y, z)$, s.t., for any subset of $k \leq R$ products

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

  where $\boldsymbol{F}^{(\boldsymbol{A})}_{\text{sub}}$ is the submatrix of $\boldsymbol{F}^{(A)}$ obtained from extracting k columns, and similar for respective columns of $\boldsymbol{F}^{(B)}$ and $\boldsymbol{F}^{(C)}$

- $\mathcal{E}_\Lambda(x, y, z)$ describes the amount of useful work we can do with m information

$$\tilde{f}(m) = \max_{x+y+z \leq m} \mathcal{E}_\Lambda(x, y, z)$$

Rank Expansion of Matrices

- Let $\sigma_{F^{(X)}}(k)$ be the minimum rank of a subset k columns of $\boldsymbol{F}^{(X)}$, then

$$\mathcal{E}_\Lambda(x,y,z) \leq \max_{x+y+z\leq m} \min(\sigma^{-1}_{F^{(A)}}(x), \sigma^{-1}_{F^{(B)}}(y), \sigma^{-1}_{F^{(C)}}(z))$$

- so we may simply consider the rank expansion $\sigma(k)$ of each bilinear algorithm matrix
- stronger bounds are sometimes possible by considering all three matrices simultaneously (since the subset of columns must be matching among the 3 matrices)
  - e.g., for classical matrix multiplication, this is needed to use the surface-to-volume-ratio argument (Loomis-Whitney inequality)
  - in other cases, it suffices to consider σ for each bilinear matrix separately, e.g., compare

$$c_{ij} \leftarrow a_{ik}b_{kj} \quad \text{to} \quad \{c_{ij}, c_{ik}, c_{jk}\} \leftarrow (a_{ij}+a_{ik}+a_{jk})(b_{ij}+b_{ik}+b_{jk})$$

Rank Expansion of Matrices

We are then interested in characterizing by $\sigma(k)$, the minimum rank of a subset of k columns of a given matrix $\boldsymbol{C}$

- $\sigma_C(k) = k$ if $\boldsymbol{C}$ is full rank
- the first kink in σ_C is determined by the Kruskal rank of $\boldsymbol{C}$ (largest k such that any subset of k columns of $\boldsymbol{C}$ are linearly independent)
- if $\boldsymbol{C}$ is sparse (e.g., banded), lower bounds on $\sigma_C(k)$ may be derived based on sparsity
- nested (recursive) bilinear algorithms, (e.g., Strassen's algorithm) result in matrices with Kronecker product structure

$$\boldsymbol{C} = \boldsymbol{A} \otimes \boldsymbol{B}$$

- since $\operatorname{rank}(\boldsymbol{C}) = \operatorname{rank}(\boldsymbol{A}) \operatorname{rank}(\boldsymbol{B})$, can we characterize σ_C in terms of σ_A and σ_B?

Rank Expansion of Kronecker-Product-Structured Matrices

- suppose we select k_A columns of $\boldsymbol{A}$ ($\boldsymbol{A}_1$) and k_B columns of $\boldsymbol{B}$ ($\boldsymbol{B}_1$), then

$$\operatorname{rank}(\boldsymbol{A}_1 \otimes \boldsymbol{B}_1) = \operatorname{rank}(\boldsymbol{A}_1)\operatorname{rank}(\boldsymbol{B}_1)$$

- consequently, we expect that

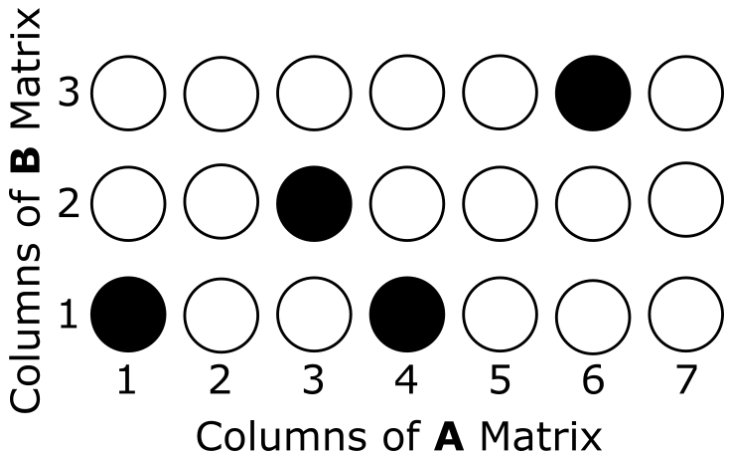
$$\sigma_{A\otimes B}(k) \geq \min_{k_A k_B \geq k} \sigma_A(k_A)\sigma_B(k_B)$$

- we formalize this intuition into a proof and quantify when it holds[1]

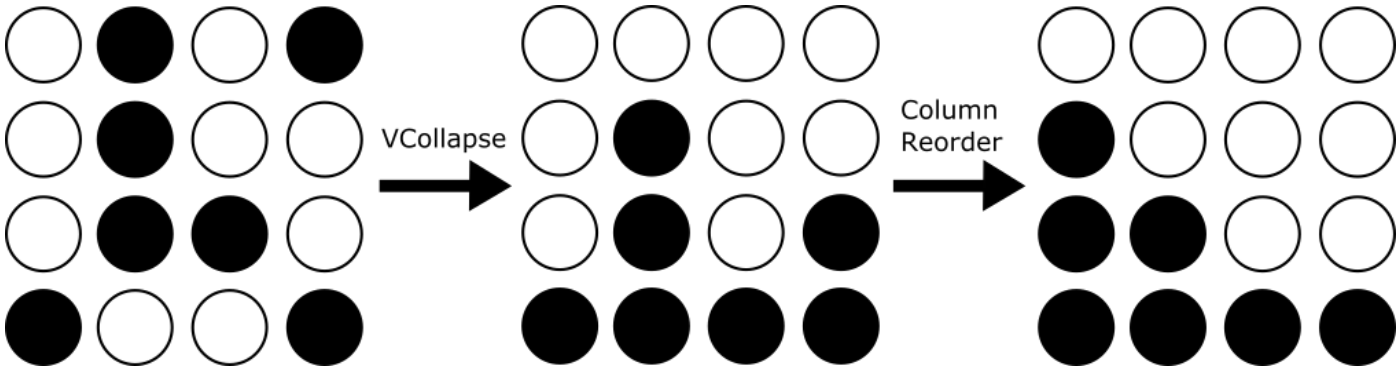

[1] Caleb Ju, Yifan Zhang, and E.S., Communication lower bounds for nested bilinear algorithms. arXiv 2021.

Proof of Rank Expansion Bound under Kronecker Product

We associate a given subset of columns of $\boldsymbol{C}$ with grid points $\boldsymbol{a}_i \otimes \boldsymbol{b}_j$

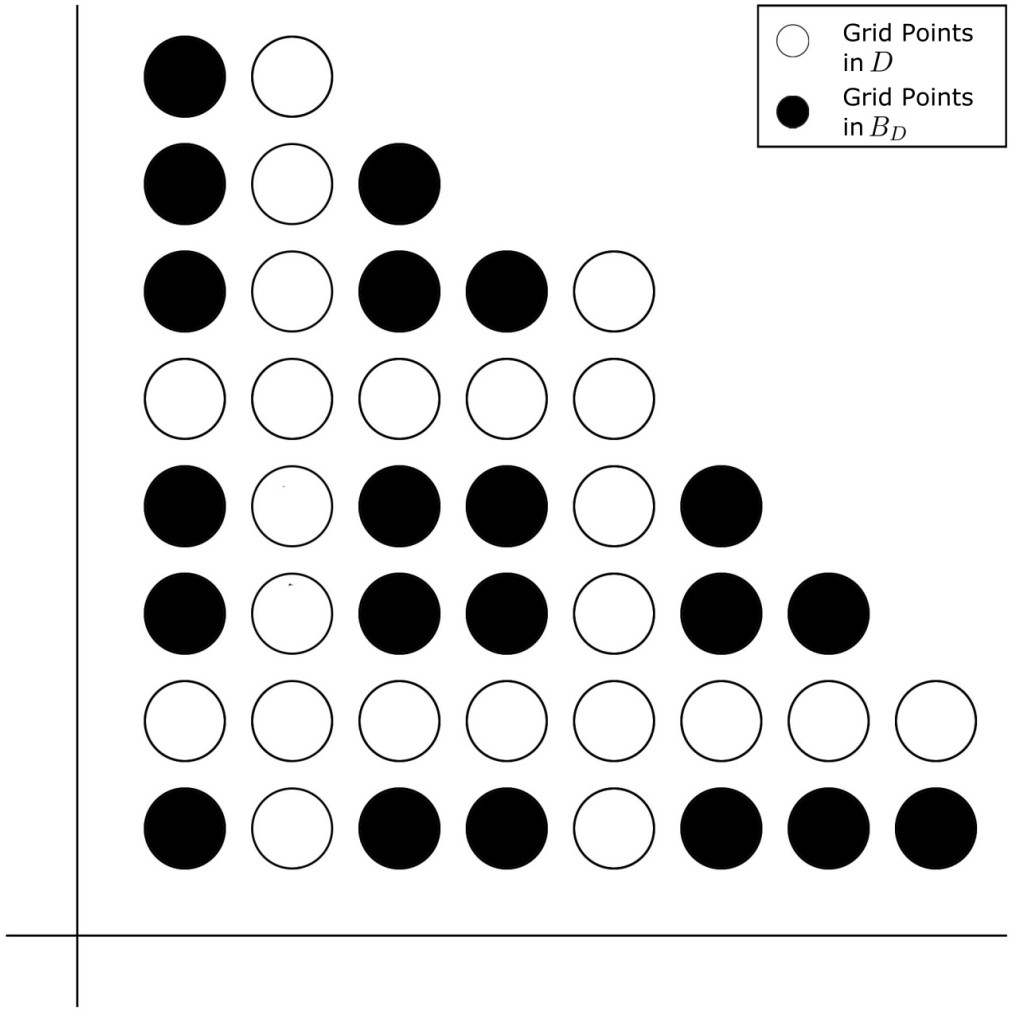


then we transform and shrink the set of grid points in a way that can only decrease rank of the associated submatrix of $\boldsymbol{C}$



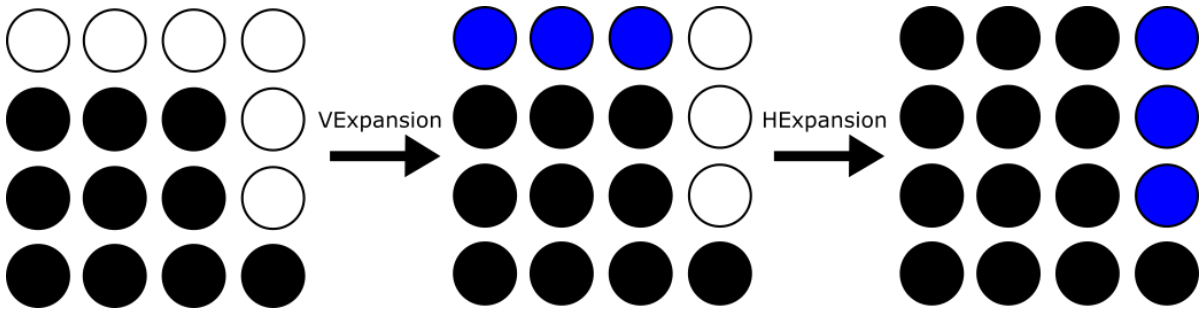
Proof of Rank Expansion Bound under Kronecker Product

We then prove that the compactified grid has a minimal basis (subset of grid points spanning original set) with a Kronecker product structure



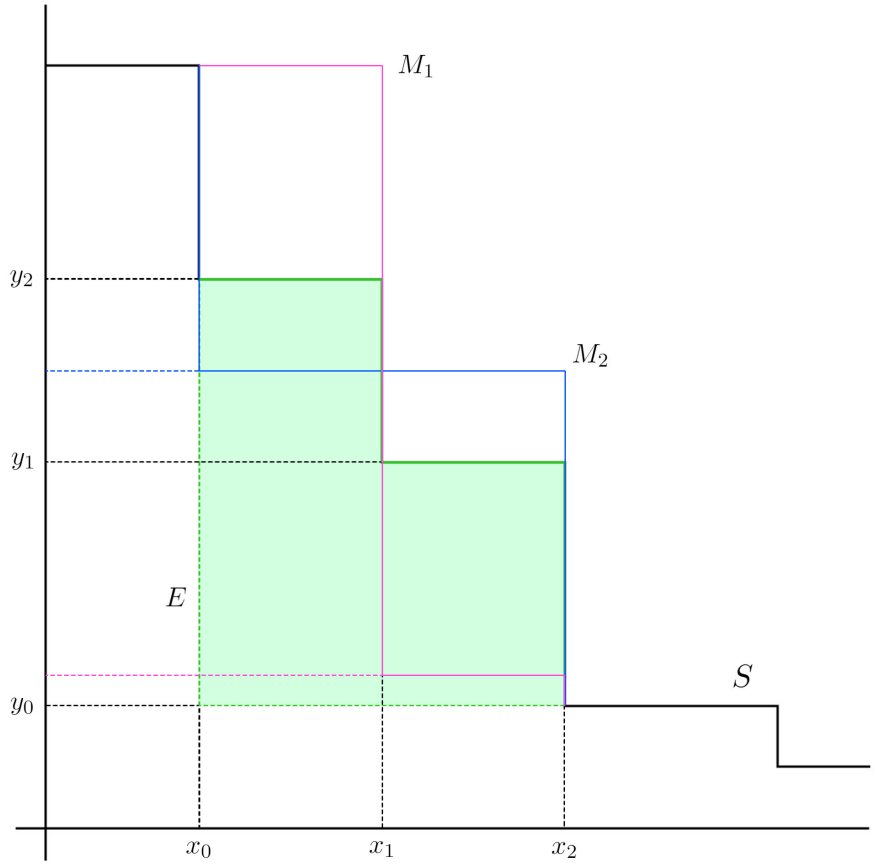
Proof of Rank Expansion Bound under Kronecker Product

Having identified a nicely-structured minimal basis, we can upper bound the total number of columns it can span using σ_A and σ_B



Proof of Rank Expansion Bound under Kronecker Product

Finally, we can transform the shape of the geometric region (set of columns spanned by basis) by removing 'stairs' without decreasing area



the optimal region for general concave σ_A and σ_B is L-shaped

General Kronecker-Product Rank-Expansion Bound

Theorem

Let $f(x) = \sigma_A^{-1}(x)$, $g(x) = \sigma_B^{-1}(x)$. *If*

$$\hat{f}(x) = \frac{f(r_A) - f(r_A - x)}{x f'(r_A - x)}, \ \hat{g}(x) = \frac{g(r_B) - g(r_B - x)}{x g'(r_B - x)},$$

are increasing on $(0, r_A)$ *and* $(0, r_B)$, *respectively, then*

$$\sigma_C(k) = \min_{\substack{k_A \in [d_A, n_A],\ k_B \in [d_B, n_B] \\ k_A \in \{d_A, n_A\} \text{ or } k_B \in \{d_B, n_B\}, \\ k_A k_B \geq k}} \sigma_A(k_A) \cdot \sigma_B(k_B)$$

is a rank expansion lower bound of $\boldsymbol{C} = \boldsymbol{A} \otimes \boldsymbol{B}$.

The conditions are satisfied e.g., when f and g are monomial or exponential

Applications and Conclusion

We apply the nested rank expansion bounds to derive communciation lower bounds for various nested bilinear algorithms

Algorithm	Previous (V)	Previous (H)	Our Work (V)	Our Work (H)
Strassen's	$\frac{n^{\log_2(7)}}{H^{\log_4(7)-1}}$	$\frac{n^2}{p^{\log_7(4)}}$	$\frac{n^{\log_2(7)}}{H^{\log_2(3)-1}}$	$\frac{n^{\log_3(7)}}{p^{\log_3(2)}}$
Recursive convolution	$\frac{n^{\log_k(2k-1)}}{H^{\log_k(2k-1)-1}}$	-	Match Prev	$\frac{n}{p^{\log_{2k-1}(k)}}$

For contraction among symmetric tensors of order $s+v$ and $v+t$ over v modes

- asymptotically more communication is required by symmetry-preserving techniques when s, t, v are unequal
- comm. lower bounds for symmetry preserving algorithms on some contractions of partially-symmetric tensors follow from nested bound

For further details, see arXiv:2107.09834

Backup slides

Symmetric Matrix Vector Product

- Consider computing $\boldsymbol{c} = \boldsymbol{A}\boldsymbol{b}$ with $\boldsymbol{A} = \boldsymbol{A}^T$
  - Typically requires n^2 multiplications since $a_{ij}b_j \neq a_{ji}b_i$ and $n^2 - n$ additions
  - Instead can compute

$$v_i = \sum_{j=1}^{i-1} u_{ij} + \sum_{j=i+1}^{n} u_{ji} \quad \text{where} \quad u_{ij} = a_{ij}(b_i + b_j)$$

    using $n(n-1)/2$ multiplications (since we only need u_{ij} for $i > j$) and about $3n^2/2$ additions, then

$$c_i = (2a_{ii} - \sum_{j=1}^{n} a_{ij})b_i + v_i$$

    using n more multiplications and n^2 additions
  - Beneficial when multiplying elements of $\boldsymbol{A}$ and $\boldsymbol{b}$ costs more than addition
  - This technique yields a bilinear algorithm with rank $n(n+1)/2$

Partially-Symmetric Tensor Times Matrix (TTM)

- Can use symmetric mat-vec algorithm to accelerate TTM with partially symmetric tensor from $2n^4$ operations to $(3/2)n^4 + O(n^3)$
  - Given $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n \times n \times n}$ with symmetry $a_{ijk} = a_{jik}$ and $\boldsymbol{B} \in \mathbb{R}^{n \times n}$, we compute

$$c_{ikl} = \sum_j a_{ijk} b_{jl}$$

  - We can think of this as a set of symmetric matrix-vector products

$$\boldsymbol{c}^{(k,l)} = \boldsymbol{A}^{(k)} \boldsymbol{b}^{(l)}$$

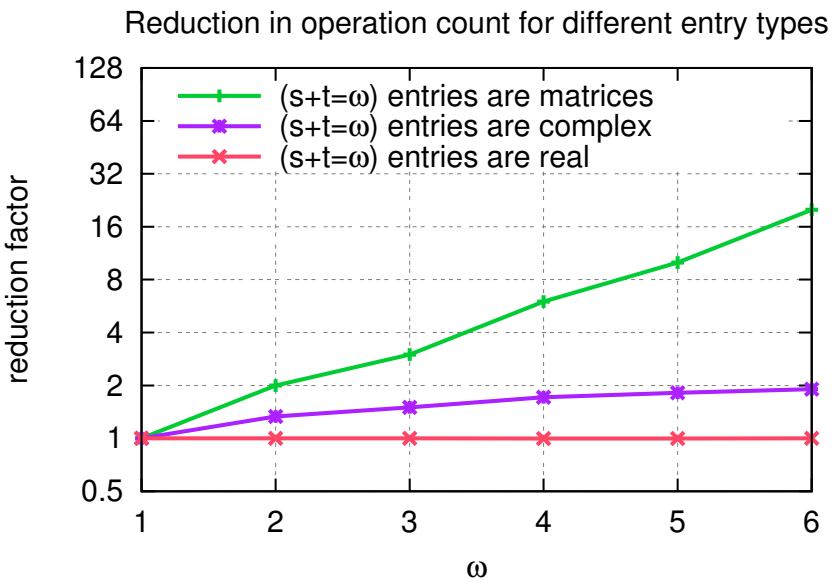
    and apply the fast bilinear algorithm

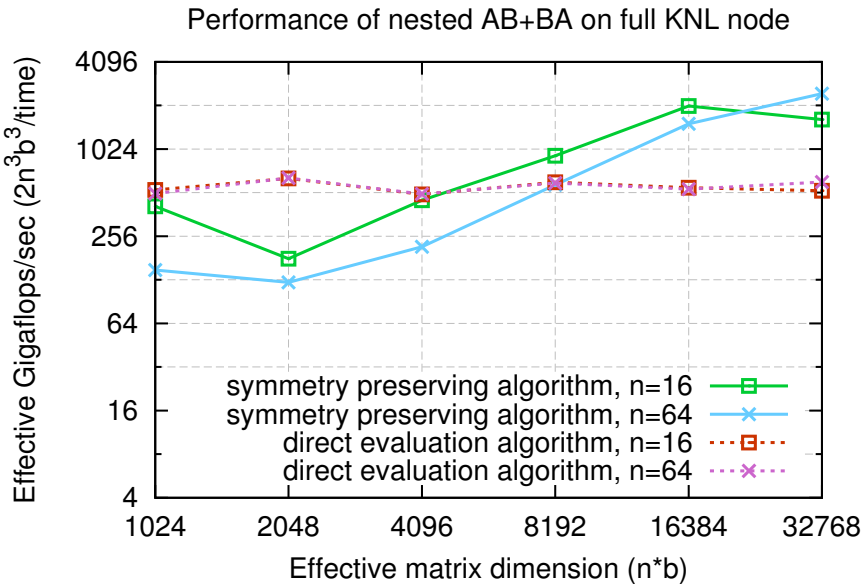
$$v_{ikl} = \sum_{j=1}^{i-1} u_{ijkl} + \sum_{j=i+1}^{n} u_{ijkl} \quad \text{where} \quad u_{ijkl} = a_{ijk}(b_{il} + b_{jl})$$

$$c_{ikl} = (2a_{iik} - \sum_{j=1}^{n} a_{ijk}) b_{il} + v_{ikl}$$

    using about $n^4/2$ multiplications and $n^4 + O(n^3)$ additions (need only n^3 distinct sums of elements of $\boldsymbol{B}$) to compute $\boldsymbol{\mathcal{V}}$, then $O(n^3)$ operations to get $\boldsymbol{\mathcal{C}}$ from $\boldsymbol{\mathcal{V}}$

Permutational Symmetry in Tensor Contractions





New contraction algorithms reduce cost via permutational symmetry[1]

- Symmetry is hard to use in contraction e.g. $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ with $\boldsymbol{A}$ symmetric
- For contraction of order $s+v$ and $v+t$ tensors to produce an order $s+t$ tensor, previously known approaches reduce cost by $s!t!v!$
- New algorithm reduces number of *products* by $\omega!$ where $\omega = s+t+v$, leads to same reduction in *cost* for partially-symmetric contractions

$$\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B} + \boldsymbol{B}\boldsymbol{A} \Rightarrow c_{ij} = \sum_k [(a_{ij} + a_{ik} + a_{jk}) \cdot (b_{ij} + b_{ik} + b_{jk})] - \ldots$$

[1] E.S, J. Demmel, CMAM 2020