High Performance Tensor Computations

Edgar Solomonik

L P N A @CS@Illinois

Department of Computer Science
University of Illinois at Urbana-Champaign

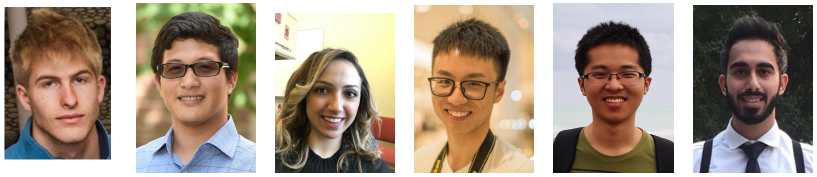
SPCL_Bcast(COMM_WORLD) seminar

Laboratory for Parallel Numerical Algorithms

Recent/ongoing research topics
(*-covered today)

- parallel matrix computations
  - matrix factorizations
  - eigenvalue problems
  - preconditioners
- 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*

L P N A @CS@Illinois

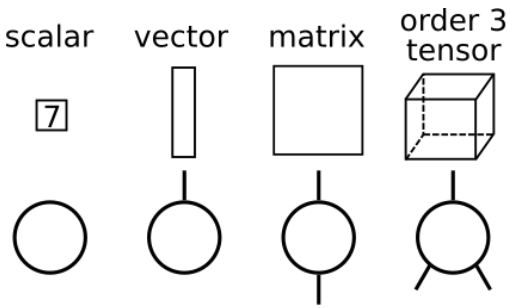


DOE CSGF NSF

http://lpna.cs.illinois.edu

Tensor Contractions

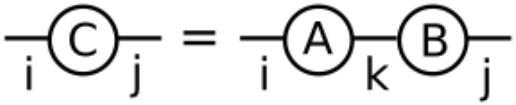
- A tensor of order N has N modes and dimensions $s \times \cdots \times s$



- Two or more tensors can be contracted together in various ways

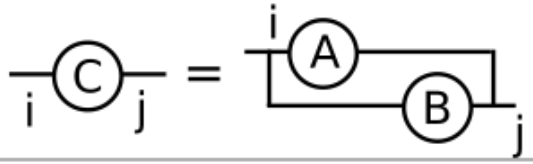
Matrix Multiplication

$c_{ij} = \Sigma_k\ a_{ik}\ b_{kj}$



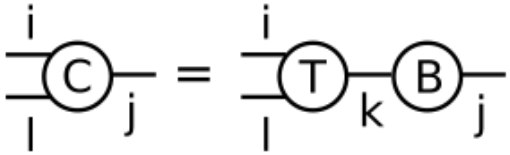
Hadamard Product

$c_{ij} = a_{ij}\ b_{ij}$



Tensor Times Matrix (TTM)

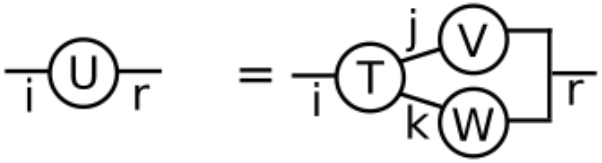
$c_{ijl} = \Sigma_k\ t_{ikl}\ b_{kj}$



MTTKRP

(matricized tensor times Khatri-Rao product)

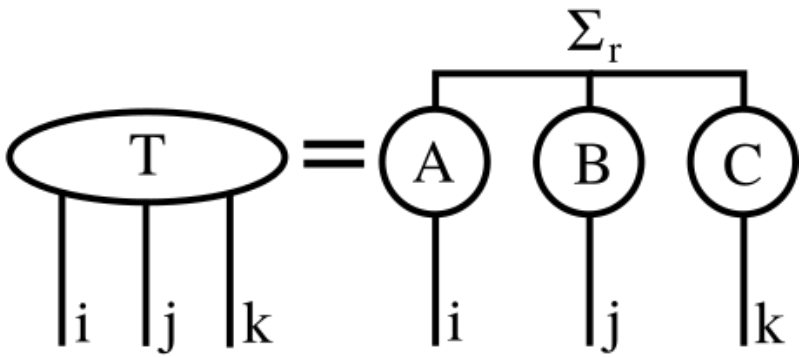
$u_{ir} = \Sigma_{jk}\ t_{ijk}\ v_{jr}\ w_{kr}$



Tensor Decompositions

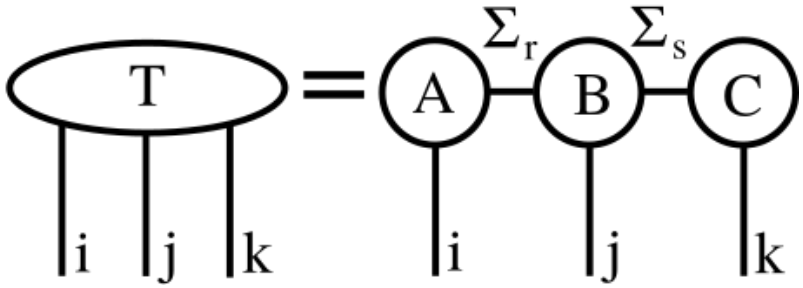
- Canonical polyadic (CP) tensor decomposition[1]

$$t_{ijk} = \sum_{r=1}^{R} u_{ir} v_{jr} w_{kr}$$

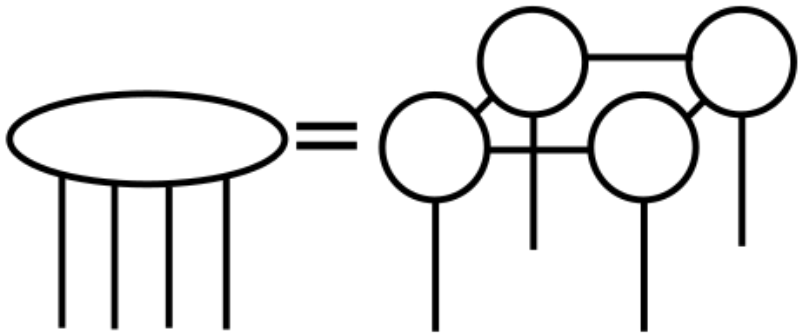


- 1D tensor network / Matrix product state (MPS) / tensor train (TT) decomposition

$$t_{ijk} = \sum_{r} \sum_{s} u_{ir} v_{rjs} w_{sk}$$

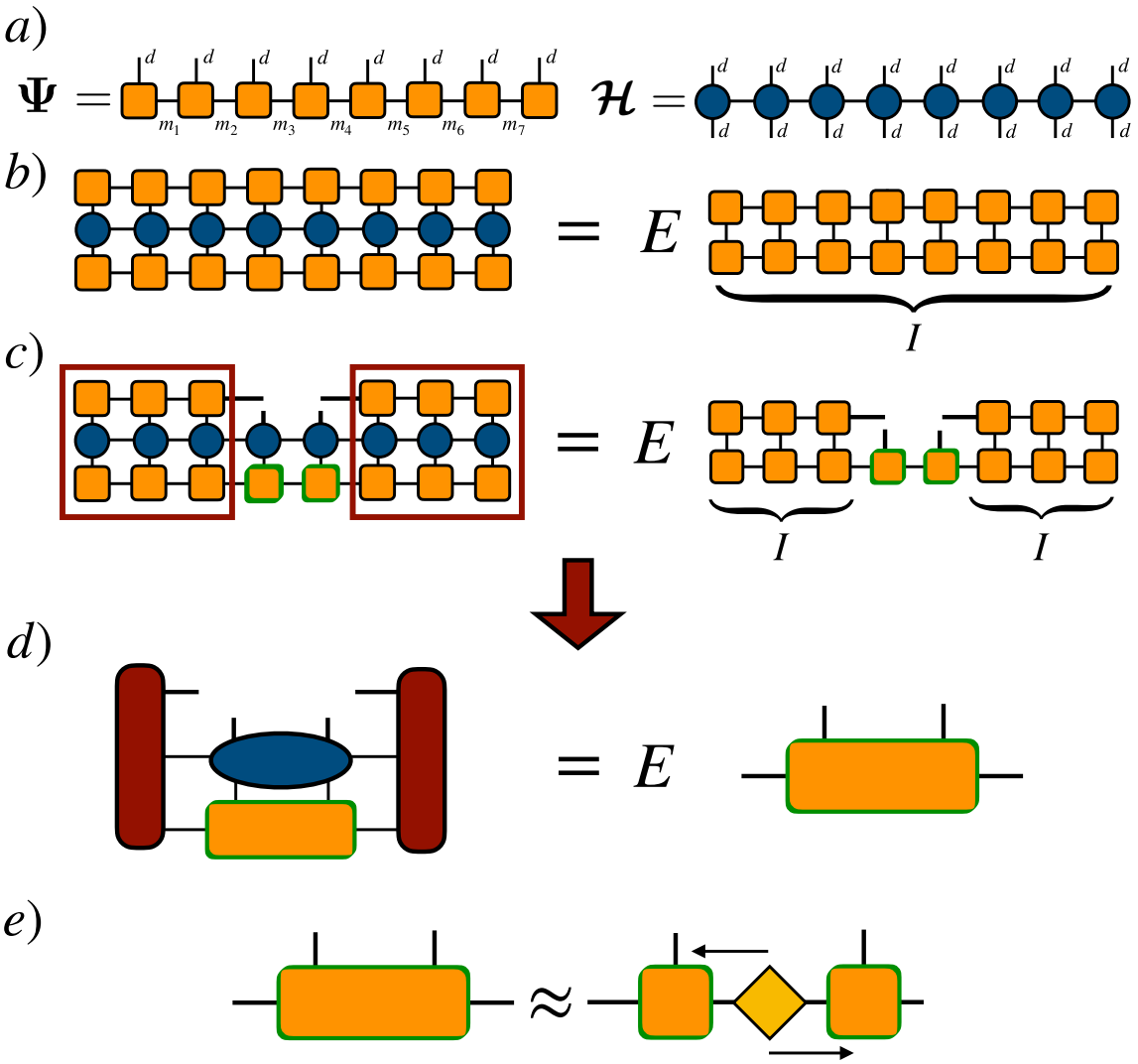


- 2D tensor network / projected entangled pair state (PEPS)



[1] T.G. Kolda and B.W. Bader, SIAM Review 2009

Tensor Network Methods



Applications

- Tensor Decompositions
  - data mining e.g., high-order clustering, typically low-rank decomposition augmented with constraints
  - model/data compression e.g., neural networks and quantum chemistry, relatively high rank needed for accuracy
  - discovery of bilinear algorithms (e.g., Strassen's algorithm for matrix multiplication), small sparse tensors with relatively high rank

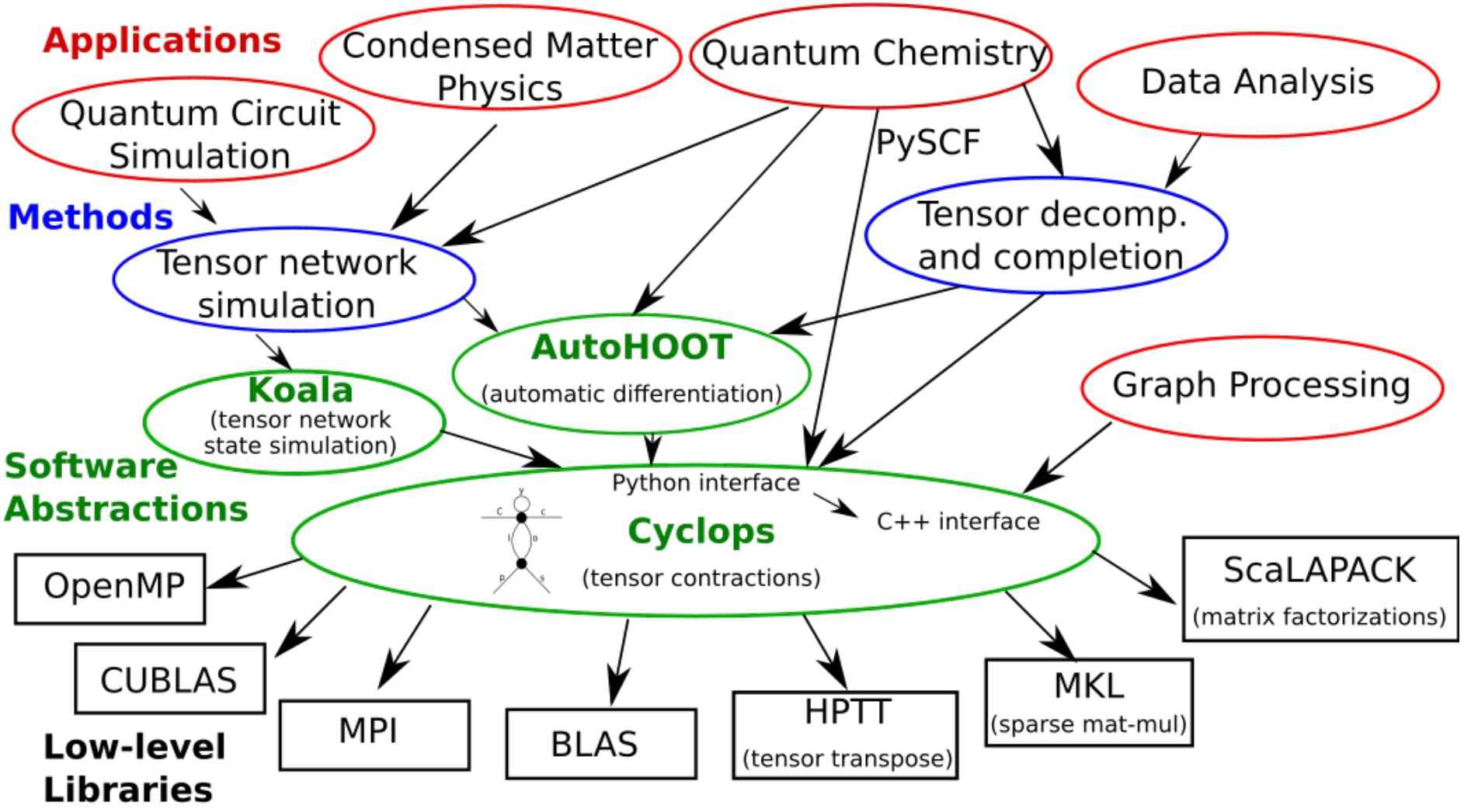
Applications

- Tensor Decompositions
  - data mining e.g., high-order clustering, typically low-rank decomposition augmented with constraints
  - model/data compression e.g., neural networks and quantum chemistry, relatively high rank needed for accuracy
  - discovery of bilinear algorithms (e.g., Strassen's algorithm for matrix multiplication), small sparse tensors with relatively high rank
- Tensor Networks
  - eigenvalue and least squares problems where system of equations and vector are represented by low rank tensor networks
  - prevalent in simulation of quantum systems (spins, electrons, qubits) given Hamiltonian or quantum circuit description

Applications

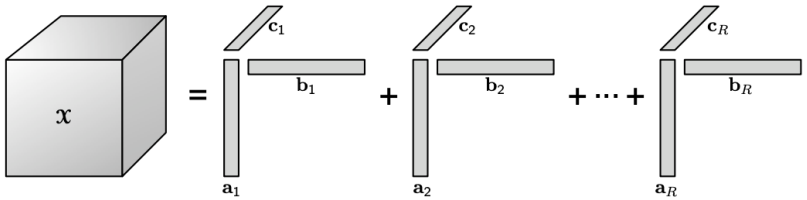
- Tensor Decompositions
  - data mining e.g., high-order clustering, typically low-rank decomposition augmented with constraints
  - model/data compression e.g., neural networks and quantum chemistry, relatively high rank needed for accuracy
  - discovery of bilinear algorithms (e.g., Strassen's algorithm for matrix multiplication), small sparse tensors with relatively high rank
- Tensor Networks
  - eigenvalue and least squares problems where system of equations and vector are represented by low rank tensor networks
  - prevalent in simulation of quantum systems (spins, electrons, qubits) given Hamiltonian or quantum circuit description
- Tensor Contractions
  - prevalent within tensor network and tensor decomposition methods
  - also arise in high-accuracy quantum chemistry methods (e.g., coupled cluster), where tensors often have symmetries

Software Abstractions for Tensor Computations



CP Tensor Decomposition Algorithms

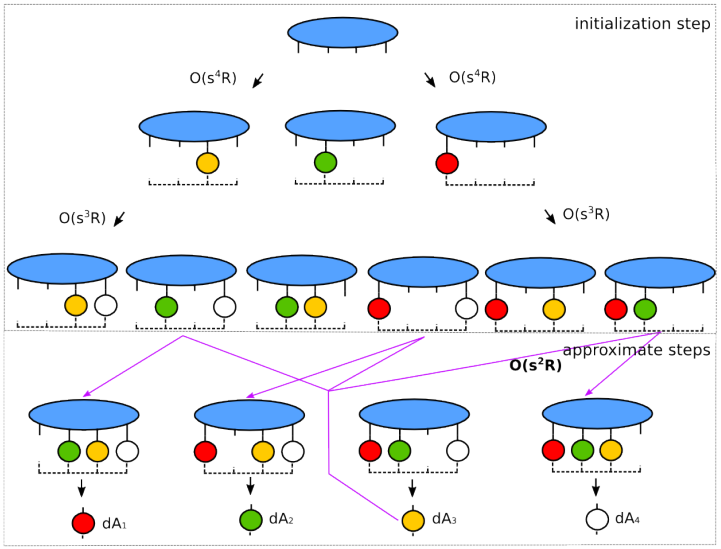
- Tensor of order N has N modes and dimensions $s \times \cdots \times s$
- Canonical polyadic (CP) tensor decomposition[1]

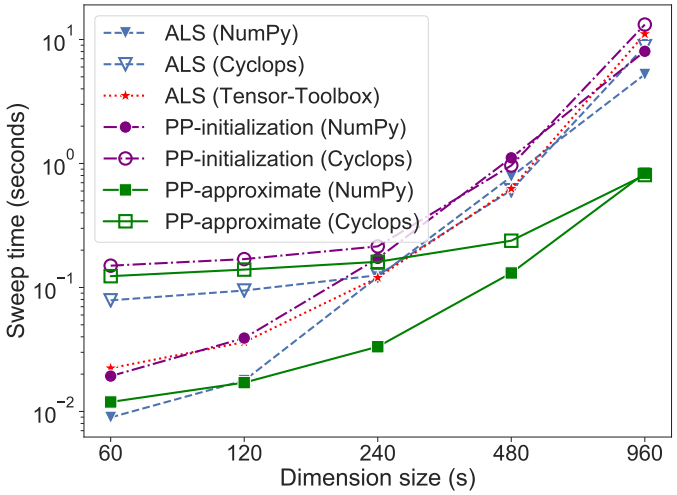


- Alternating least squares (ALS) is most widely used method
  - Optimize one factor matrix at a time, yielding quadratic optimization subproblems
  - Achieves monotonic linear convergence
- Gauss-Newton method is an emerging alternative
  - Optimizes all factor matrices at once by quadratic approximation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

[1] T.G. Kolda and B.W. Bader, SIAM Review 2009

Pairwise Perturbation Algorithm





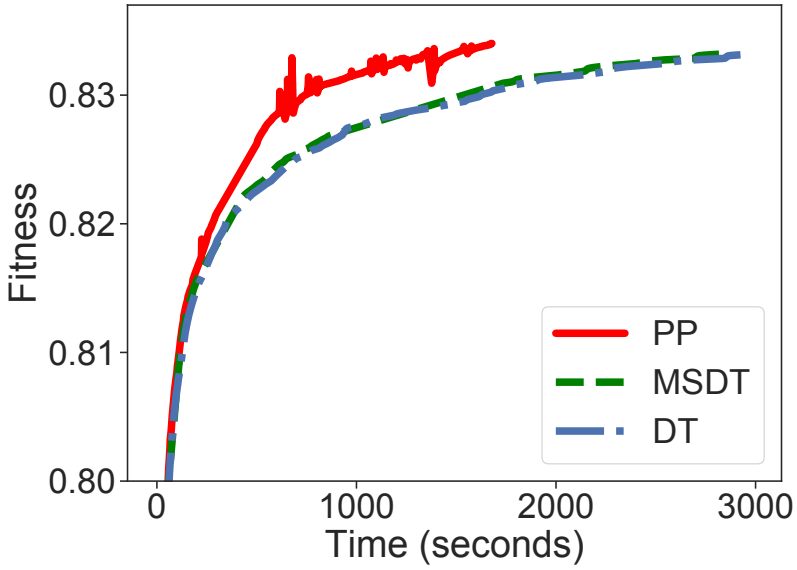
New algorithm: pairwise perturbation (PP)[1] approximates ALS

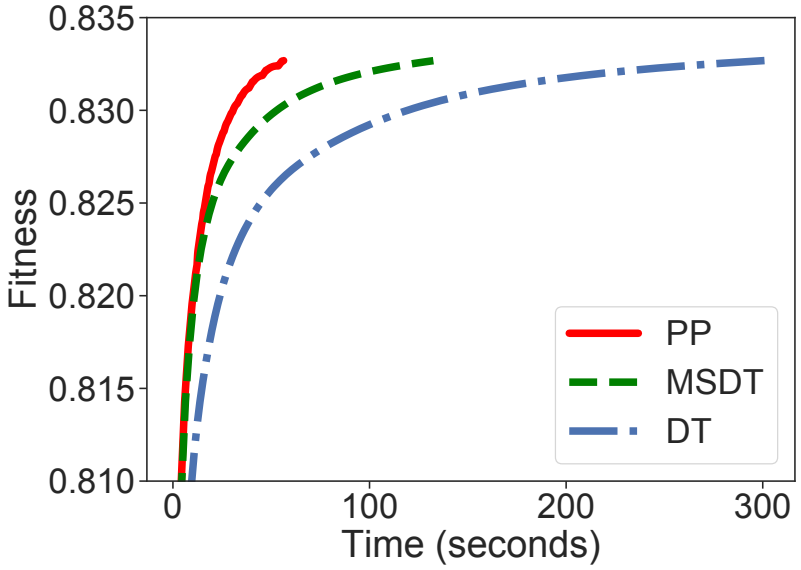
- based on perturbative expansion of ALS update to approximate MTTKRP
- approximation is accurate when ALS updates stagnate
- rank $R < s^{N-1}$ CP decomposition:
  - ALS sweep cost $O(s^N R) \Rightarrow O(s^2 R)$, up to 33x speed-up

Linjian Ma

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

Parallel Pairwise Perturbation Algorithm





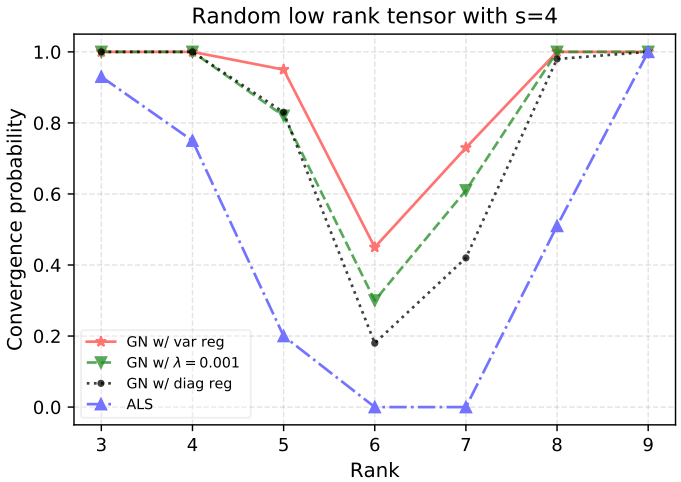
Effective parallelization by decomposing MTTKRP into local MTTKRPs [1]

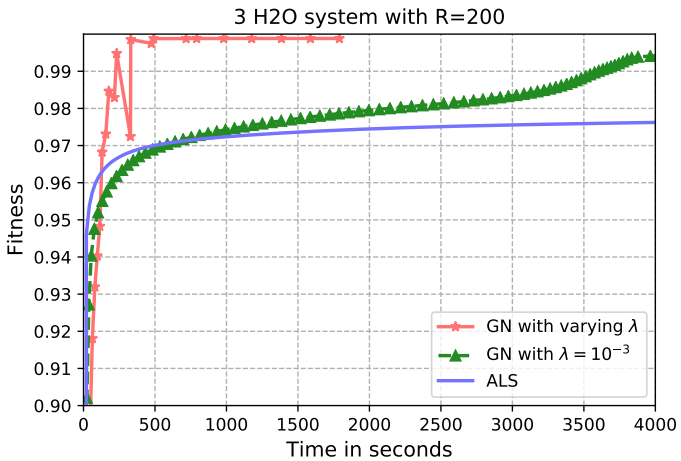
$$\boldsymbol{U} = \text{MTTKRP}(\boldsymbol{\mathcal{T}}, \boldsymbol{V}, \boldsymbol{W}) \Rightarrow \boldsymbol{U}_i = \sum_{j,k} \text{MTTKRP}(\boldsymbol{\mathcal{T}}_{ijk}, \boldsymbol{V}_j, \boldsymbol{W}_k)$$

- 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, reducing communication cost
- multi-sweep dimension-tree (MSDT) amortizes terms across sweeps

[1] L. Ma, E.S. to appear on arXiv, October 2020.

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

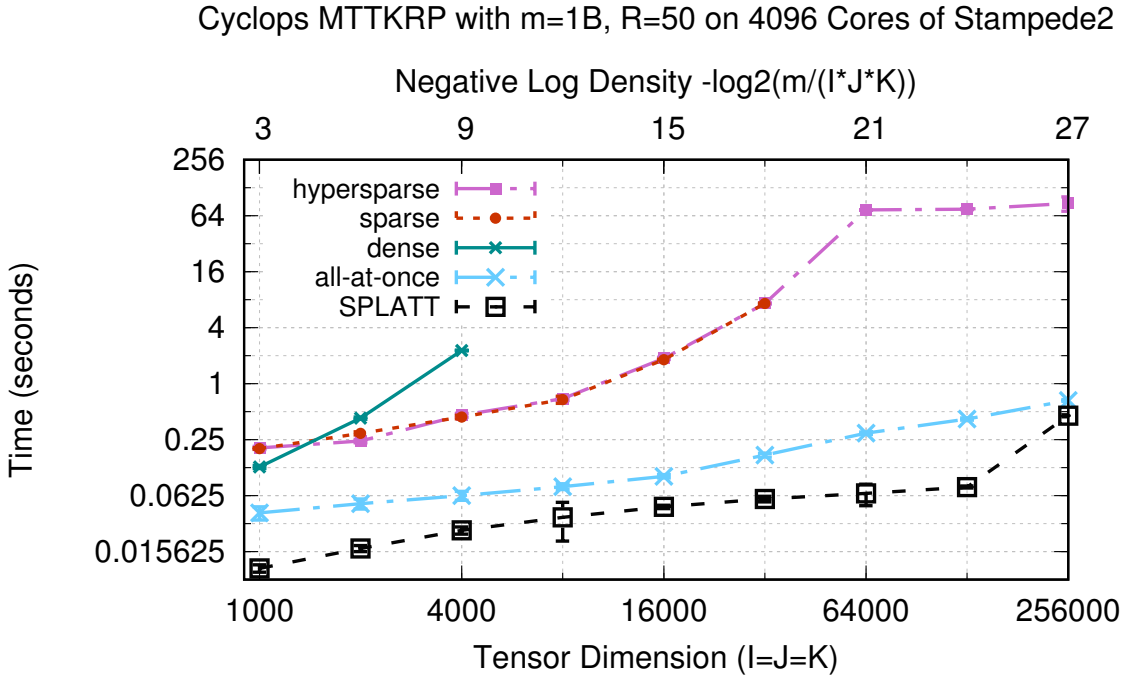
[2] P. Tichavsky, A. H. Phan, and A. Cichocki., 2013

Sparse Tensor Decomposition

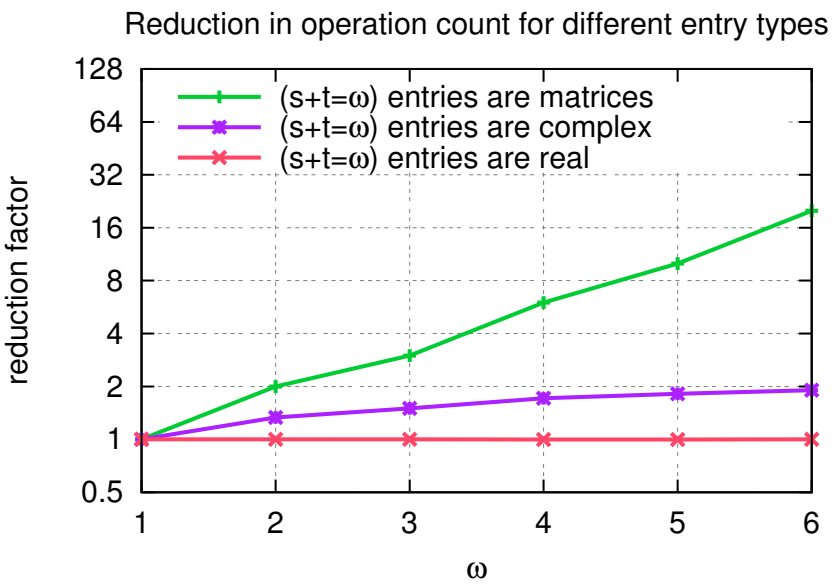
- Sparse tensor decomposition is dominated by MTTKRP

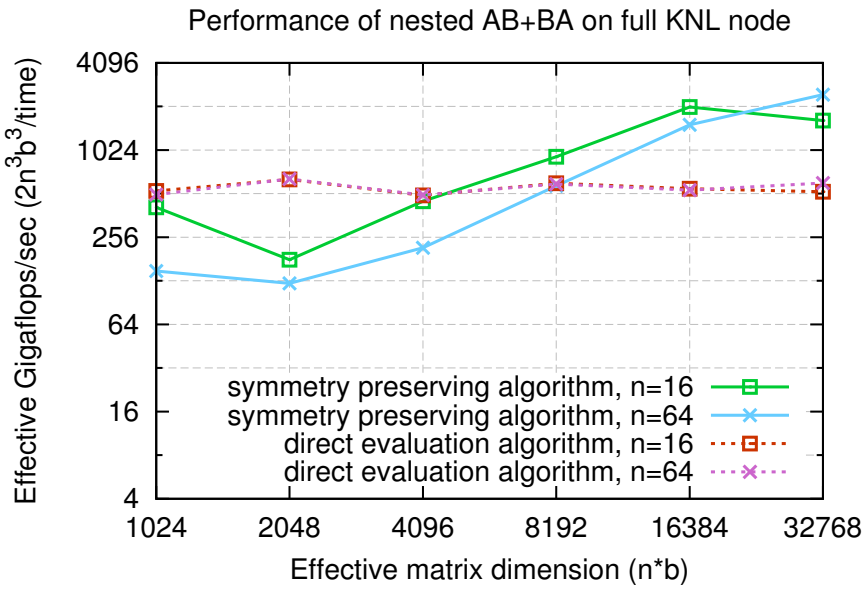
$$u_{ir} = \sum_{j,k} t_{ijk} v_{jr} w_{kr}$$

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



Permutational Symmetry in Tensor Contractions





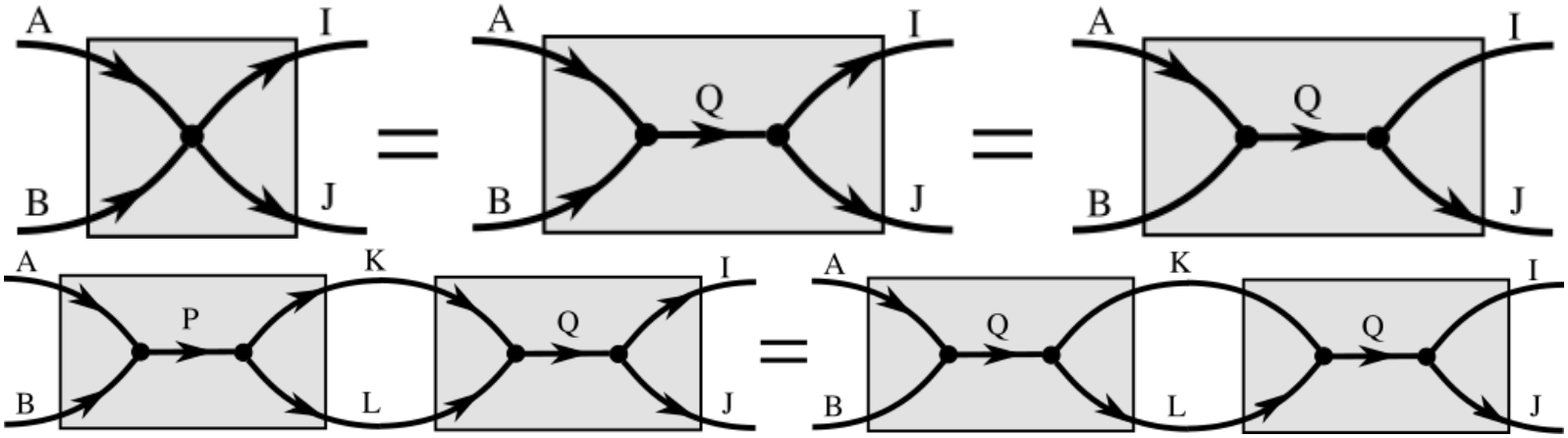
New contraction algorithms reduce cost via permutational symmetry[1]

- 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

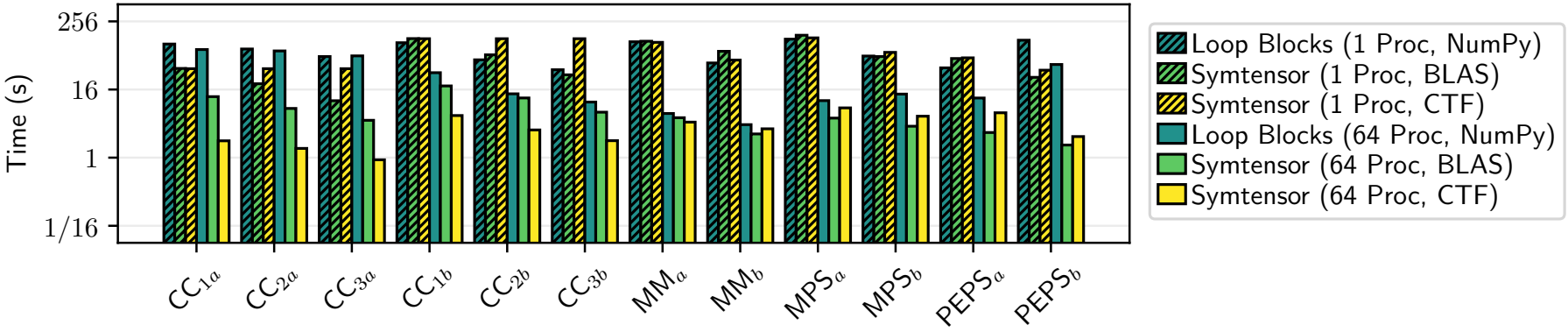
$$\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B} + \boldsymbol{B}\boldsymbol{A} \Rightarrow c_{ij} = \sum_k [(a_{ij} + a_{ik} + a_{jk}) \cdot (b_{ij} + b_{ik} + b_{jk})] - \ldots$$

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

Group Symmetry in Tensor Contractions



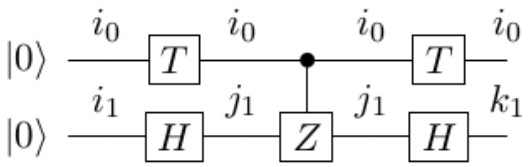
New contraction algorithm, *irreducible representation alignment* uses new reduced form to handle group symmetry (momentum conservation, spin, quantum numbers, etc.) without looping over blocks or sparsity[1]

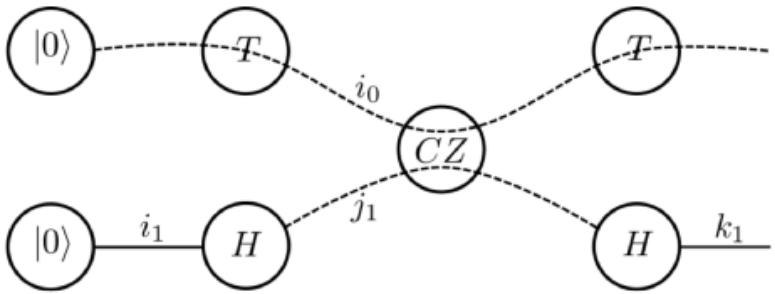


[1] Y. Gao, P. Helms, G. Chan, and E.S., arXiv:2007.08056

Quantum Circuit Simulation with Tensor Networks

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

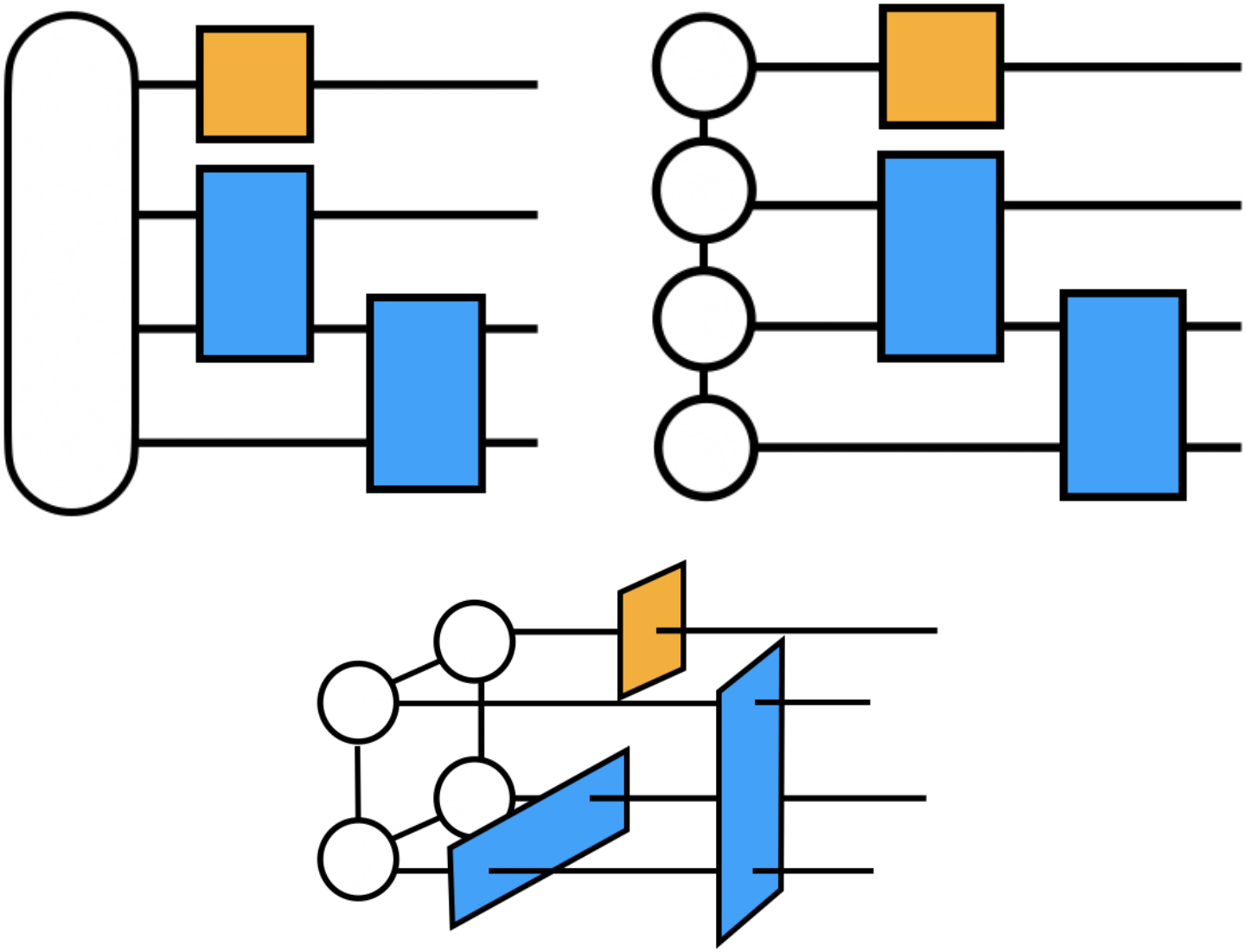




- Why use HPC to (approximately) simulate quantum circuits?
  - enable development/testing/tuning of larger quantum circuits
  - understand approximability of different quantum algorithms
  - quantify sensitivity of algorithms to noise/error
  - potentially enable new hybrid quantum-classical algorithms
- 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]

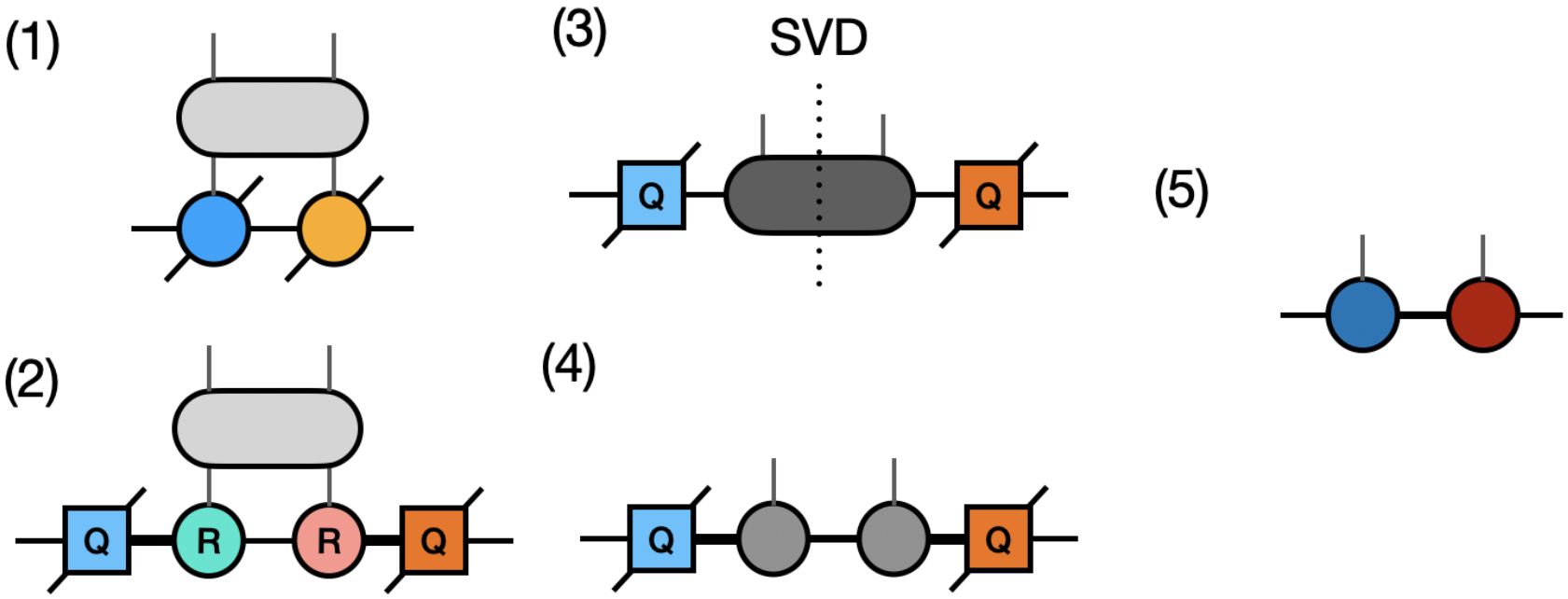
[1] Markov and Shi SIAM JC 2007
[2] Pednault et al. arXiv:1710.05867
[3] Guo et al. Phys Rev Letters, 2019

Tensor Network State Simulation



Approximate Application of Two-Site Operators

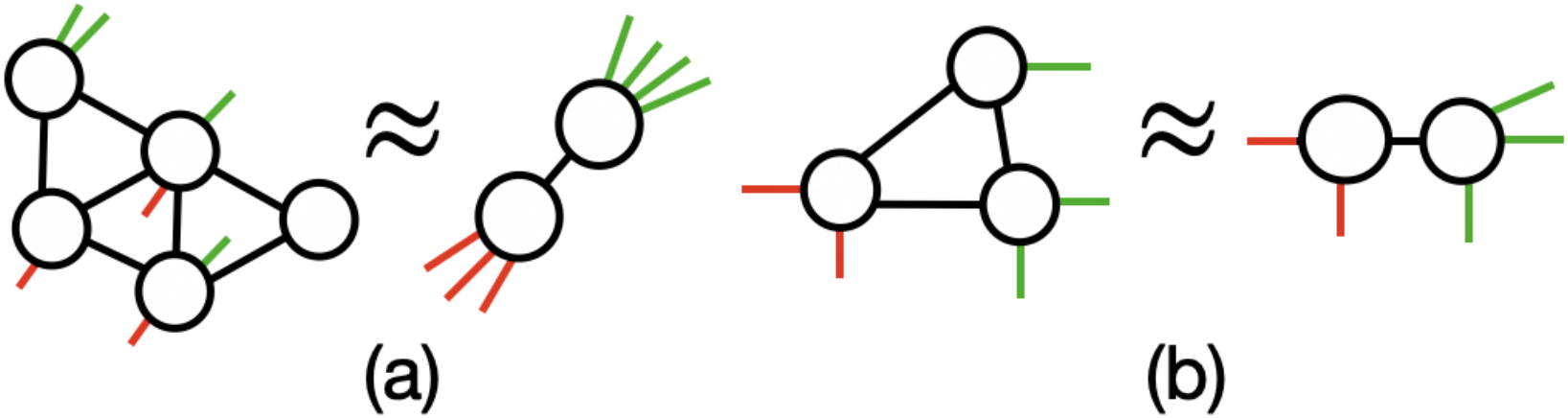
- Consider application of a two-site operator on neighboring PEPS sites
- *Simple update (QR-SVD)* algorithm:



- We provide an efficient distributed implementation of QR-SVD
- This operation is an instance of what we'll refer to as `einsumsvd` and QR-SVD is one algorithm/implementation

Implicit Randomized `einsumsvd`

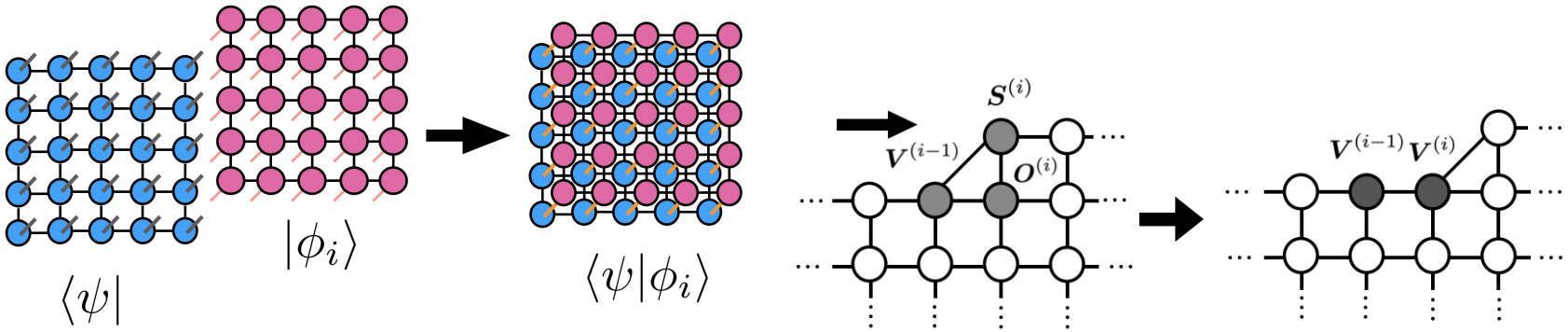
- The `einsumsvd` primitive will also enable effective algorithms for PEPS contraction



- 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 Contraction

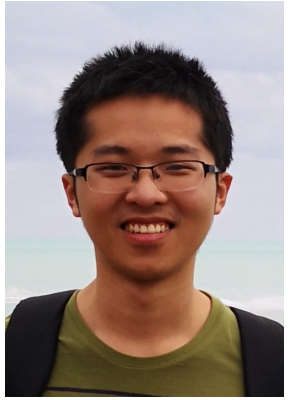
- Exact contraction of PEPS is #P-complete, so known methods have exponential cost in the number of sites
- PEPS contraction is needed to compute expectation values
- *Boundary contraction* is common for finite PEPS and can be simplified with `einsumsvd`



Koala

- We introduce a new library, Koala[1], for high-performance simulation of quantum circuits and time evolution with PEPS[2]

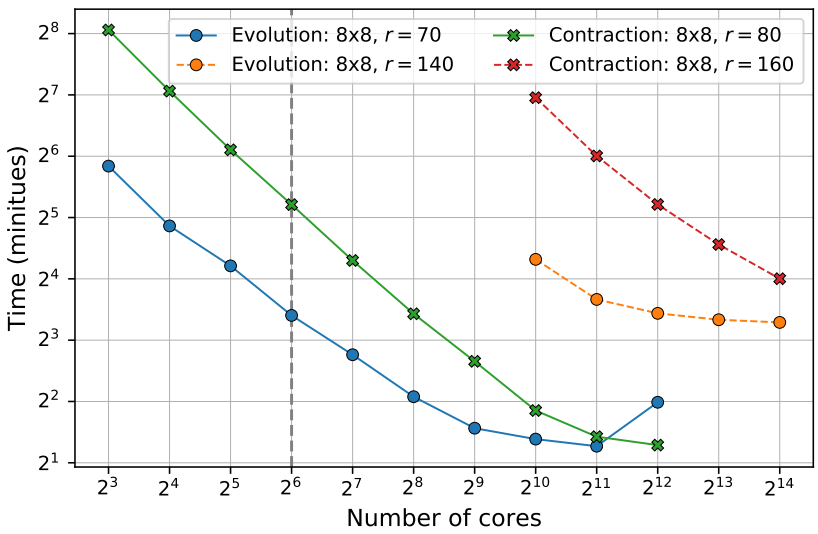
```python
from koala import peps, Observable
from tensorbackends.interface import ImplicitRandomizedSVD

# Create a 2-by-3 PEPS in distributed memory using CTF
qstate = peps.computational_zeros(nrow=2, ncol=3, backend='ctf')

# Construct operators and apply them to the quantum state
Y = qstate.backend.astensor([0,-1j,1j,0]).reshape(2,2)
CX = qstate.backend.astensor([1,0,0,0,0,1,0,0,0,0,0,1,0,0,1,0,])
CX = CX.reshape(2,2,2,2)

qstate.apply_operator(Y, [1])
qstate.apply_operator(CX, [1,4], update_option=peps.QRUpdate(rank=2))

# Construct an observable and calculate the expectation with IBMPS
observable = Observable.ZZ(3, 4) + 0.2 * Observable.X(1)
result = qstate.expectation(
    observable, use_cache=True,
    contract_option=peps.BMPS(ImplicitRandomizedSVD(rank=4)),
)
```

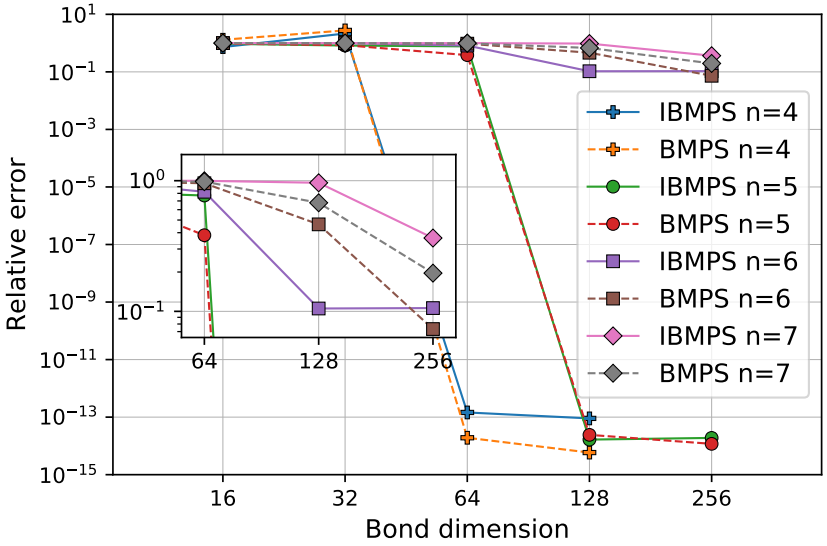


[1] https://github.com/cyclops-community/koala

[2] Yuchen Pang, Tianyi Hao, Annika Dugad, Yiqing Zhou, and E.S., to appear in proceedings of SC 2020, arXiv:2006.15234.

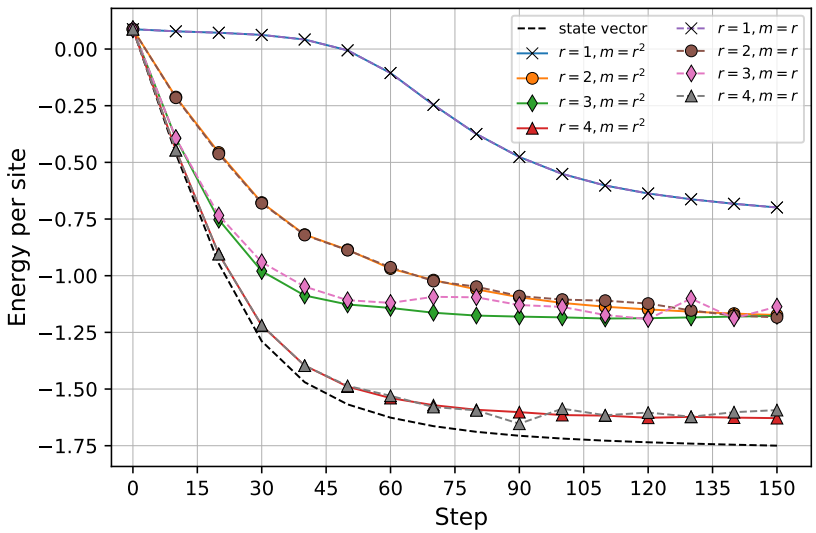
PEPS Benchmark Performance

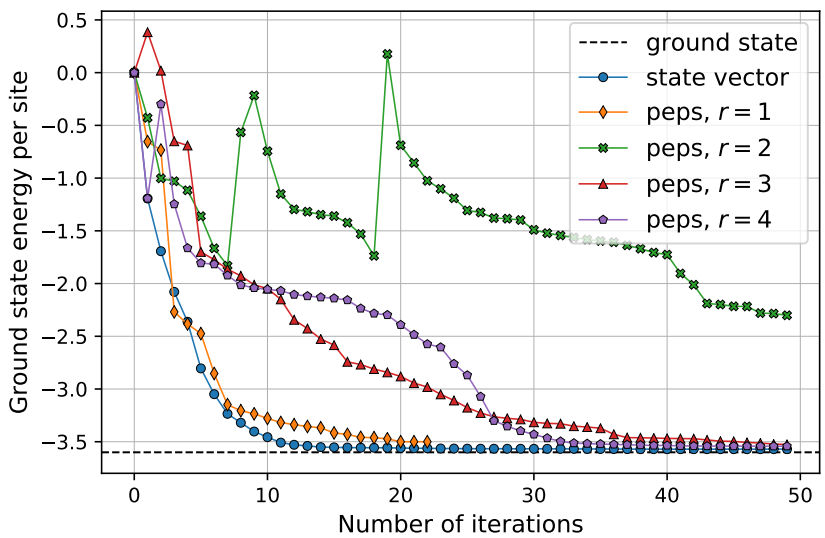




- Koala achieves good parallel scalability for approximate gate application (evolution) and contraction
- Approximation can be effective even for adversarially-designed circuits such as Google's random quantum circuit model (figure on right)

PEPS Accuracy for Quantum Simulation





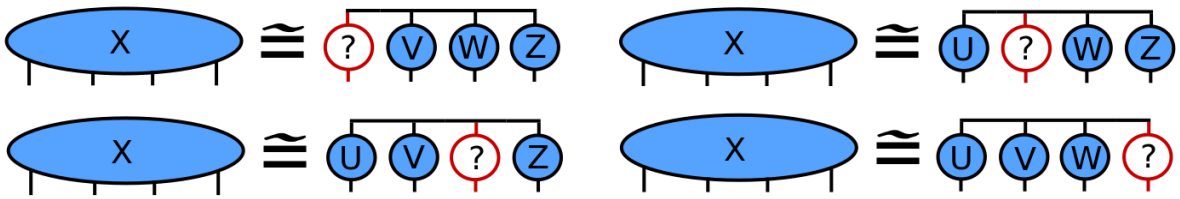
- ITE code achieves improvable accuracy with increased PEPS bond dimension, but approximation in PEPS contraction is not variational
- Variational quantum eigensolver (VQE), which represents a wavefunction using a parameterized circuit $U(\theta)$ and minimizes

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

also achieves improvable accuracy with higher PEPS bond dimension

Automatic Differentiation for Tensor Computations

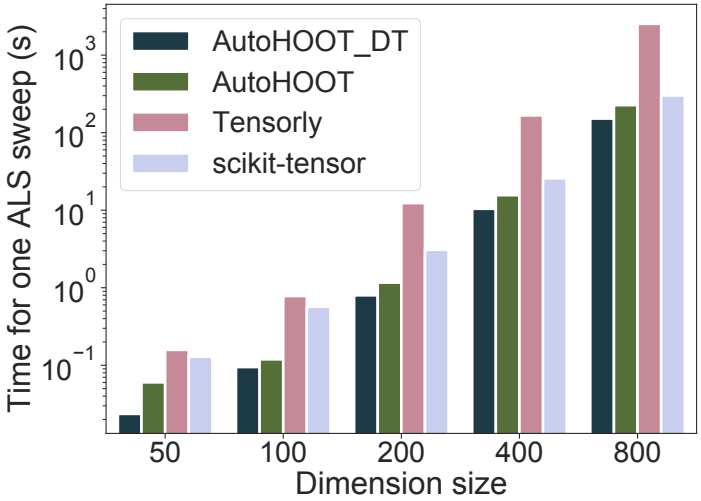
- Tensor network and tensor decomposition methods all typically based on applying Newton's method on a sequence of subsets of variables

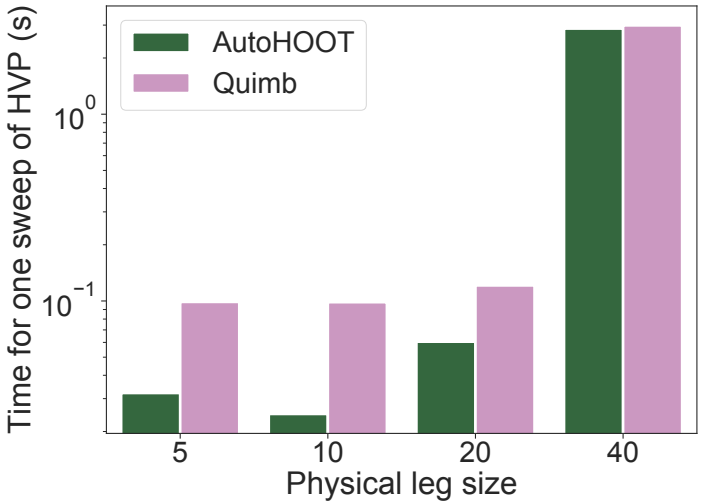


- Automatic differentiation (AD) in principle enables automatic generation of these methods
- However, existing AD tools such as Jax (used by TensorFlow) are designed for deep learning and are ineffective for more complex tensor computations
  - these focus purely on first order optimization via Jacobian-vector 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

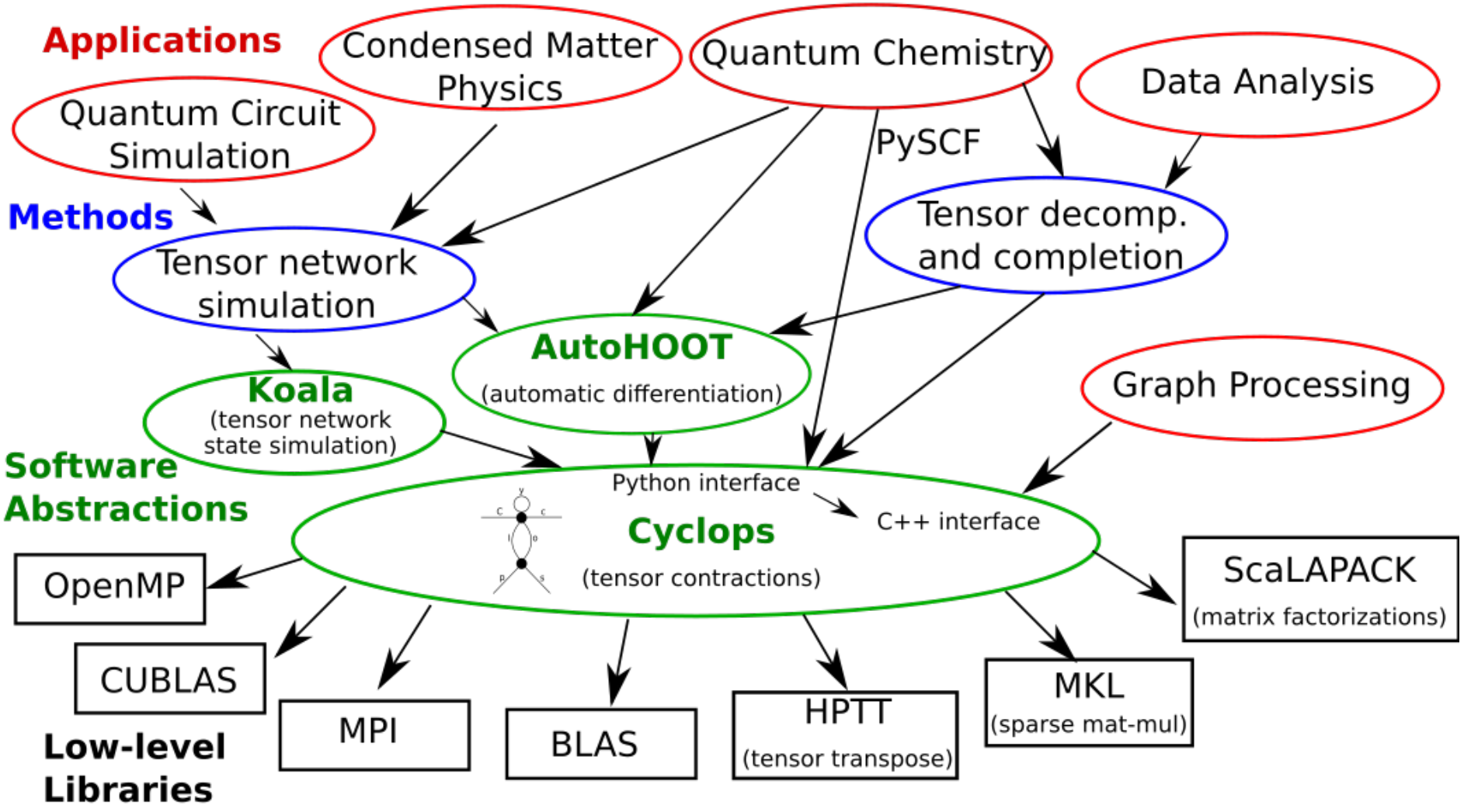
- AutoHOOT[1] provides a tensor-algebra centric AD engine
- Designed for `einsum` expressions and alternating minimization common in tensor decomposition and tensor network methods
- Python-level AD is coupled with optimization of contraction order and caching of intermediates
- Generates code for CPU/GPU/supercomputers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





[1] Linjian Ma, Jiayu Ye, and E.S. arXiv:2005.04540, 2020

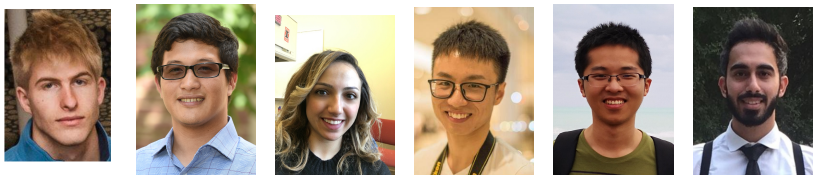
Software Abstractions for Tensor Computations



Acknowledgements

- Laboratory for Parallel Numerical Algorithms (LPNA) at University of Illinois, `lpna.cs.illinois.edu` and collaborators
- Funding from 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`

Backup slides

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

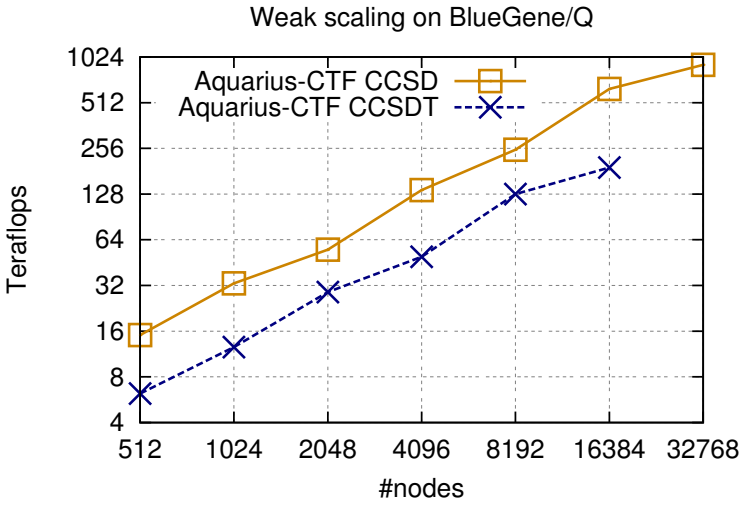
[2] E.S., D. Matthews, J. Hammond, J.F. Stanton, J. Demmel, JPDC 2014

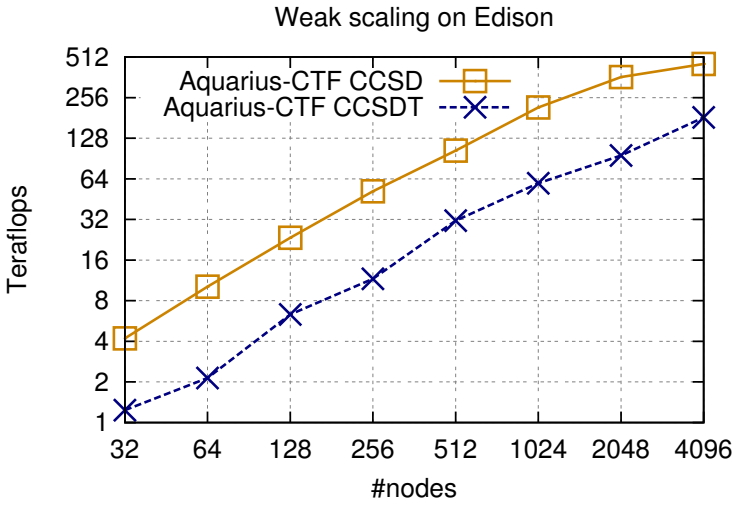
[3] E. Pednault, J.A. Gunnels, G. Nannicini, L. Horesh, T. Magerlein, E. S., E. Draeger, E. Holland, and R. Wisnieff, 2017

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

Electronic structure calculations with Cyclops

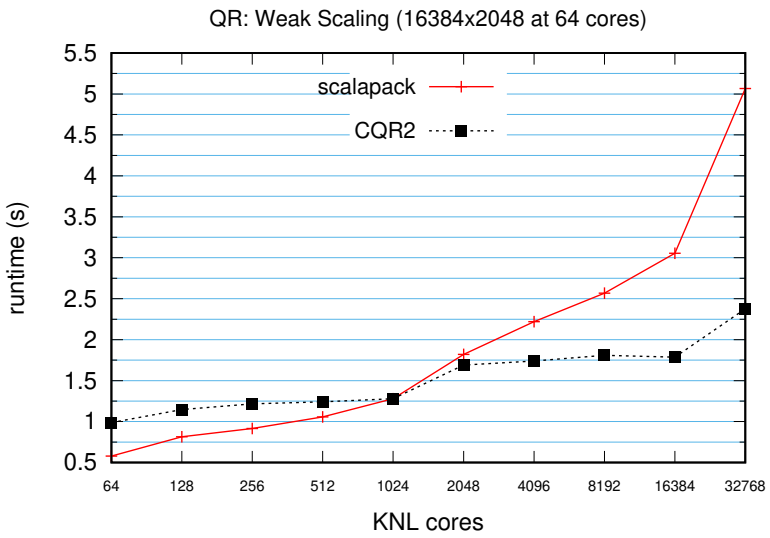
CCSD up to 55 (50) water molecules with cc-pVDZ
CCSDT up to 10 water molecules with cc-pVDZ

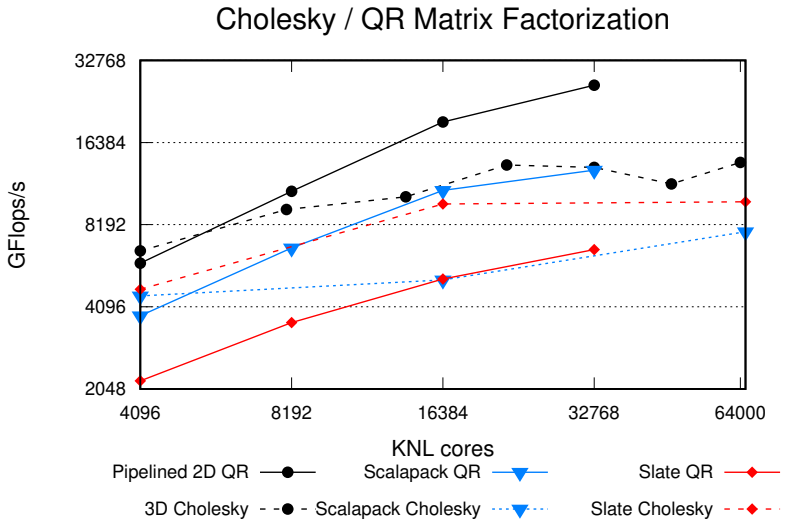




compares well to NWChem (up to 10x speed-ups for CCSDT)

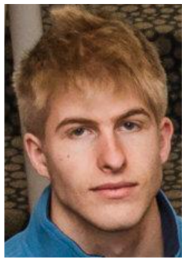
Faster Parallel Algorithms for Cholesky and QR





Cholesky-QR2[1] with 3D Cholesky gives a practical 3D QR algorithm[2]

- Compute $\boldsymbol{A} = \hat{\boldsymbol{Q}}\boldsymbol{R}$ using Cholesky $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{R}^T\boldsymbol{R}$
- Correct computed factorization by Cholesky-QR of $\hat{\boldsymbol{Q}}$
- Attains full accuracy so long as $\text{cond}(\boldsymbol{A}) < 1/\sqrt{\epsilon_{\text{mach}}}$

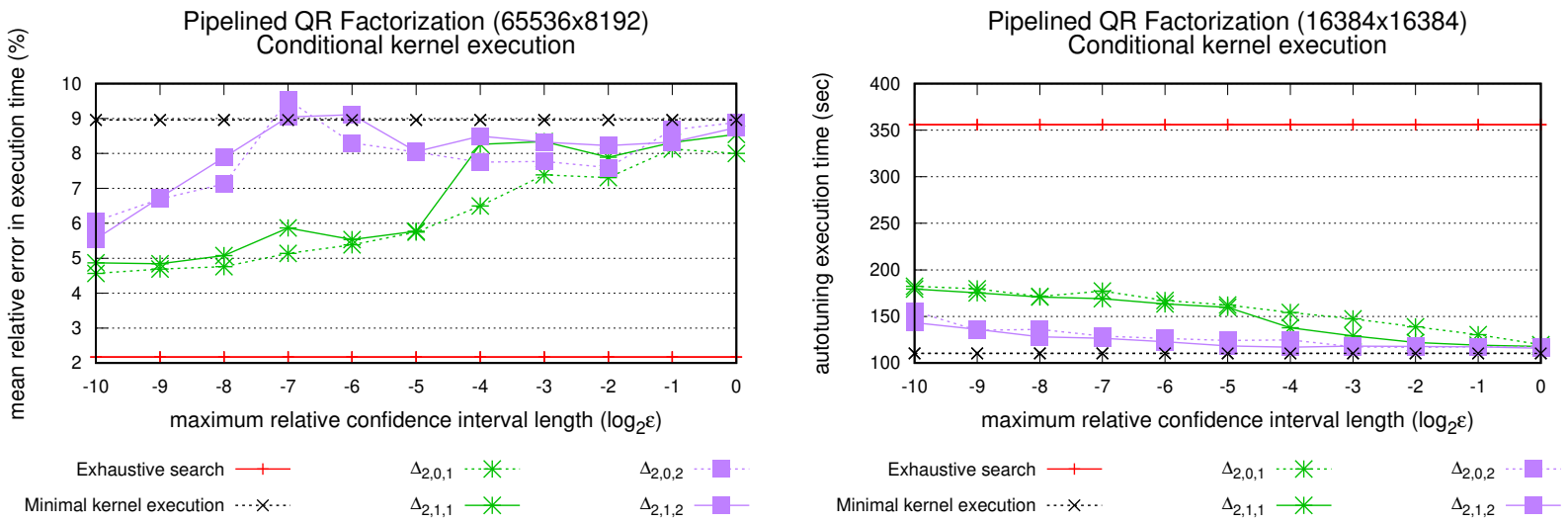


Edward Hutter

[1] T. Fukaya, Y. Nakatsukasa, Y. Yanagisawa, Y. Yamamoto, 2014

[2] E. Hutter, E.S., IPDPS 2019

Accelerating Autotuning using Critter



New tool critter accelerates autotuning by conditional execution[1]

- Avoids execution of computation and communication kernels if their performance is predictable using past observations
- Leverages critical path profiling to determine needed model accuracy for each kernel

Edward Hutter

[1] E. Hutter, Edgar Solomonik, to appear on arXiv