Scalable Algorithms for Tensor Computations

Edgar Solomonik

L P N A @CS@Illinois

Department of Computer Science
University of Illinois at Urbana-Champaign

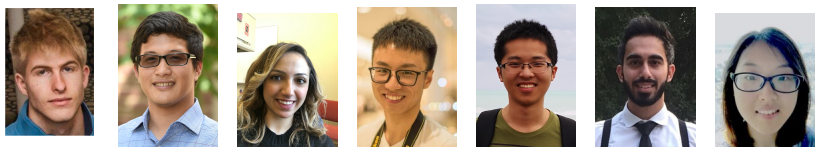
Fast Code Seminar, MIT

Laboratory for Parallel Numerical Algorithms

Recent/ongoing research topics

- parallel matrix computations
  - QR factorization
  - triangular solve
  - eigenvalue problems
- tensor computations
  - tensor decomposition
  - sparse tensor kernels
  - tensor completion
- simulation of quantum systems
  - tensor networks
  - quantum chemistry
  - quantum circuits
- fast bilinear algorithms
  - convolution algorithms
  - tensor symmetry
  - fast matrix multiplication

L P N A @CS@Illinois



DOE CSGF

NSF

http://lpna.cs.illinois.edu

Outline

Library for Massively-Parallel Tensor Computations

Cyclops Tensor Framework[1] sparse/dense generalized tensor algebra

- Cyclops is a C++ library that distributes each tensor over MPI
- Used in chemistry (PySCF, QChem)[2], quantum circuit simulation (IBM/LLNL)[3], and graph analysis (betweenness centrality)[4]
- Summations and contractions specified via Einstein notation

```
E["aixbjy"] += X["aixbjy"]-U["abu"]*V["iju"]*W["xyu"]
```

- Best distributed contraction algorithm selected at runtime via models
- Support for Python (`numpy.ndarray` backend), OpenMP, and GPU
- Simple interface to core ScaLAPACK matrix factorization routines

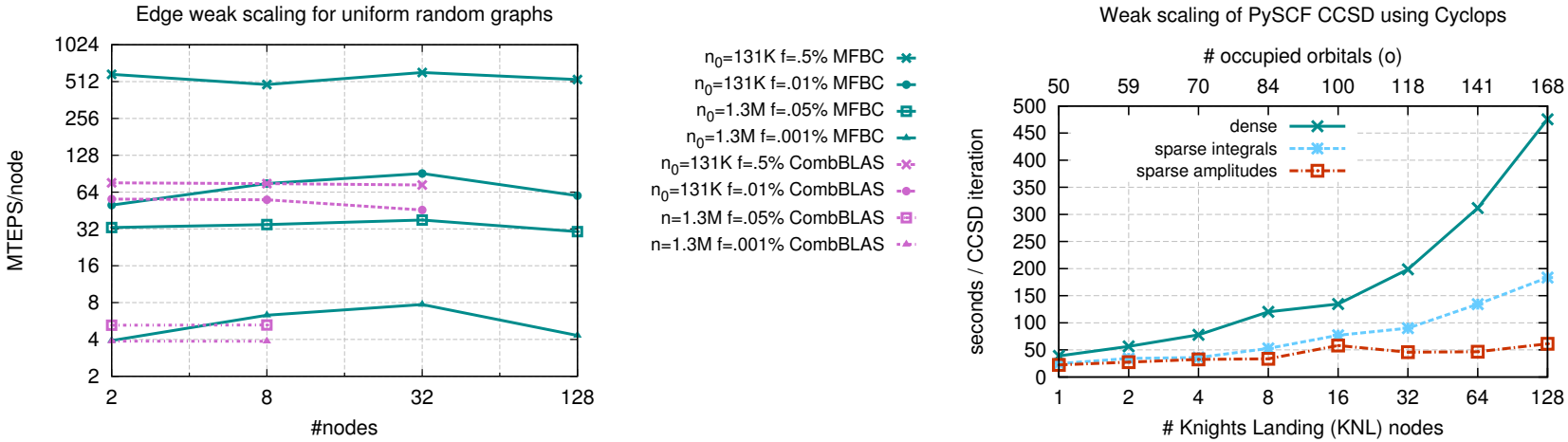
[1] https://github.com/cyclops-community/ctf

[2] E.S., D. Matthews, J. Hammond, J.F. Stanton, J. Demmel, JPDC 2014

[3] E. Pednault, J.A. Gunnels, G. Nannicini, L. Horesh, T. Magerlein, E. S., E. Draeger, E. Holland, and R. Wisnieff, 2017

[4] E.S., M. Besta, F. Vella, T. Hoefler, SC 2017

Sparsity in Tensor Contractions



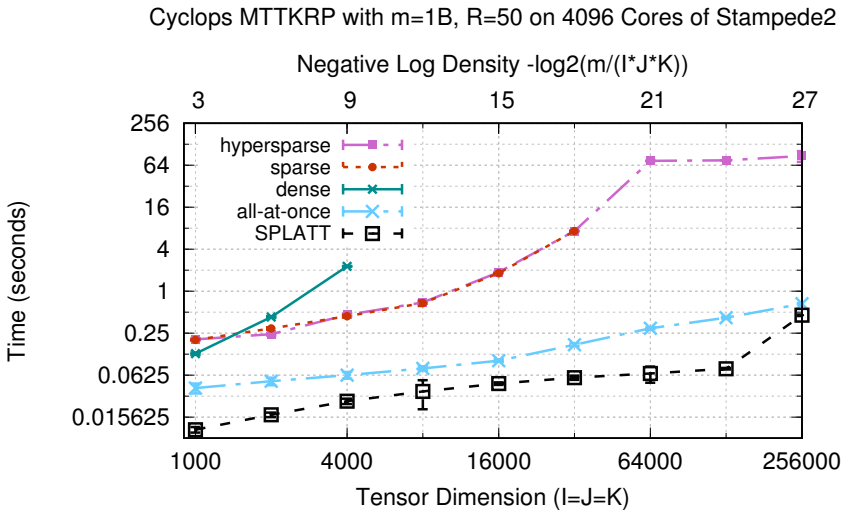
- Cyclops supports sparse representation of tensors[1]
- Choice of representation specified in tensor constructor
- CSR or DCSR[2] (2-index CSF[3]) representation used locally for contractions

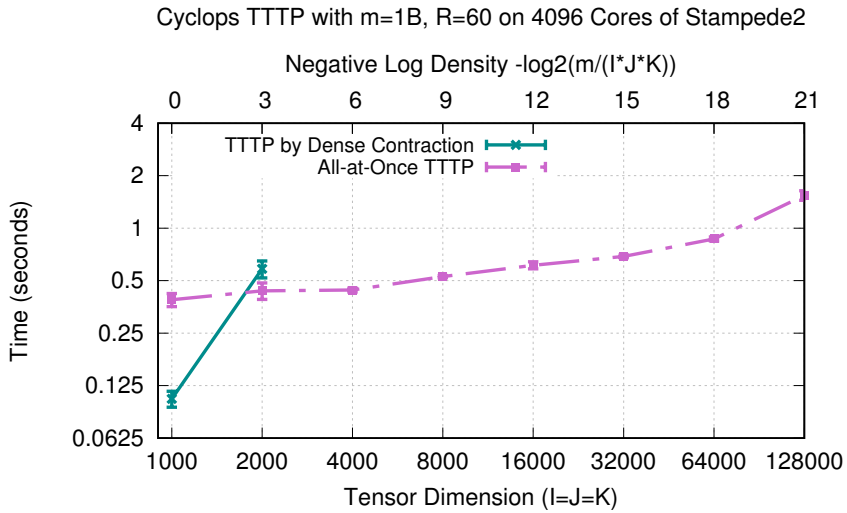
[1] E.S., T. Hoefler 2015

[2] A. Bulúc, J.R. Gilbert, 2008

[3] S. Smith, G. Karypis 2015

All-at-Once Multi-Tensor Contraction





With sparsity, all-at-once contraction[1] of multiple tensors can be faster[2].

- Sparse CP decomposition methods dominated in cost by MTTKRP

$$u_{ir} = \sum_{j,k} t_{ijk} v_{jr} w_{kr}$$

- All-at-once sparse MTTKRP needs less communication than pairwise

- Tensor times tensor product (TTTP) enables sparse residual and CP tensor completion

$$r_{ijk} = \sum_{r} t_{ijk} u_{ir} v_{jr} w_{kr}$$

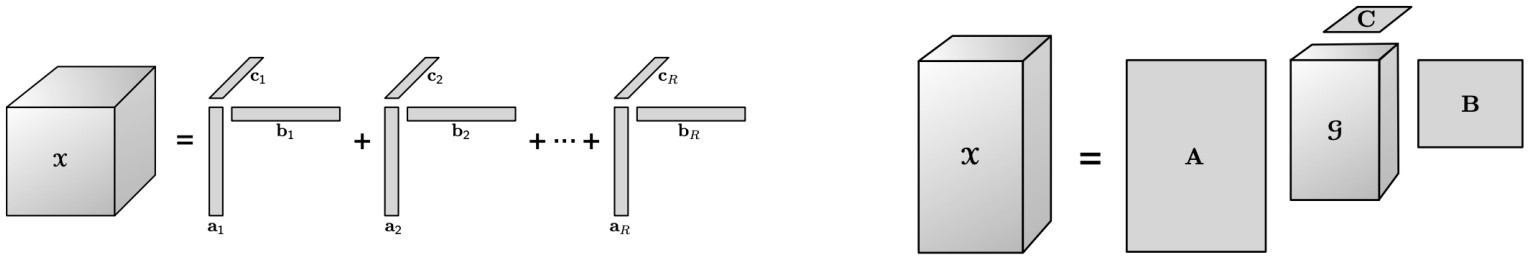
- Cost and memory footprint reduced asymptotically

[1] S. Smith, J. Park, G. Karypis, 2018

[2] Zecheng Zhang, Xiaoxiao Wu, Naijing Zhang, Siyuan Zhang, and E.S. arXiv:1910.02371

CP Tensor Decomposition Algorithms

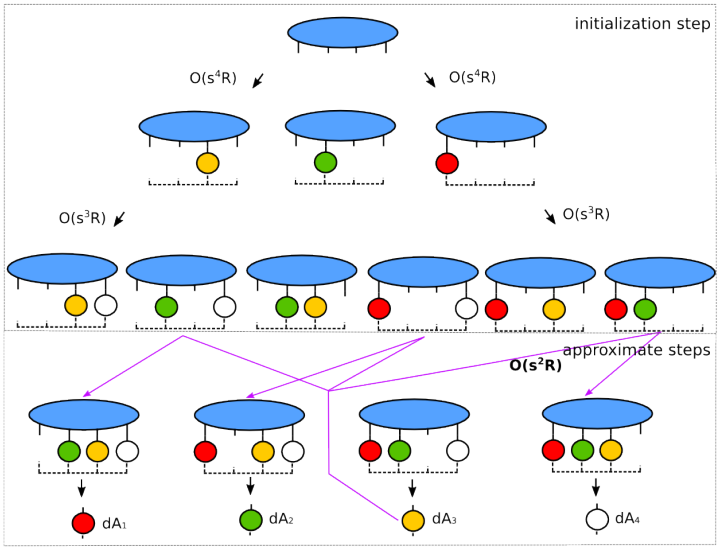
- Tensor of order N has N modes and dimensions $s \times \cdots \times s$
- CP and Tucker tensor decompositions[1]

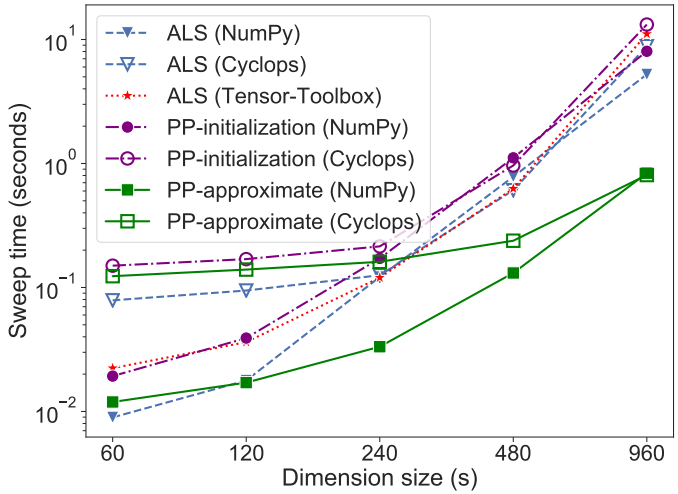


- Alternating least squares (ALS) is most widely used method
  - Optimize one factor matrix at a time, yielding quadratic optimization subproblems
  - Achieves monotonic linear convergence
- Gauss-Newton method is an emerging alternative
  - Optimizes all factor matrices at once by quadratic approximation of nonlinear objective function
  - Non-monotonic, but can achieve quadratic convergence

[1] Kolda and Bader, SIAM Review 2009

Pairwise Perturbation Algorithm





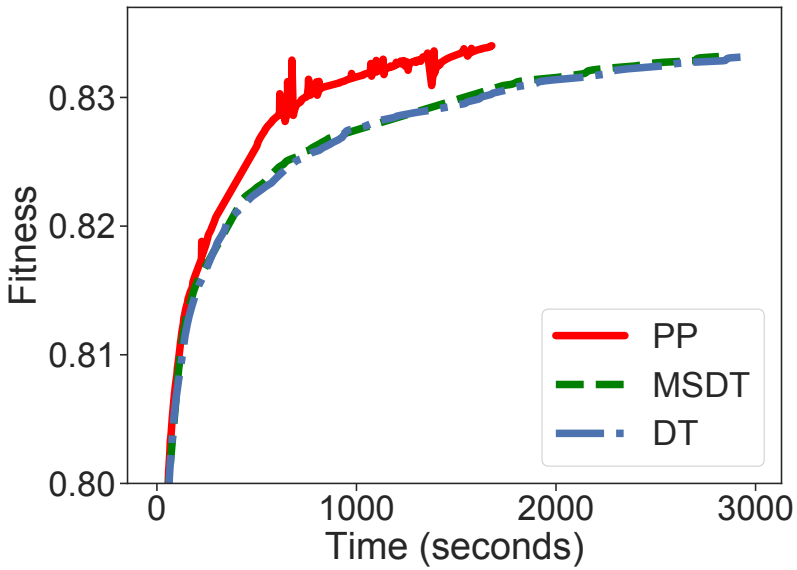
New algorithm: pairwise perturbation (PP)[1] approximates ALS

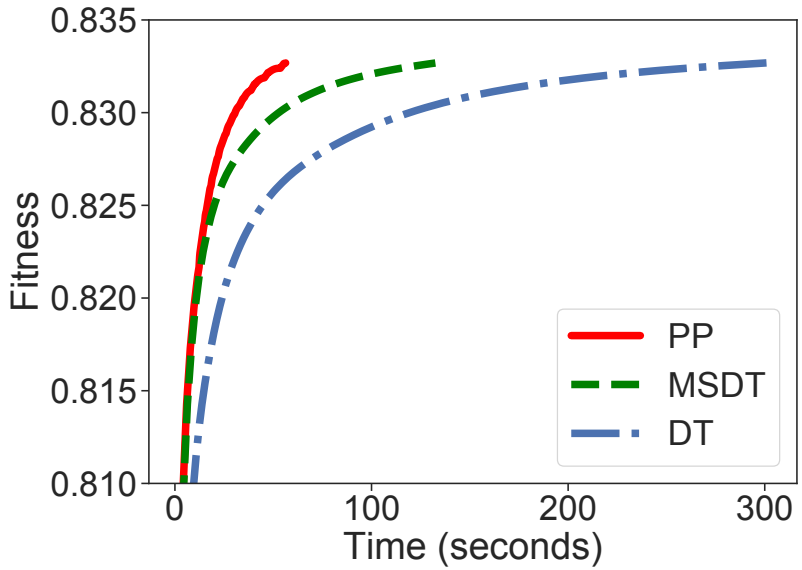
- based on perturbative expansion of ALS update to approximate MTTKRP
- approximation is accurate when ALS updates stagnate
- rank $R < s^{N-1}$ CP decomposition:
  - ALS sweep cost $O(s^N R) \Rightarrow O(s^2 R)$, up to 33x speed-up

Linjian Ma

[1] Linjian Ma, E.S. arXiv:1811.10573

Parallel Pairwise Perturbation Algorithm





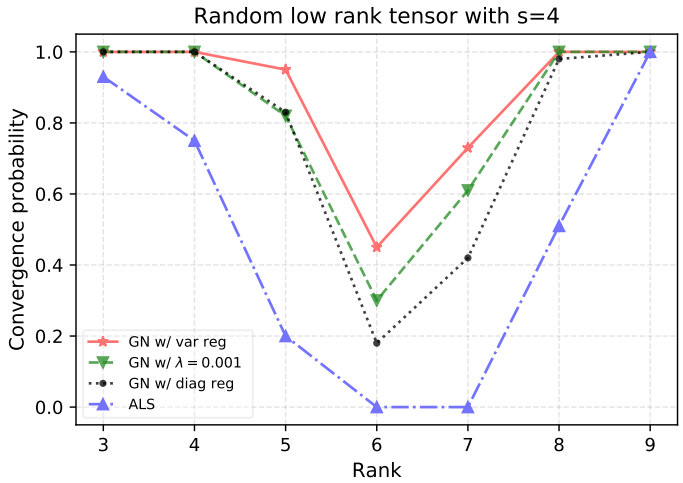
Effective parallelization by decomposing MTTKRP into local MTTKRPs [1]

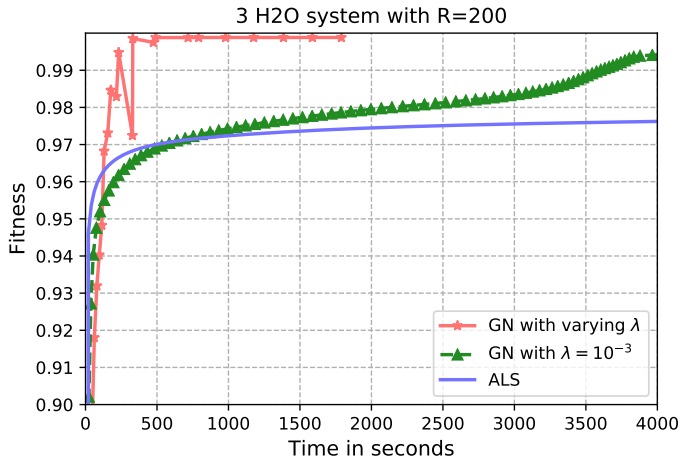
$$\boldsymbol{U} = \text{MTTKRP}(\boldsymbol{\mathcal{T}}, \boldsymbol{V}, \boldsymbol{W}) \Rightarrow \boldsymbol{U}_i = \sum_{j,k} \text{MTTKRP}(\boldsymbol{\mathcal{T}}_{ijk}, \boldsymbol{V}_j, \boldsymbol{W}_k)$$

- processor (i, j, k) owns $\boldsymbol{\mathcal{T}}_{ijk}$, $\boldsymbol{V}_j$, and $\boldsymbol{W}_k$
- pairwise perturbation can be used to approximate local MTTKRPs
- multi-sweep dimension-tree (MSDT) amortizes terms across sweeps

[1] Linjian Ma, E.S. IPDPS 2021

Regularization and Parallelism for Gauss-Newton





New regularization scheme[1] for Gauss-Newton CP with implicit CG[2]

- Oscillates regularization parameter geometrically between lower and upper thresholds
- Achieves higher convergence likelihood
- More accurate than ALS in applications
- Faster than ALS sequentially and in parallel

Navjot Singh

[1] Navjot Singh, Linjian Ma, Hongru Yang, and E.S. arXiv:1910.12331

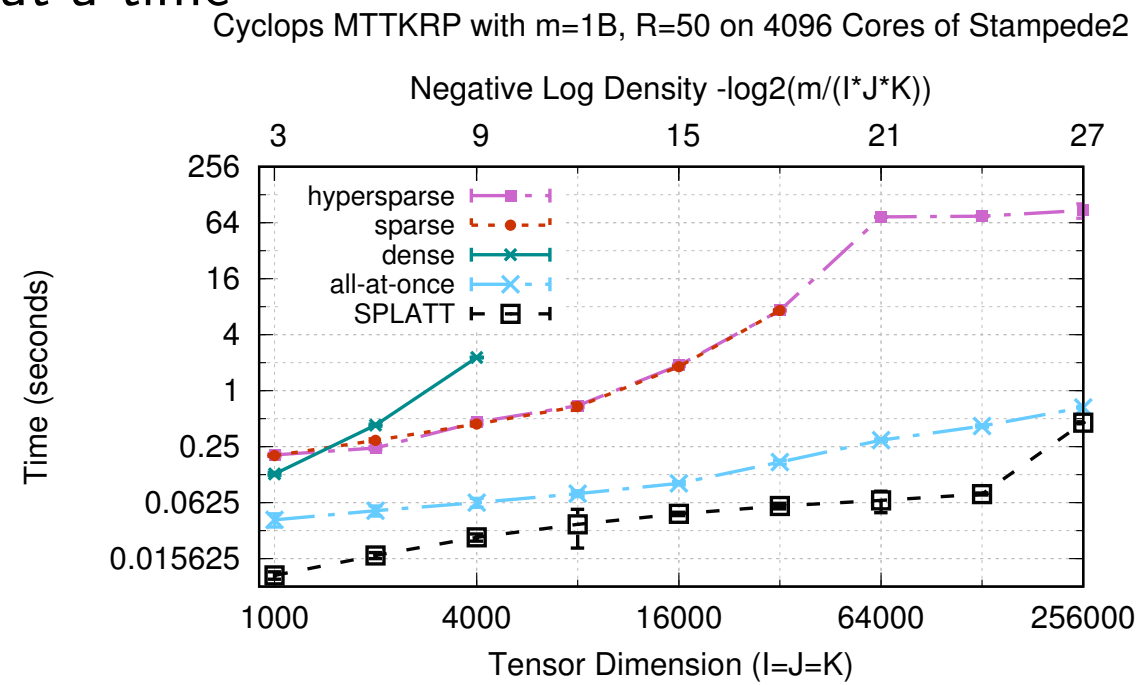
[2] P. Tichavsky, A. H. Phan, and A. Cichocki., 2013

Sparse Tensor Decomposition

- MTTKRP is the most costly operation in spars CP-ALS

$$u_{ir} = \sum_{j,k} t_{ijk} v_{jr} w_{kr}$$

- Sparse MTTKRP can be done faster all-at-once than by contracting two tensors at a time



Randomized Methods for Sparse Tensor Decomposition

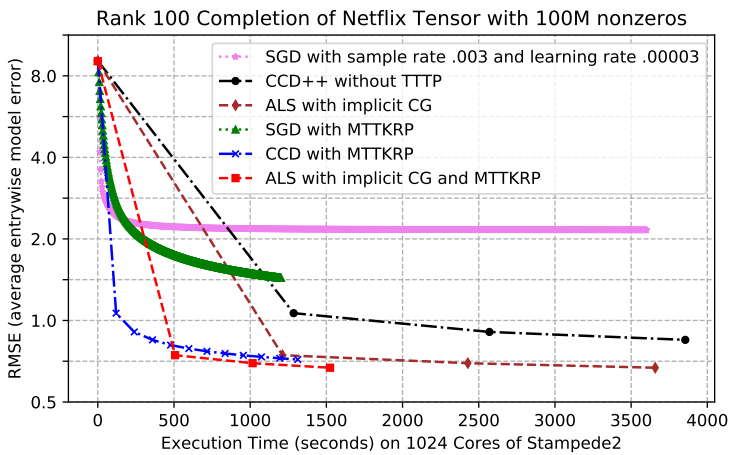
- When seeking a low-rank $R = O(1)$ decomposition for a sparse tensor, sketching schemes have been shown to be efficient
- In this regime, Tucker can be used to construct a CP decomposition
- Leverage score sampling on the rank-constrained least squares problem $\min_{\boldsymbol{X},\mathrm{rank}(\boldsymbol{X})\leq R} \|\boldsymbol{A}\boldsymbol{X} - \boldsymbol{B}\|_F$ leads to a state-of-the-art cost-accuracy trade-off[1] in approximations to Tucker-ALS
- Ideas similar to work by Liu and Moitra (2020) on tensor completion

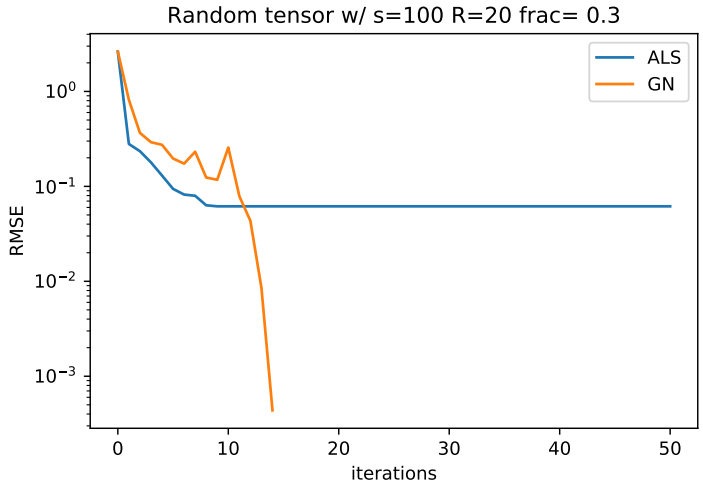
Algorithm for Tucker	LS solve cost	Sample size (m)
ALS	$O(\mathrm{nnz}(\boldsymbol{\mathcal{T}})R^{N-1})$	/
ALS + TensorSketch[2]	$\tilde{O}(mR^N + msR)$	$O(R^{2(N-1)} \cdot 3^{N-1}/(\epsilon^2\delta))$
ALS + TTMTS[2]	$\tilde{O}(msR^{N-1})$	$O(R^{2(N-1)} \cdot 3^{N-1}/(\epsilon^2\delta))$
ALS + TensorSketch[1]	$\tilde{O}(mR^{2N-2} + sR^{N-1})$	$O\big((R^{(N-1)} + 1/\epsilon^2) \cdot (3R)^{(N-1)}/\delta\big)$
ALS + leverage scores[1]	$\tilde{O}(mR^{2N-2} + sR^{N-1})$	$O(R^{(N-1)}/(\epsilon^2\delta))$

[1] Linjian Ma and E.S., in preparation

[2] O. Malik and S. Becker, 2018 (assuming unconstrained LSQ)

Tensor Completion



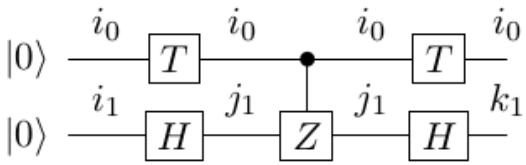


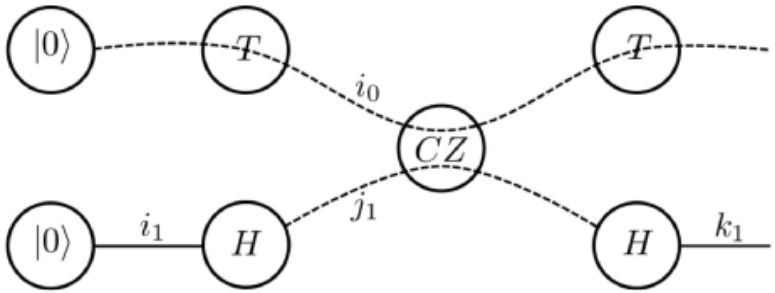
- Via the CTF Python interface, we have implemented SGD, CCD, ALS (with iterative and direct solves), and Gauss-Newton[1]
- Can also handle a variety of loss functions (generalized decomposition)
- All-at-once primitives for MTTKRP, TTTP, and explicit solves in completion ALS drastically improve performance

[1] Navjot Singh, Zecheng Zhang, Xiaoxiao Wu, Naijing Zhang, Siyuan Zhang, and Edgar Solomonik arXiv:1910.02371

Quantum Circuit Simulation with Tensor Networks

- A quantum circuit is a direct description of a tensor network[1]

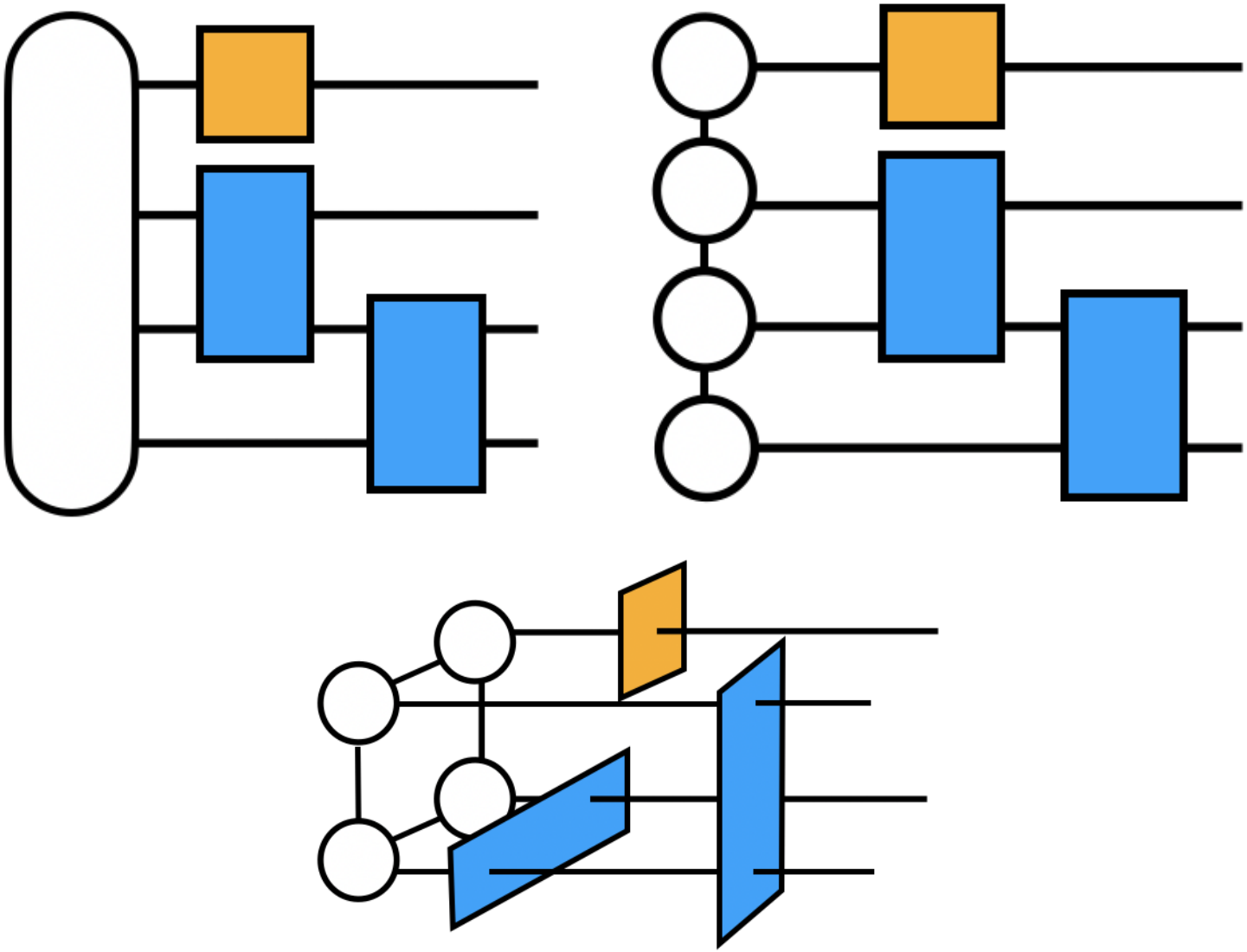




- Why use HPC to (approximately) simulate quantum circuits?
  - enable development/testing/tuning of larger quantum circuits
  - understand approximability of different quantum algorithms
  - quantify sensitivity of algorithms to noise/error
  - potentially enable new hybrid quantum-classical algorithms
- Cyclops utilized to simulate 49-qubit circuits by IBM+LLNL team via direct contraction[2] and by another team from via exact PEPS evolution/contraction[3]

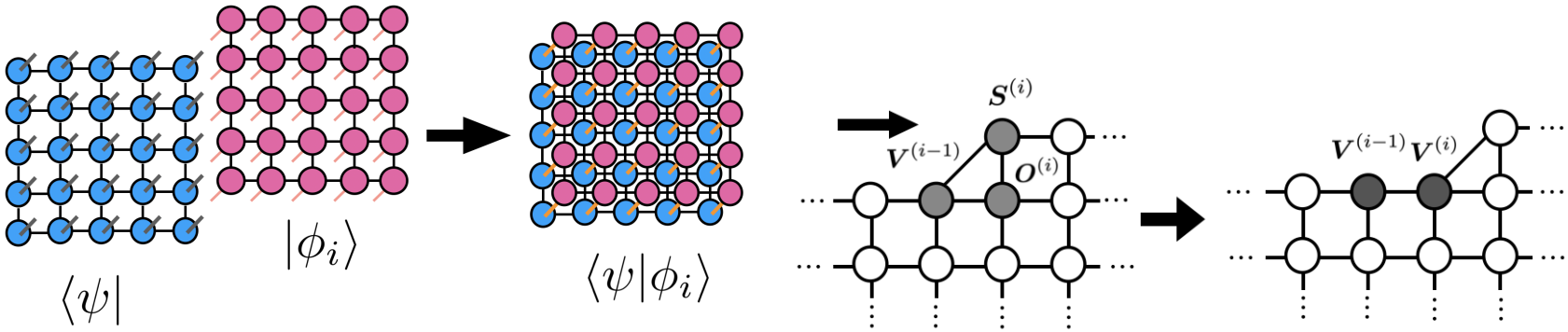
[1] Markov and Shi SIAM JC 2007
[2] Pednault et al. arXiv:1710.05867
[3] Guo et al. Phys Rev Letters, 2019

Tensor Network State Simulation



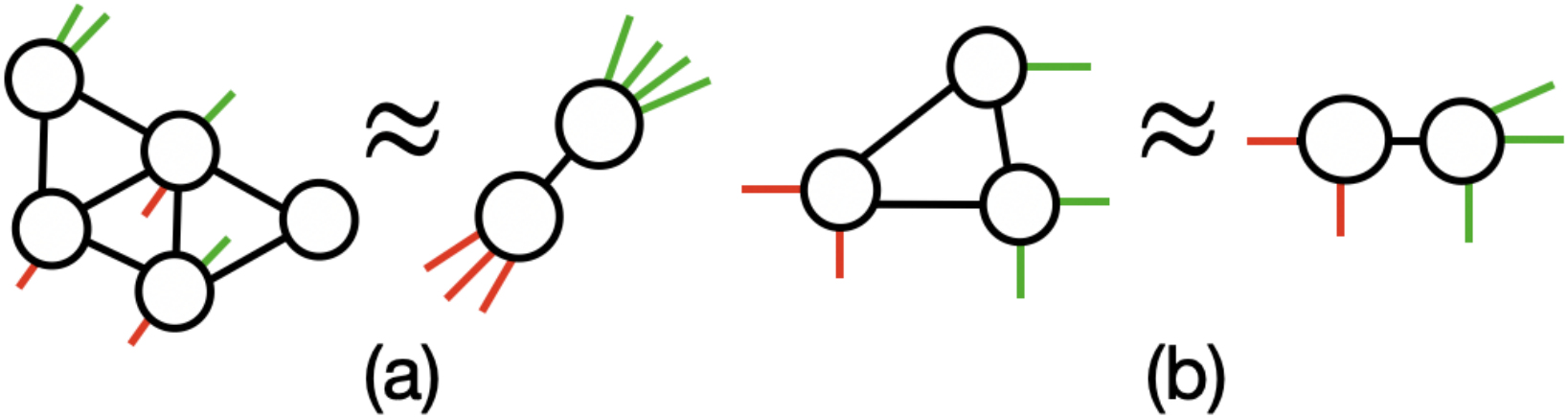
PEPS Contraction

- Exact contraction of PEPS is #P-complete, so known methods have exponential cost in the number of sites
- PEPS contraction is needed to compute expectation values
- *Boundary contraction* is common for finite PEPS and can be simplified with `einsumsvd`



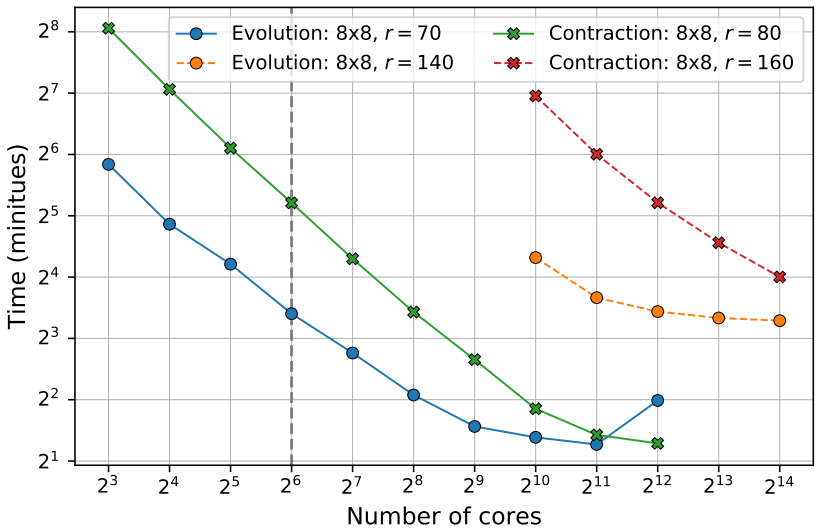
Implicit Randomized `einsumsvd`

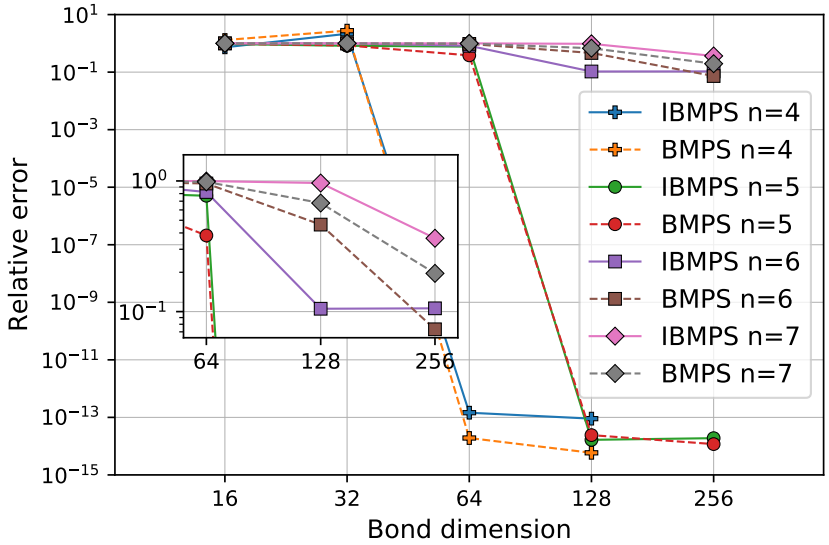
- The `einsumsvd` primitive provides an effective abstraction for tensor network simulation methods



- An efficient general implementation is to leverage randomized SVD / orthogonal iteration, which iteratively computes a low-rank SVD by a matrix–matrix product that can be done implicitly via tensor contractions

PEPS Benchmark Performance



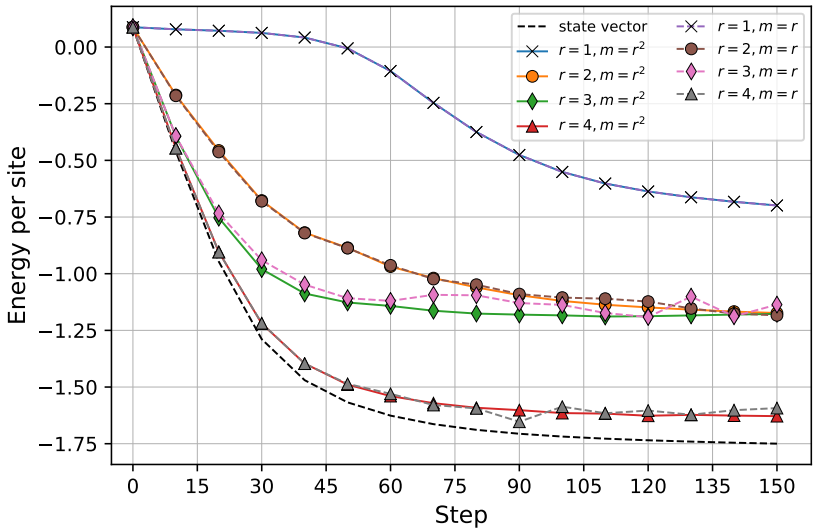


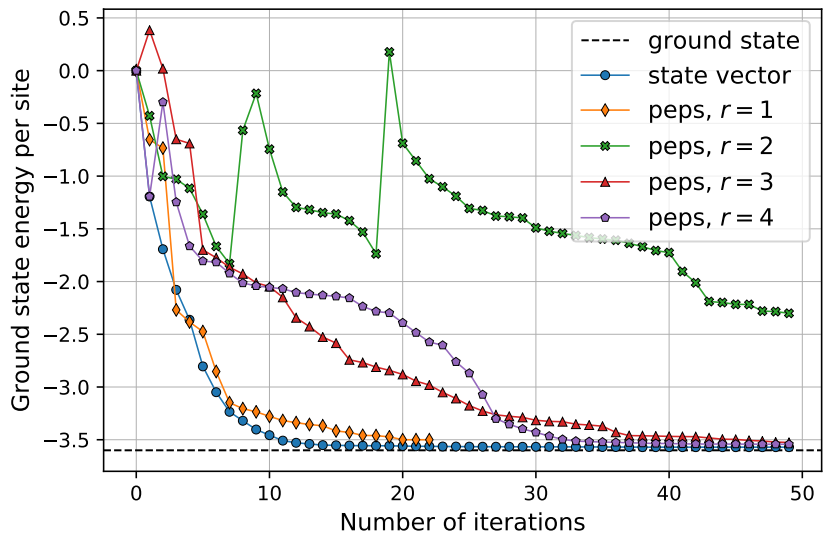
- We introduce a new library, Koala[1], for high-performance simulation of quantum circuits and time evolution with PEPS[2]
- Koala achieves good parallel scalability for approximate gate application (evolution) and contraction
- Approximation can be effective even for adversarially-designed circuits such as Google's random quantum circuit model (figure on right)

[1] https://github.com/cyclops-community/koala

[2] Yuchen Pang, Tianyi Hao, Annika Dugad, Yiqing Zhou, and E.S. SC 2020

PEPS Accuracy for Quantum Simulation





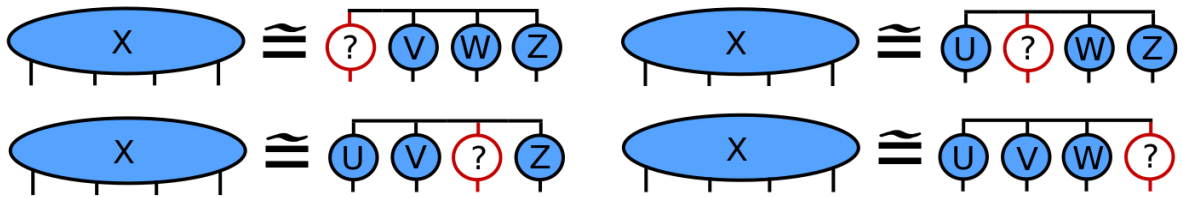
- ITE code achieves improvable accuracy with increased PEPS bond dimension, but approximation in PEPS contraction is not variational
- Variational quantum eigensolver (VQE), which represents a wavefunction using a parameterized circuit $U(\theta)$ and minimizes

$$\langle U(\theta)| H |U(\theta)\rangle ,$$

also achieves improvable accuracy with higher PEPS bond dimension

Automatic Differentiation for Tensor Computations

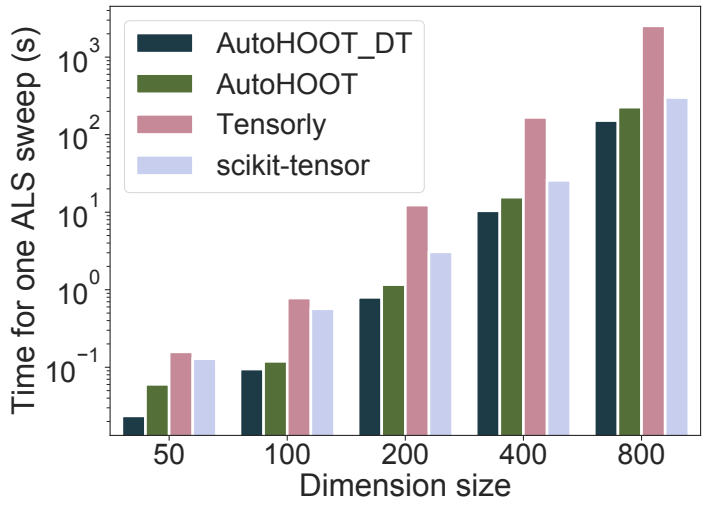
- Tensor network and tensor decomposition methods all typically based on applying Newton's method on a sequence of subsets of variables

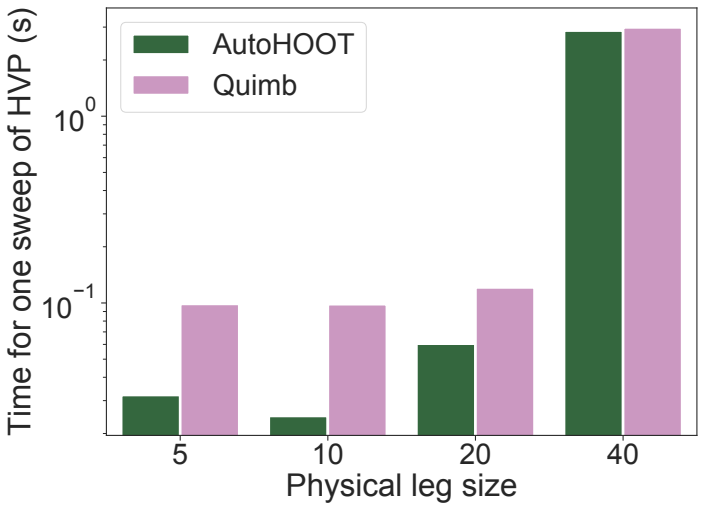


- Automatic differentiation (AD) in principle enables automatic generation of these methods
- However, existing AD tools such as Jax (used by TensorFlow) are designed for deep learning and are ineffective for more complex tensor computations
  - these focus purely on first order optimization via Jacobian-vector products
  - unable to propagate tensor algebra identities such as $(\boldsymbol{A} \otimes \boldsymbol{B})^{-1} = \boldsymbol{A}^{-1} \otimes \boldsymbol{B}^{-1}$ to generate efficient code

AutoHOOT: Automatic High-Order Optimization for Tensors

- AutoHOOT[1] provides a tensor-algebra centric AD engine
- Designed for `einsum` expressions and alternating minimization common in tensor decomposition and tensor network methods
- Python-level AD is coupled with optimization of contraction order and caching of intermediates
- Generates code for CPU/GPU/supercomputers using high-level back-end interface to tensor contractions





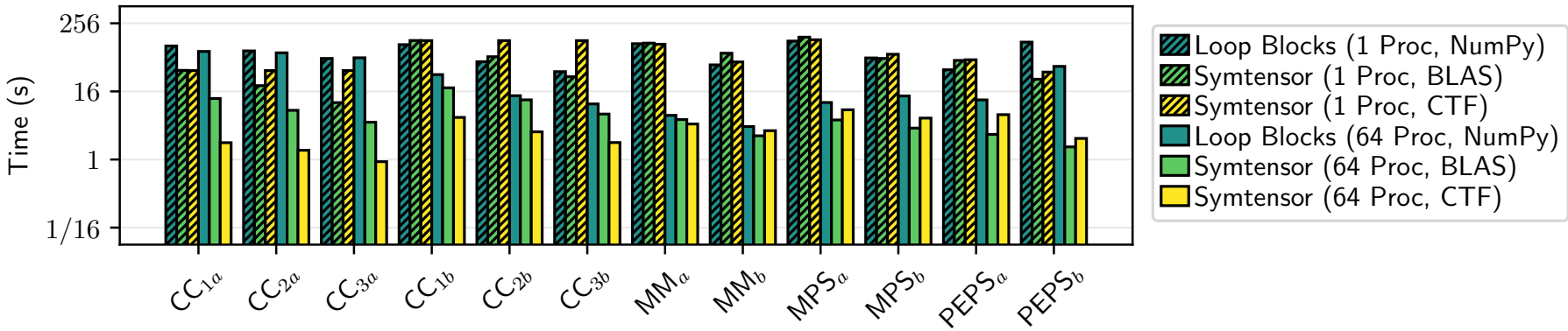
[1] Linjian Ma, Jiayu Ye, and E.S. PACT 2020

Group Symmetry in Tensor Contractions

- Tensor with cyclic group symmetry can be represented as block-sparse

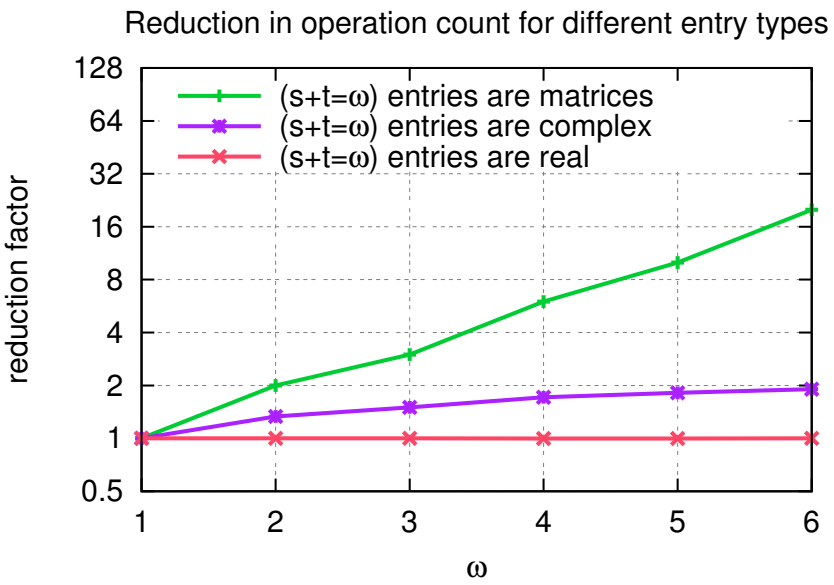
$$t_{ijk\ldots} = 0 \quad \text{if} \quad \lfloor i/G_1 \rfloor + \lfloor j/G_2 \rfloor + \lfloor k/G_3 \rfloor + \cdots \neq 0 \pmod{G}.$$

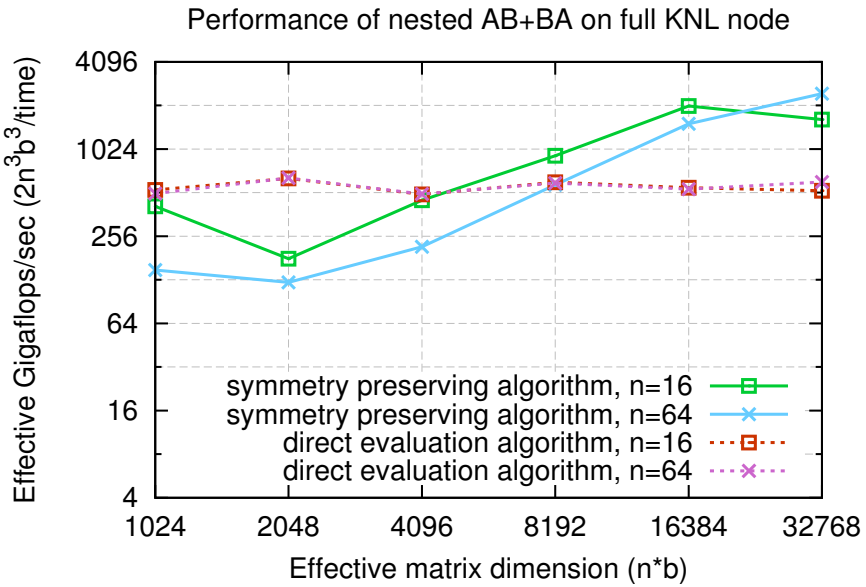
- Group symmetries of multiple types arise due to conservation laws when physical systems (quantum number symmetry, spin symmetry, rotational symmetry, translational symmetry)
- New contraction algorithm, *irreducible representation alignment* uses new dense reduced form tensor to handle group symmetry without looping over blocks or sparsity[1]



[1] Y. Gao, P. Helms, G. Chan, and E.S., arXiv:2007.08056

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

Communication Cost of Fast Bilinear Algorithms

- Given inputs $\boldsymbol{a}$ and $\boldsymbol{b}$, a bilinear algorithm computes

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

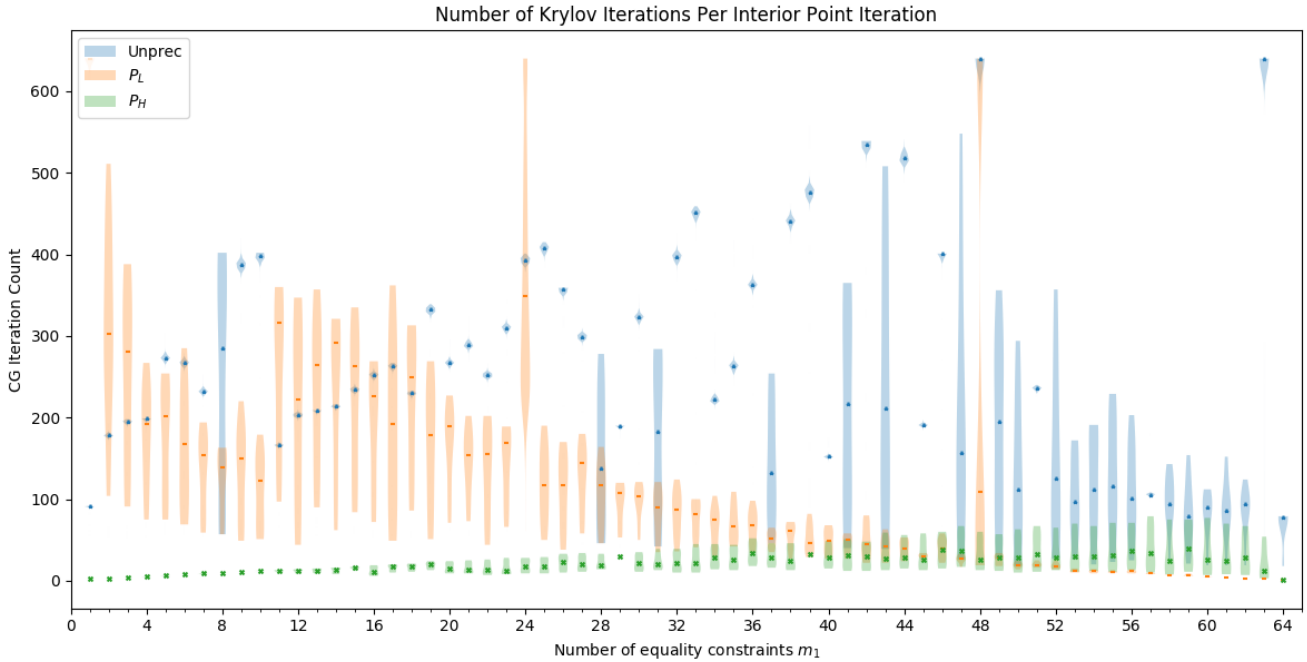
- All fast algorithmns for matrix multiplication, convolution, and symmetric tensor contractions are bilinear algorithms
- Communication lower bounds can be attained for any execution of a bilinear algorithm, given a lower bound on the rank of subset of columns of $\boldsymbol{F}^{(\boldsymbol{A})}$, $\boldsymbol{F}^{(\boldsymbol{B})}$, or $\boldsymbol{F}^{(\boldsymbol{C})}$[1]
- Can automatically obtain rank lower bounds for $\boldsymbol{A} \otimes \boldsymbol{B}$ from that of $\boldsymbol{A}$ and $\boldsymbol{B}$, enabling application of these lower bounds to nested algorithms such Strassen's algorithm, convolution, and symmetry preserving algorithms applied to partially symmetric contractions[2]

[1] E.S., J. Demmel, T. Hoefler arXiv:1707.04618

[2] Caleb Ju, Yifan Zhang, E.S. in preparation

Conclusion and Future Directions

- Talk introduced new algorithms and software for tensor contractions, tensor decomposition, and tensor networks, considering challenges involved in handling symmety, sparsity, and parallelism
- We are also exploring solvers for QP interior point methods via a new Schur complementation strategy and preconditioners[1]

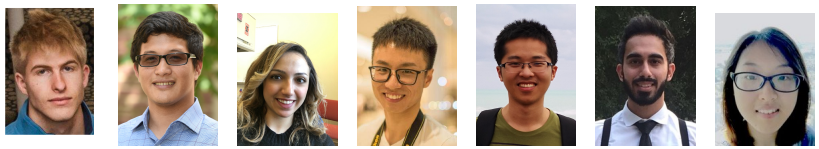


[1] Samah Karim and E.S. in preparation

Acknowledgements

- Laboratory for Parallel Numerical Algorithms (LPNA) at University of Illinois
- This work has been support by NSF awards #1839204 (RAISE-TAQS), #1931258 (CSSI), #1942995 (CAREER)
- Stampede2 resources at TACC via XSEDE

L P N A @CS@Illinois



DOE CSGF

NSF

http://lpna.cs.illinois.edu