Scalable Algorithms for Tensor Computations

Edgar Solomonik

L P N A @CS@Illinois

Department of Computer Science
University of Illinois at Urbana-Champaign

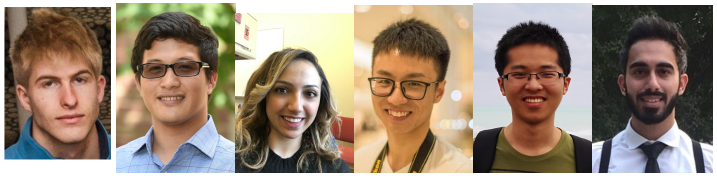
School of Computing, University of Utah

Laboratory for Parallel Numerical Algorithms

Recent/ongoing research topics

- parallel matrix computations
  - QR factorization
  - triangular solve
  - eigenvalue problems
- tensor computations (today)
  - 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

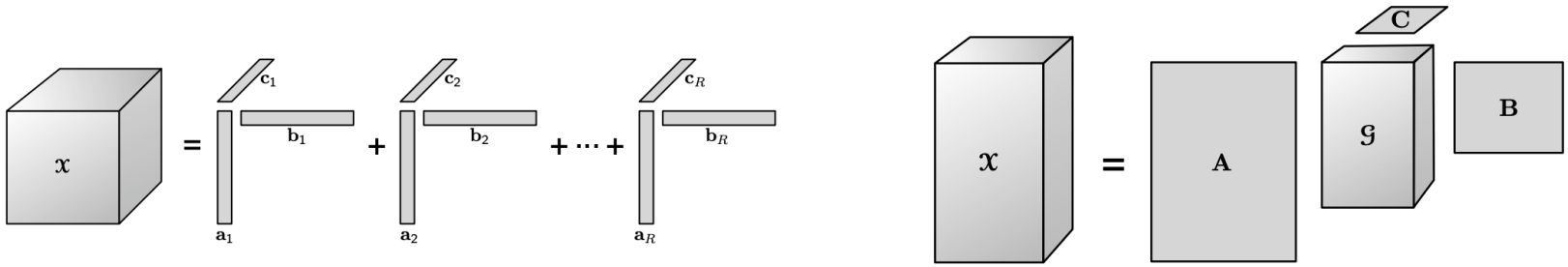
NSF

http://lpna.cs.illinois.edu

Outline

Tensor Decompositions

- 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
- Gauss-Newton method is an emerging alternative

[1] Kolda and Bader, SIAM Review 2009

Applications of CP Decomposition

- CP and Tucker are both used for data compression
- In quantum chemistry, CP decomposition is used to obtain tensor hypercontraction (THC) format

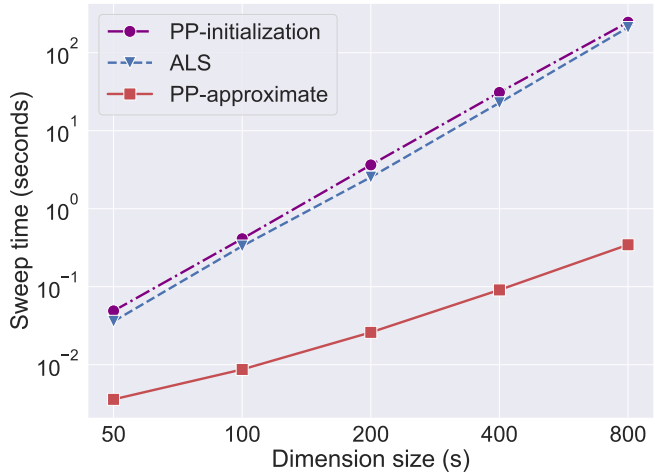
$$t_{abij} = \underbrace{\sum_{s=1}^{P} d_{abs} d_{sij}}_{\text{Cholesky}}, \qquad d_{abs} = \underbrace{\sum_{r=1}^{R} u_{ar} u_{br} v_{sr}}_{\text{CP with repeating factor}}$$

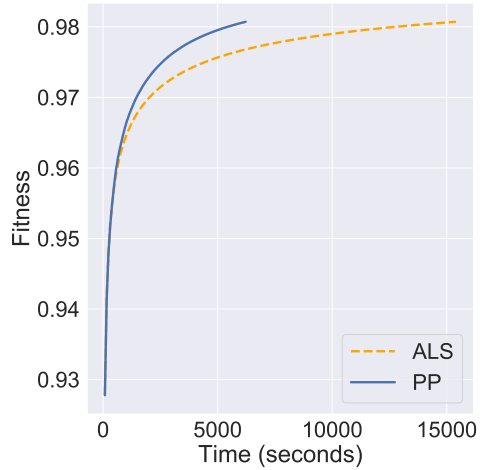
  - THC asymptotically reduces cost of post-Hartree-Fock methods
- CP can be used to find fast bilinear algorithms, such as Strassen's matrix multiplication algorithm ($s = 4$, $R = 7$),

$$t_{ijklmn} = \delta_{lm}\delta_{ik}\delta_{nj} \quad \text{so} \quad c_{ij} = \sum_{klmn} t_{ijklmn} a_{kl} b_{mn} = \sum_{l} a_{il} b_{lj}$$

$$t_{ijklmn} = \sum_{r=1}^{R} u_{ijr} v_{klr} w_{mnr} \;\Rightarrow\; c_{ij} = \sum_{r=1}^{R} u_{ijr} \Big(\sum_{kl} v_{klr} a_{kl} \Big) \Big(\sum_{mn} w_{mnr} b_{mn} \Big)$$

Accelerating Alternating Least Squares





New algorithm: pairwise perturbation (PP) approximates ALS

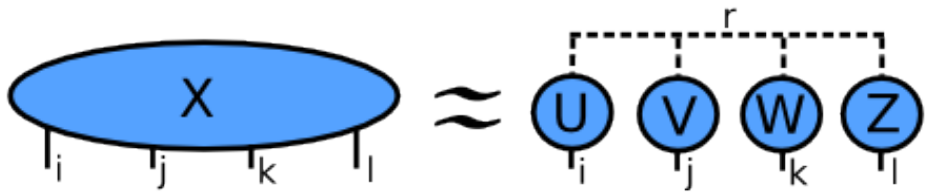
- accurate when ALS sweep over factor matrices stagnates
- rank $R < s^{N-1}$ CP decomposition:
  - ALS sweep cost $O(s^N R) \Rightarrow O(s^2 R)$, up to 600x speed-up
- rank $R < s$ Tucker decomposition:
  - ALS sweep cost $O(s^N R) \Rightarrow O(s^2 R^{N-1})$

Linjian Ma

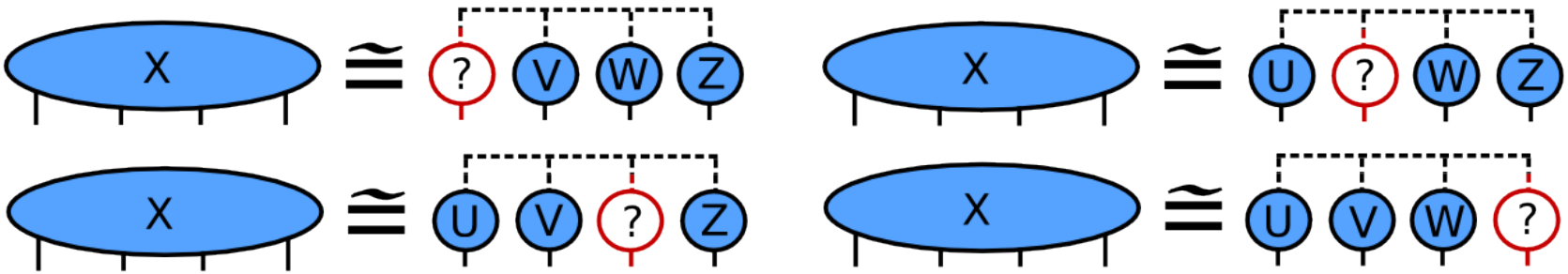
Alternating Least Squares for CP Decomposition

Consider rank R CP decomposition of an $s \times s \times s \times s$ tensor

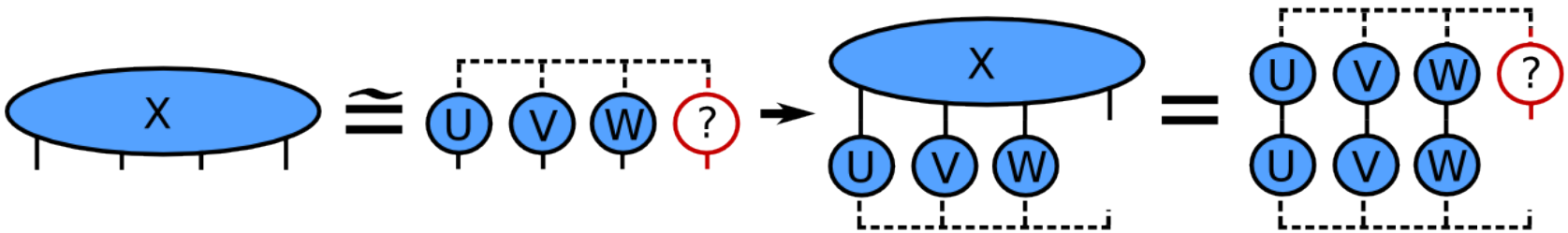
$$x_{ijkl} \approx \sum_{r=1}^{R} u_{ir} v_{jr} w_{kr} z_{lr}$$



ALS updates factor matrices in an alternating manner



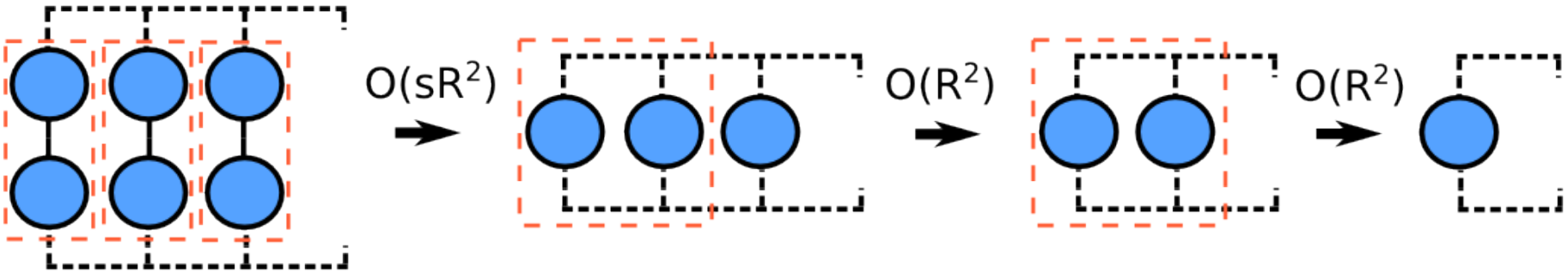
Each quadratic subproblem is typically solved via normal equations



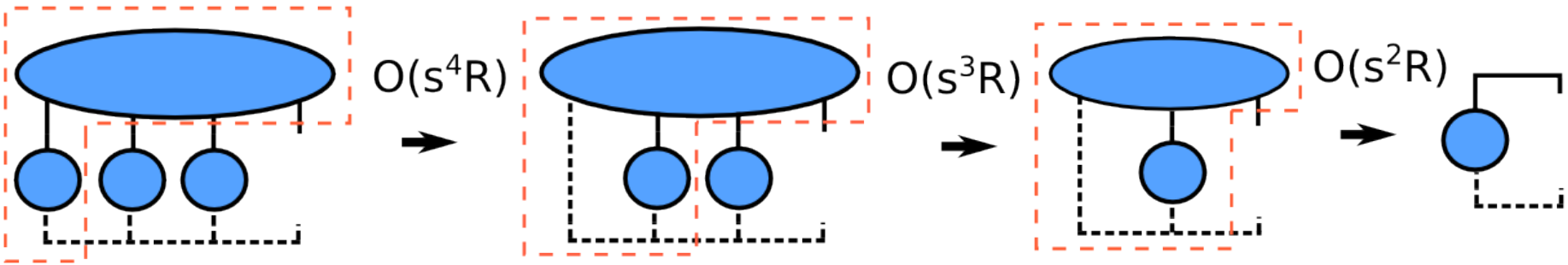
$$\boldsymbol{X} \cong \boldsymbol{A}\boldsymbol{Y} \qquad \Rightarrow \qquad \boldsymbol{A}^T\boldsymbol{X} = \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{Y}$$

Tensor Contractions in CP ALS

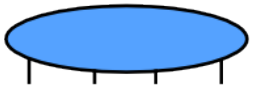
The normal equations matrix is cheap to compute

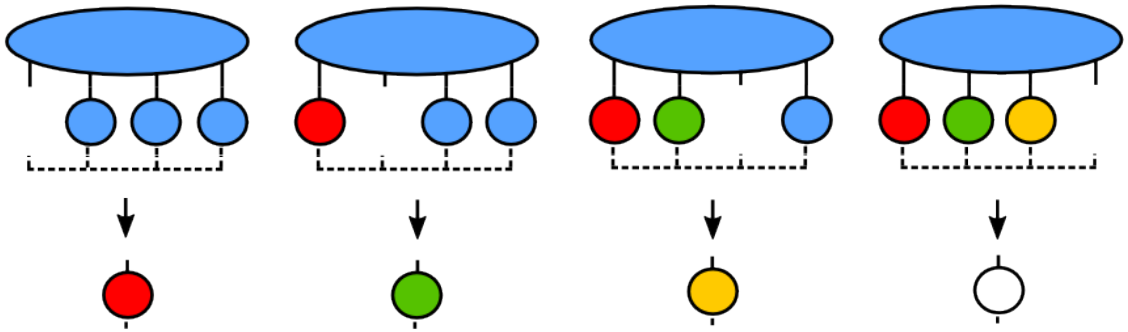


But forming the right-hand sides requires expensive MTTKRP (matricized tensor-times Khatri-Rao product)



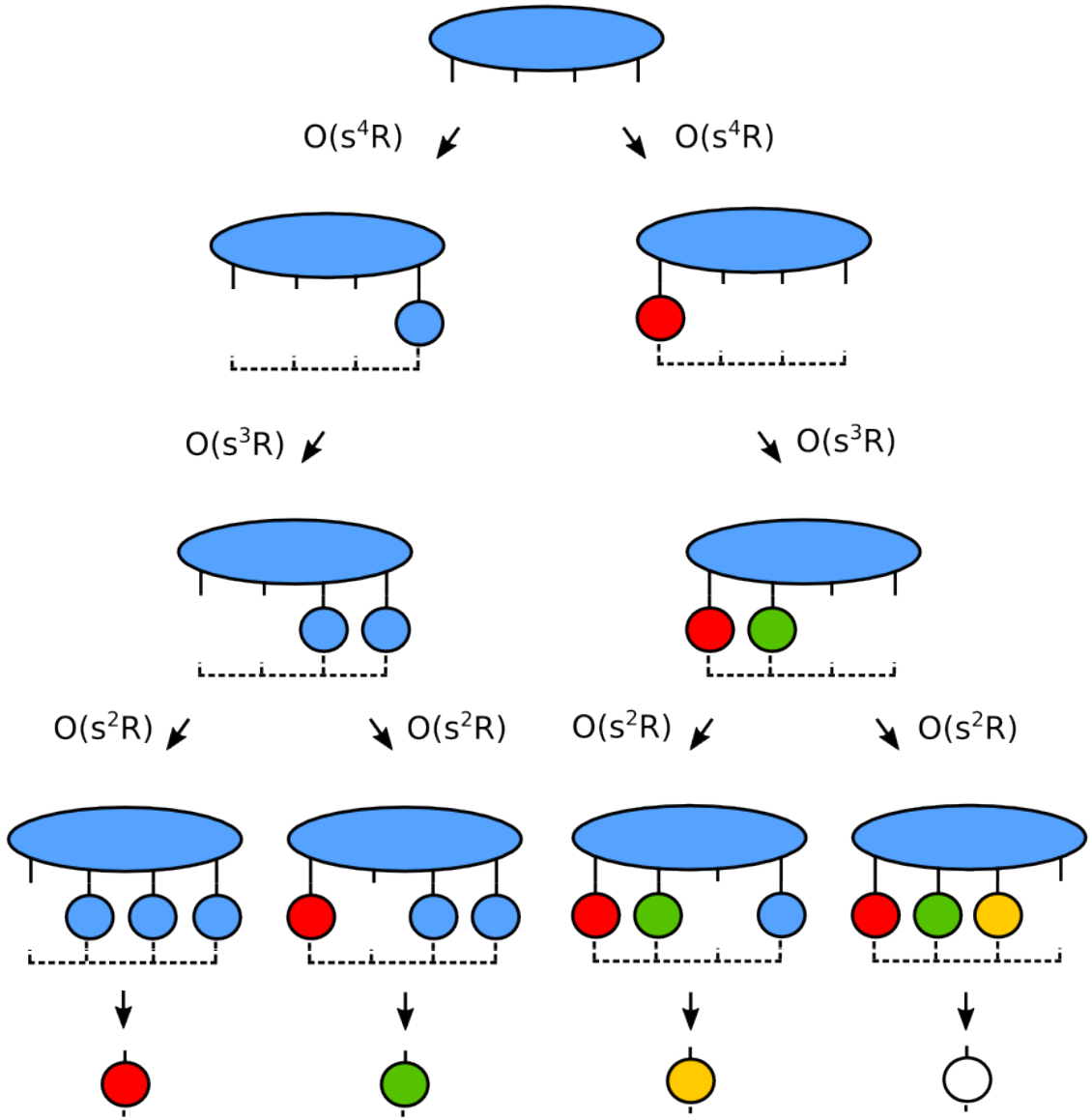
CP ALS Dimension Trees[2]





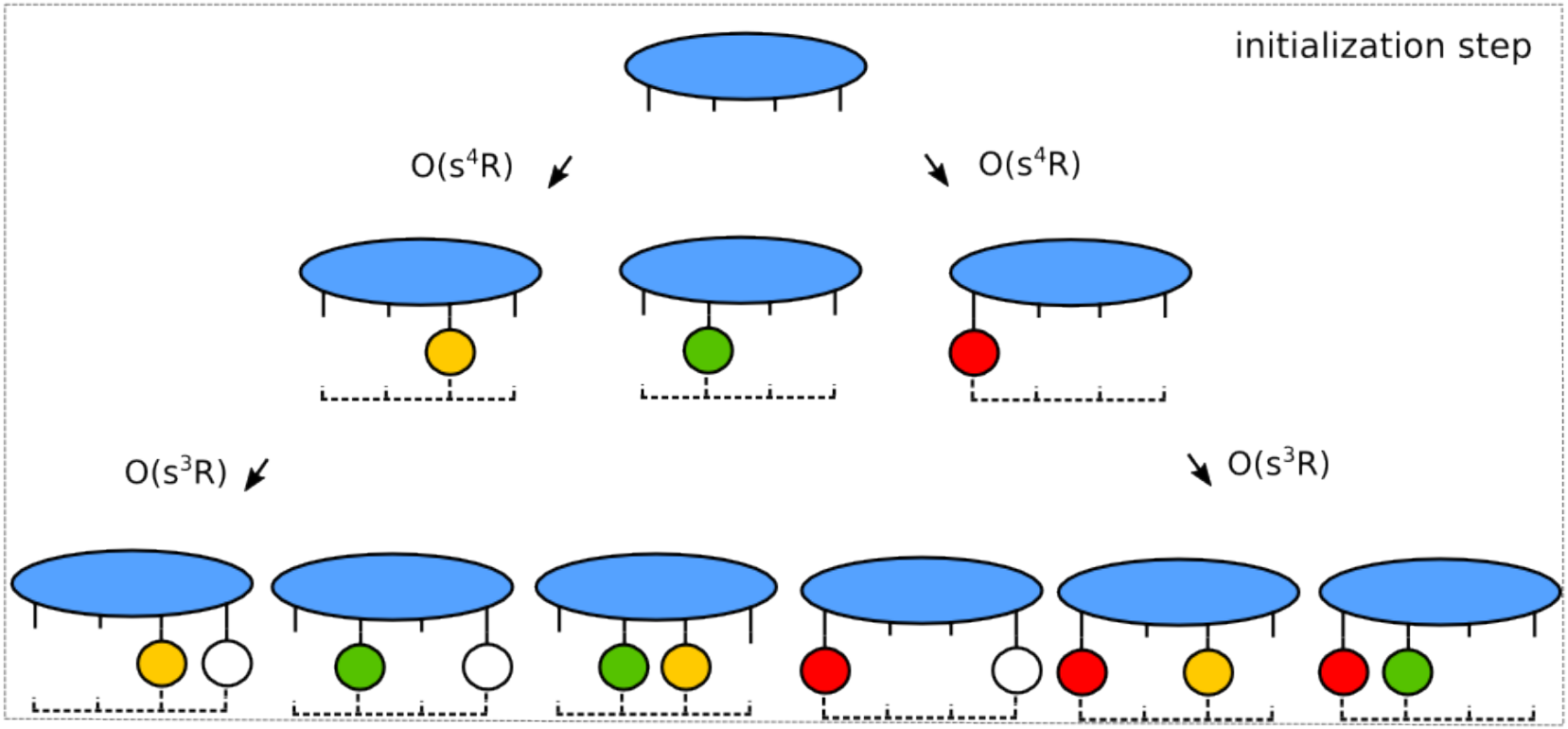
[2] Phan, Tichavskỳ, and Cichocki, IEEE Transactions on Signal Processing 2013

CP ALS Dimension Trees[3]

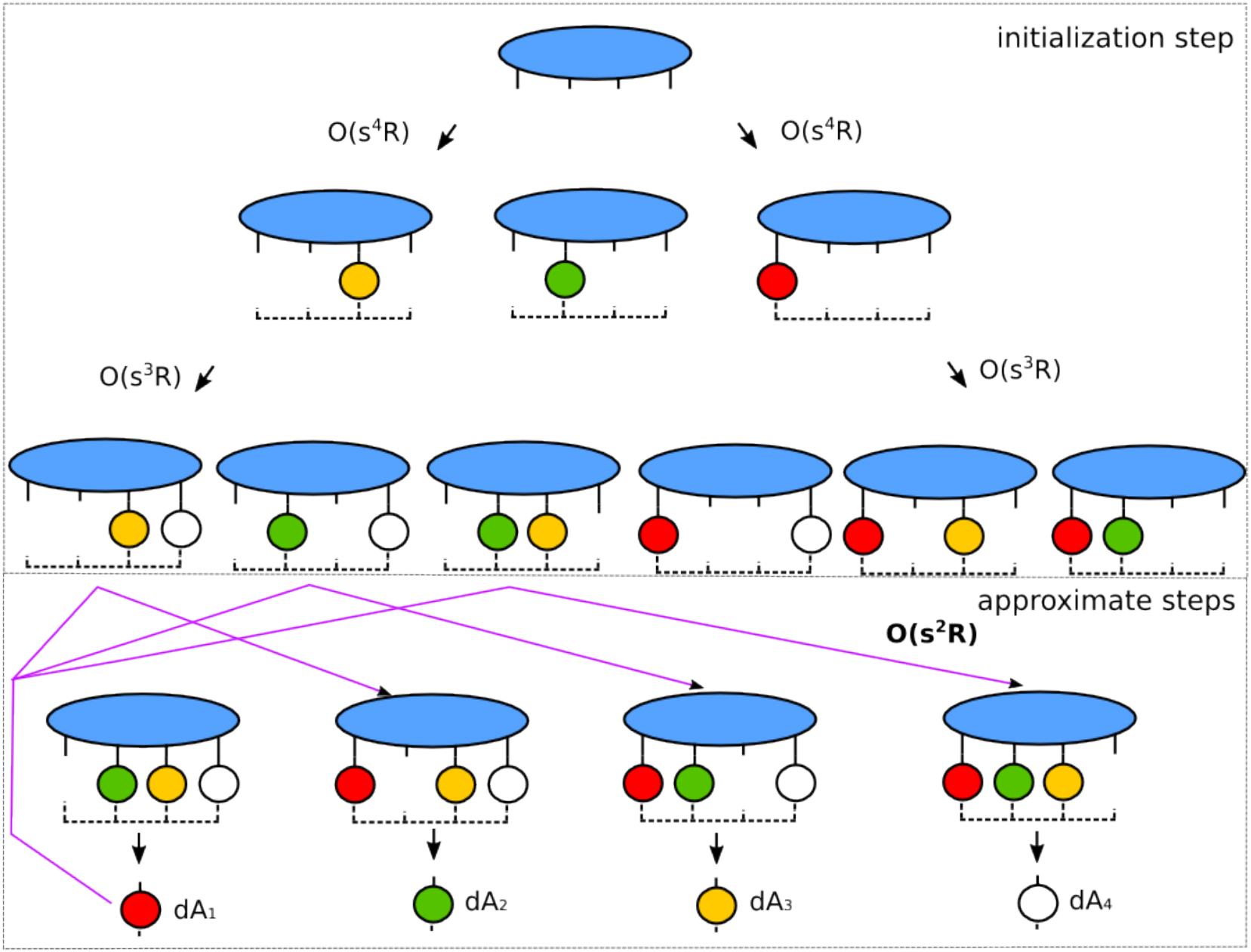


[3] Phan, Tichavskỳ, and Cichocki, IEEE Transactions on Signal Processing 2013

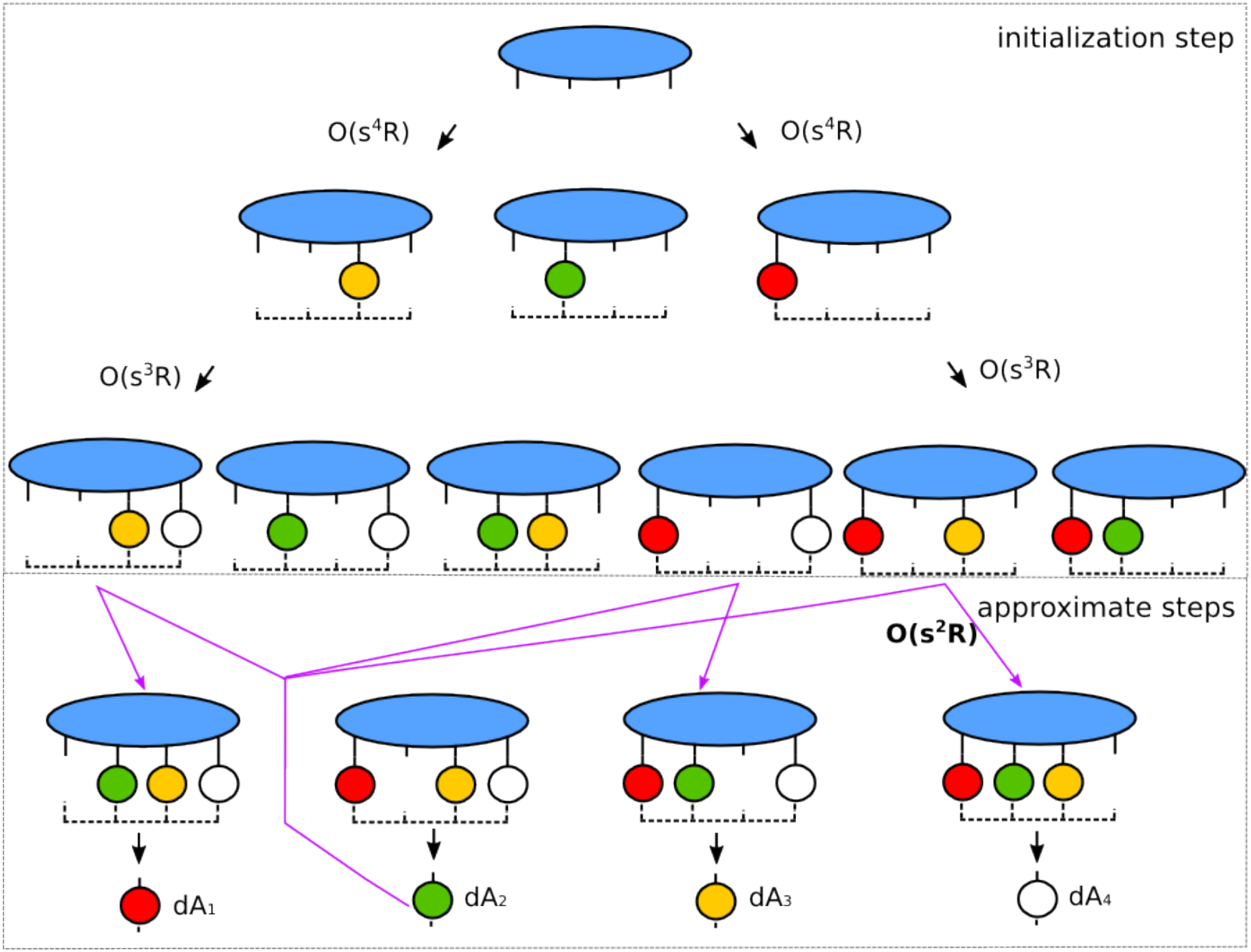
CP ALS with Pairwise Perturbation



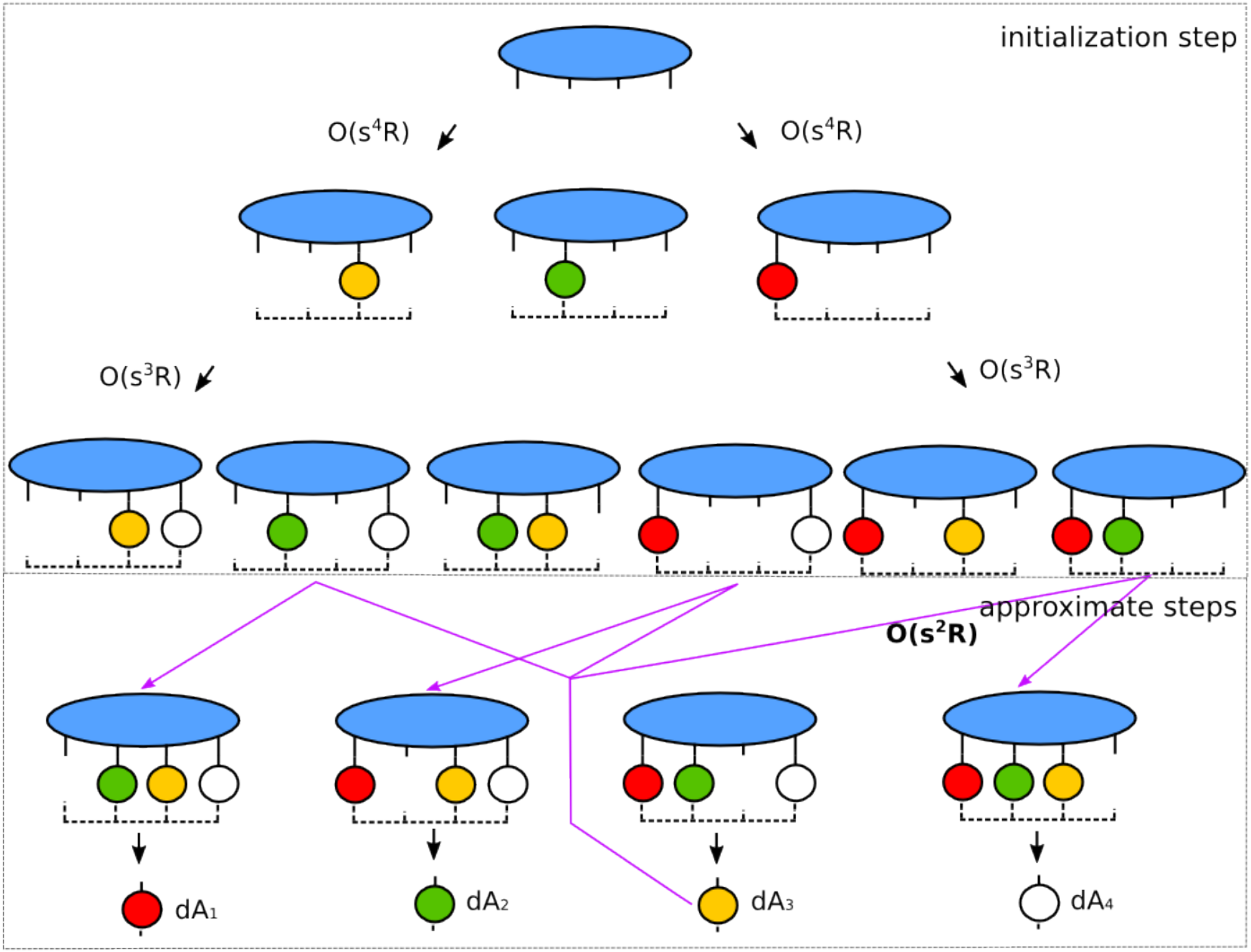
CP ALS with Pairwise Perturbation



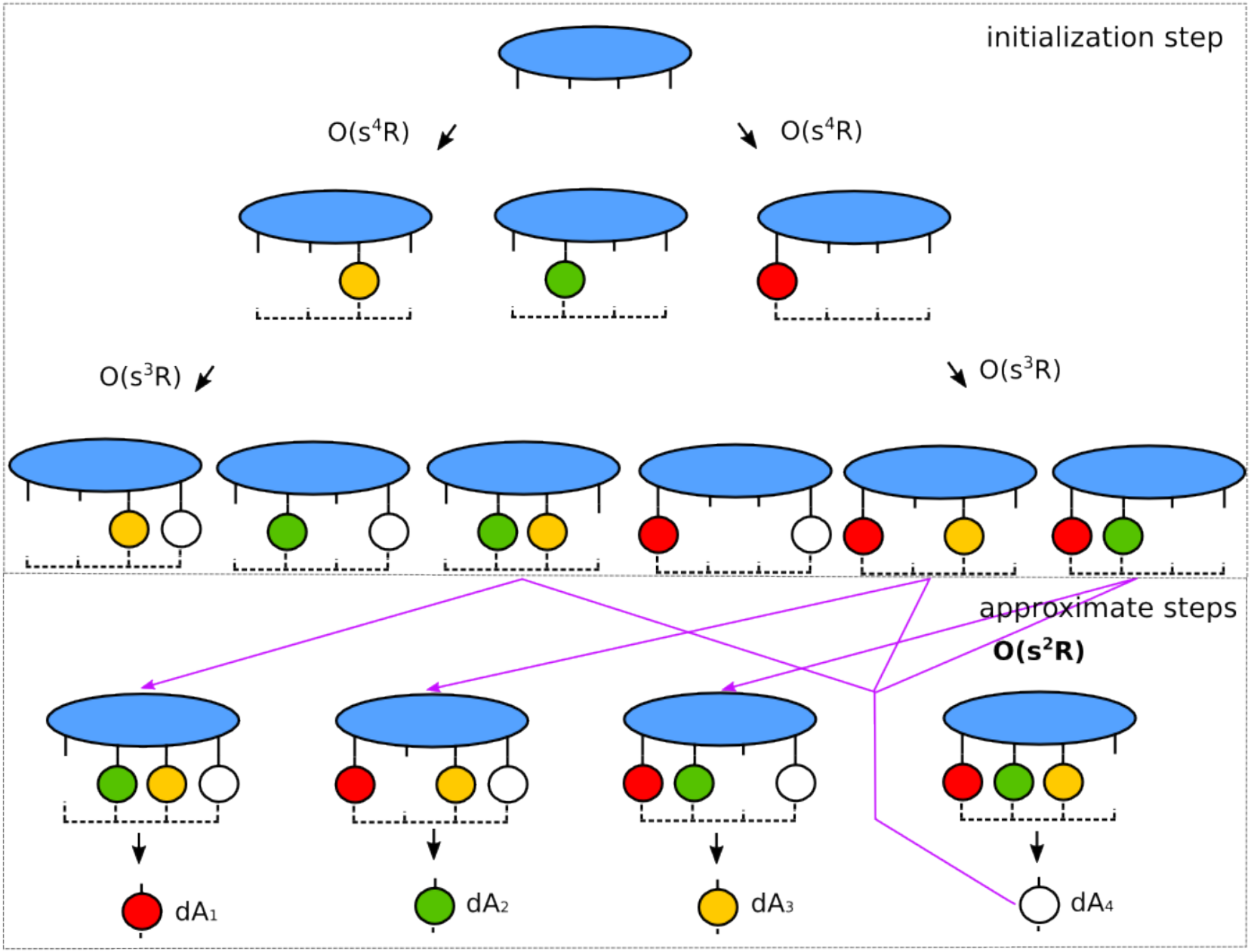
CP ALS with Pairwise Perturbation



CP ALS with Pairwise Perturbation



CP ALS with Pairwise Perturbation



Error Analysis

- Can derive columnwise worst case error bound with respect to ALS
- For simplicity, assume $N = 3$ and $R = 1$, so that

$$\boldsymbol{\mathcal{T}} \approx \boldsymbol{a}^{(1)} \times \boldsymbol{a}^{(2)} \times \boldsymbol{a}^{(3)}$$

- Updating $\boldsymbol{a}^{(n)}$ by $\delta\boldsymbol{a}^{(n)}$ yields update $\boldsymbol{h}^{(m,n)}$ to mth factor matrix
- $\boldsymbol{h}^{(1,3)} = \boldsymbol{R}^{(3)}\boldsymbol{a}^{(2)}$ where $\boldsymbol{R}^{(3)} \in \mathbb{R}^{s \times s}$ is $\boldsymbol{\mathcal{T}}$ contracted along the last mode with with $\delta\boldsymbol{a}^{(3)}$
- PP approximate step performs update $\tilde{\boldsymbol{h}}^{(1,3)}$ with relative error

$$\frac{\|\tilde{\boldsymbol{h}}^{(1,3)} - \boldsymbol{h}^{(1,3)}\|_2}{\|\boldsymbol{h}^{(1,3)}\|_2} \leq \kappa(\boldsymbol{R}^{(3)})\|\delta\boldsymbol{a}^{(2)}\|_2$$

- where $\delta\boldsymbol{a}^{(2)}$ is the change to $\boldsymbol{a}^{(2)}$ since PP initialization

Implementation

We used NumPy and Cyclops Tensor Framework[4] to implement standard dimension tree ALS and pairwise perturbation

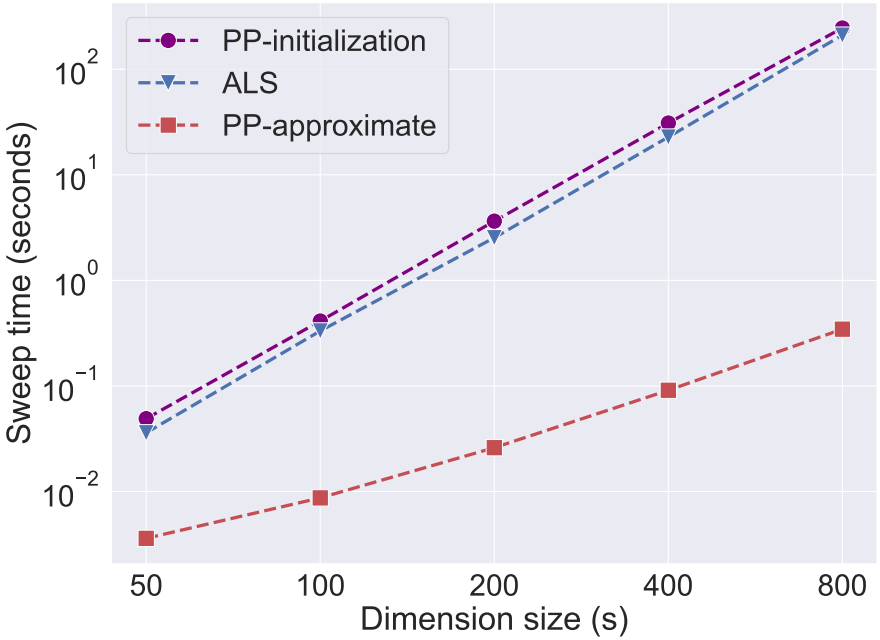
- Cyclops is a C++ library that distributes each tensor over MPI
- Used in chemistry (PySCF, QChem), quantum circuit simulation (IBM/LLNL), and graph analysis (betweenness centrality)
- Summations and contractions specified via Einstein notation

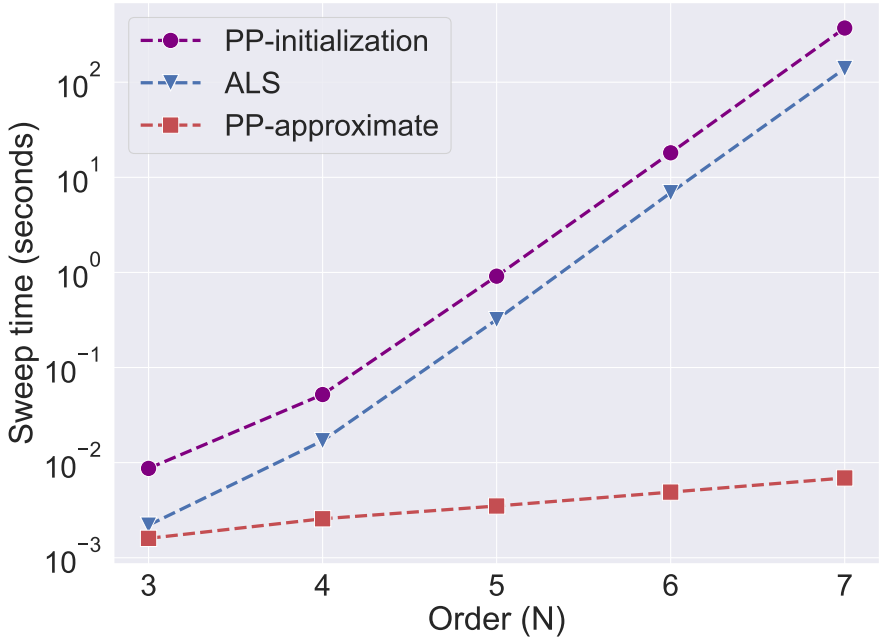
```
E["aixbjy"]+=X["aixbjy"]-U["abu"]*V["iju"]*W["xyu"]
```

- Best distributed contraction algorithm auto-selected at runtime
- Python interface, OpenMP, and GPU support present but unused
- Used interface to ScaLAPACK to solve linear systems

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

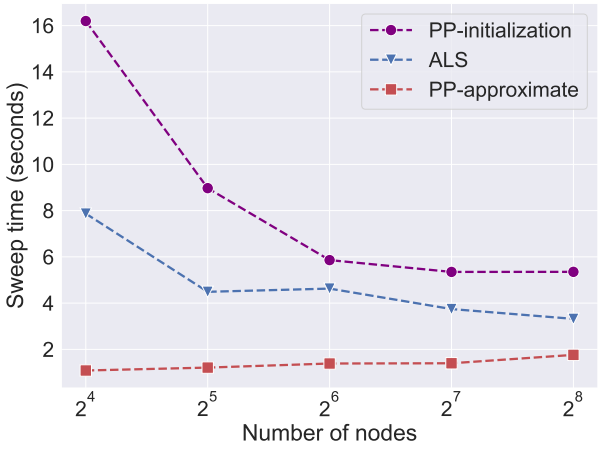
Pairwise Perturbation Performance



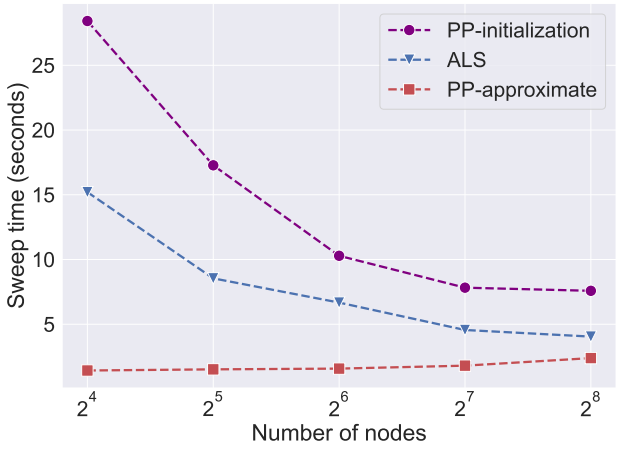


- Performance of pairwise perturbation (PP) initialization costs a bit more than an ALS sweep
- PP approximate sweep costs orders of magnitude less for large dimension s or larder order N

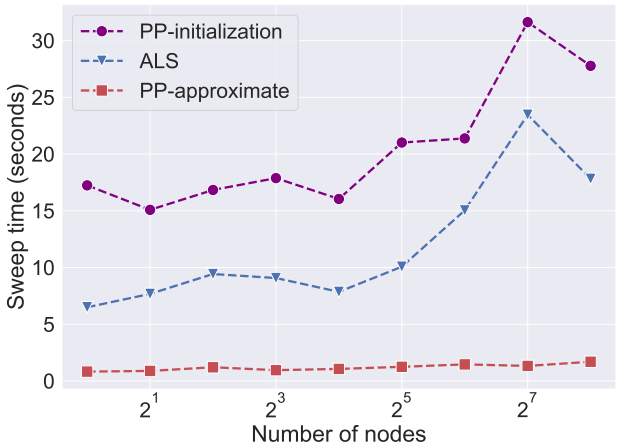
Pairwise Perturbation Parallel Scaling



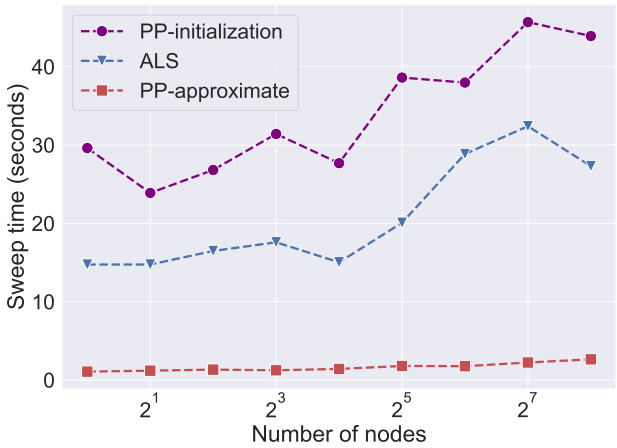
(a) Strong scaling of CP decomposition



(b) Strong scaling of Tucker

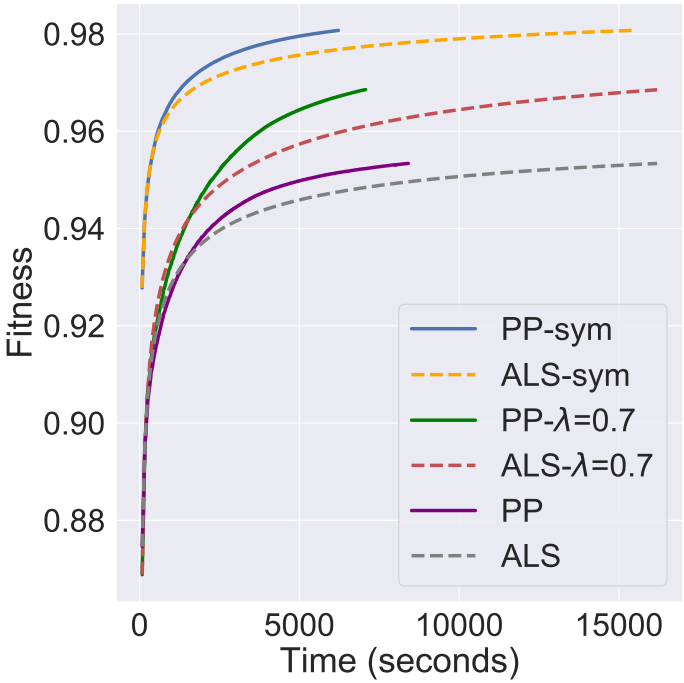


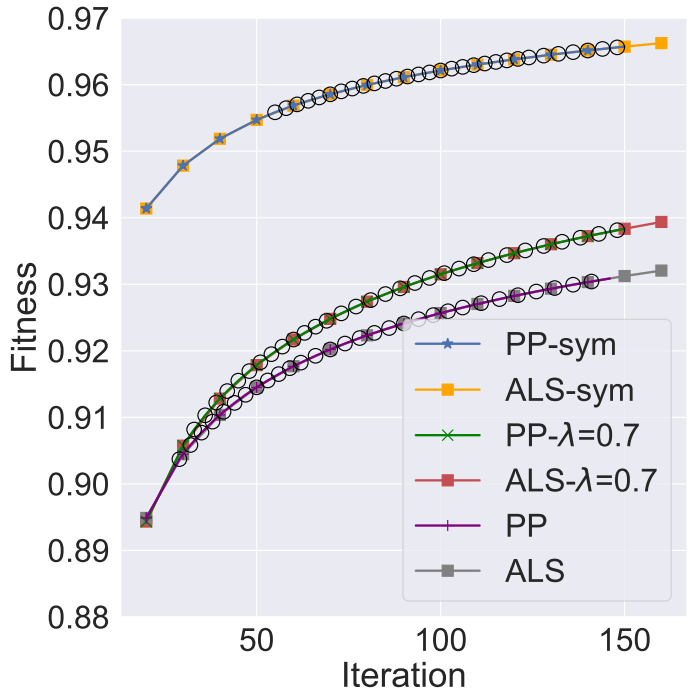
(c) Weak scaling of CP decomposition



(d) Weak scaling of Tucker

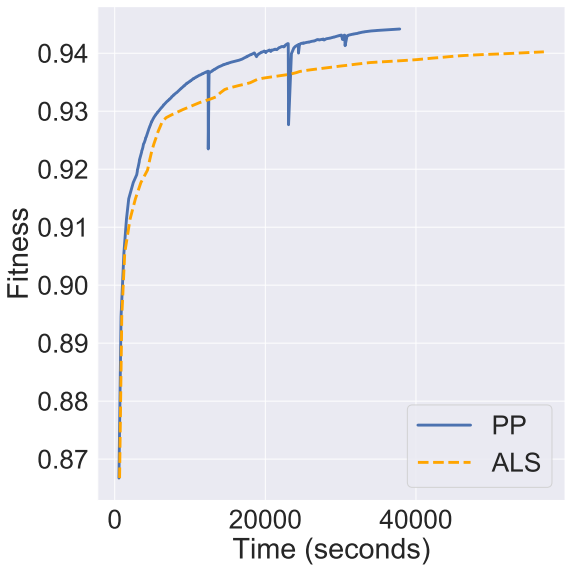
Sequential PP Performance for Quantum Chemistry Tensor

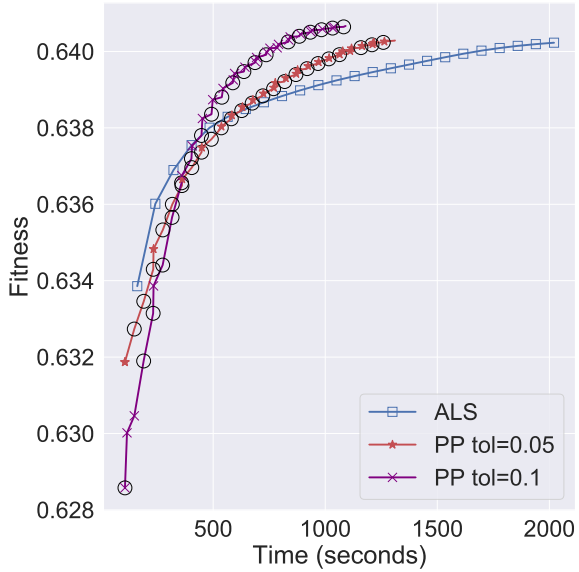


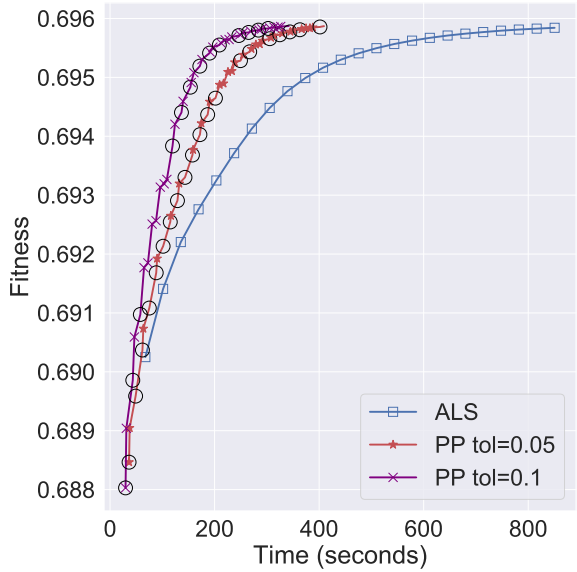


- $900 \times 56 \times 56$ tensor with $R = 1000$ CP
- Best fitness achieved by enforcng symmetry (repeating factor matrix)
- Step-size regularization helps PP
- In right plot, squares are 10 sweeps of ALS, circles are PP (re)initializations

Parallel PP Performance

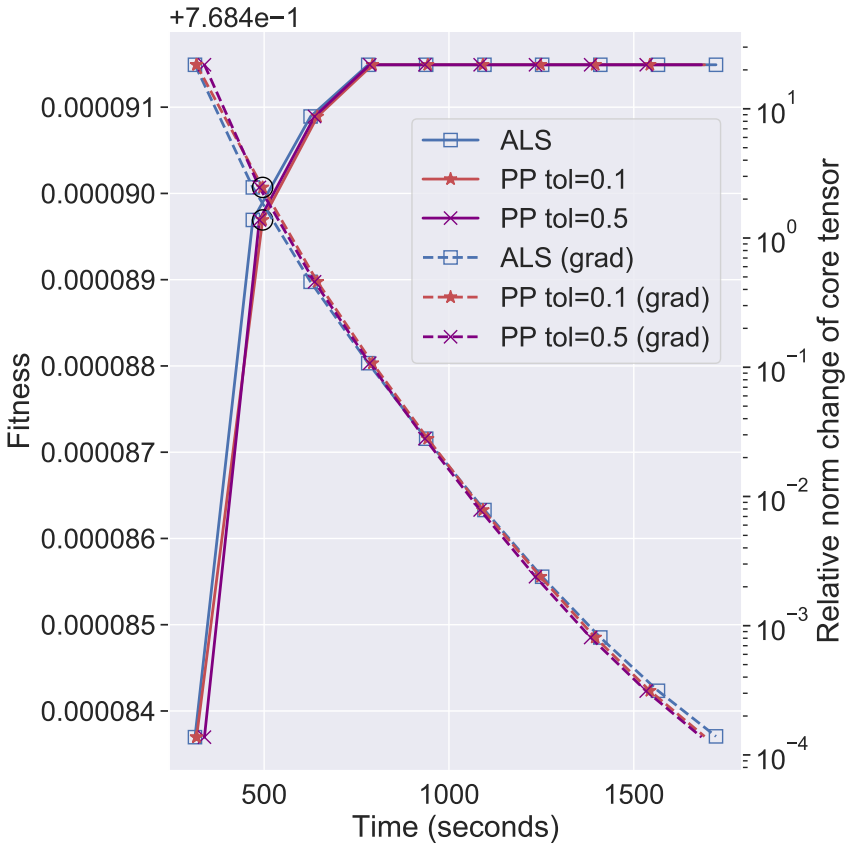


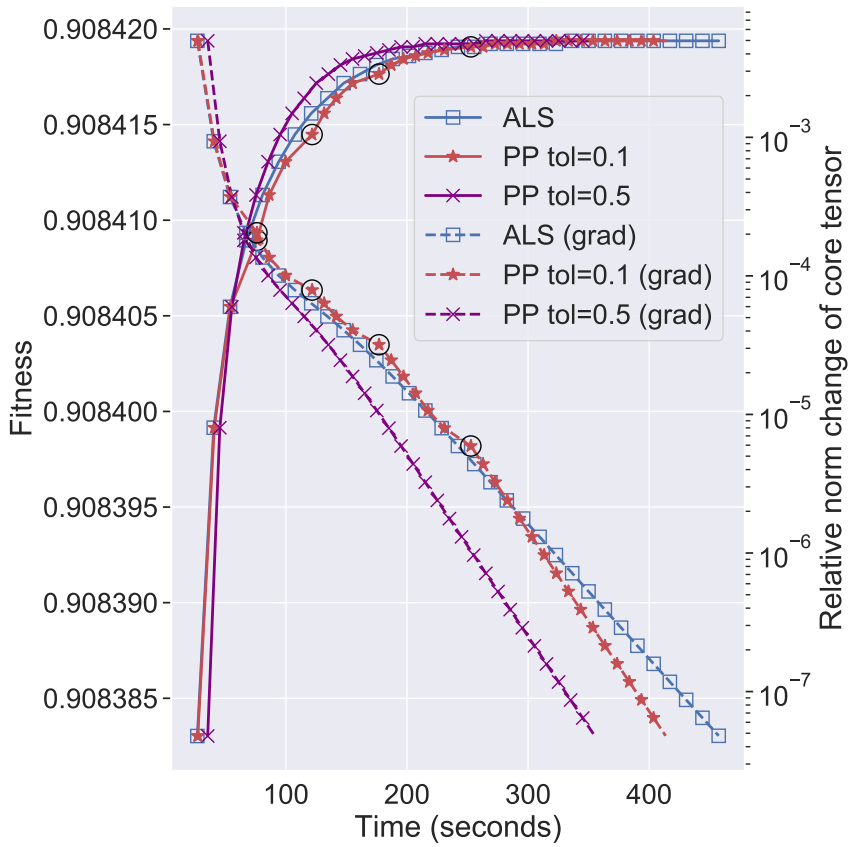




- $4520 \times 280 \times 280$ quantum chemistry tensor with $R = 2000$ CP
- $128 \times 128 \times 3 \times 7200$ image-recognition data with $R = 10$ CP
- $1024 \times 1344 \times 33 \times 9$ hyperspectral imaging dataset with $R = 10$ CP

Parallel PP Tucker Decomposition





- Performance of Tucker decomposition for image processing datasets
- $128 \times 128 \times 3 \times 7200$ tensor with rank $10 \times 10 \times 3 \times 70$ Tucker.
- $1024 \times 1344 \times 33 \times 9$ tensor with rank $100 \times 100 \times 10 \times 5$ Tucker

Gauss-Newton Algorithm

- Newton-like converge quadratically while ALS converges linearly
- Nonlinear least squares problems like CP decomposition minimize

$$\phi(\boldsymbol{x}) = \frac{1}{2} \| \underbrace{\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x})}_{\boldsymbol{r}(\boldsymbol{x})} \|^2$$

- Newton's method computes $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \boldsymbol{H}_\phi(\boldsymbol{x})^{-1} \nabla \phi(\boldsymbol{x})$
- For nonlinear least squares problems, the gradient and Hessian are

$$\nabla \phi(\boldsymbol{x}) = \boldsymbol{J}_r^T(\boldsymbol{x}) \boldsymbol{r}(\boldsymbol{x}),$$
$$\boldsymbol{H}_\phi(\boldsymbol{x}) = \boldsymbol{J}_r^T(\boldsymbol{x}) \boldsymbol{J}_r(\boldsymbol{x}) + \sum_i r_i(\boldsymbol{x}) \boldsymbol{H}_{r_i}(\boldsymbol{x})$$

- The Gauss-Newton method approximates $\boldsymbol{H}_\phi(\boldsymbol{x}) \approx \boldsymbol{J}_r^T(\boldsymbol{x}) \boldsymbol{J}_r(\boldsymbol{x})$, so

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \boldsymbol{s}^{(k)}, \quad \boldsymbol{s}^{(k)} = (\boldsymbol{J}_r^T(\boldsymbol{x}^{(k)}) \boldsymbol{J}_r(\boldsymbol{x}^{(k)}))^{-1} \boldsymbol{J}_r^T(\boldsymbol{x}^{(k)}) \boldsymbol{r}(\boldsymbol{x}^{(k)}),$$
$$\boldsymbol{J}_r(\boldsymbol{x}^{(k)}) \boldsymbol{s}^{(k)} \cong \boldsymbol{r}(\boldsymbol{x}^{(k)})$$

Gauss-Newton for CP Decomposition

- CP decomposition for $N = 3$ ($N > 3$ is similar) minimizes

$$\phi(\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \boldsymbol{A}^{(3)}) = \frac{1}{2}\Big(\sum_{ijk} \big(t_{ijk} - \sum_{r=1}^{R} a_{ir}^{(1)} a_{jr}^{(2)} a_{kr}^{(3)}\big)^2 \Big)$$

- The Gauss-Newton approximate Hessian is $NsR \times NsR$,

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}^{(1,1)} & \dots & \boldsymbol{H}^{(1,N)} \\ \vdots & \ddots & \vdots \\ \boldsymbol{H}^{(N,1)} & \dots & \boldsymbol{H}^{(N,N)} \end{bmatrix}, \text{ where } \boldsymbol{H}^{(n,n)} = \boldsymbol{G}^{(n,n)} \otimes \boldsymbol{I}$$

while for $n \neq p$, $\quad h_{krlz}^{(n,p)} = a_{kz}^{(n)} a_{lr}^{(p)} g_{rz}^{(n,p)}$,

where in both cases $\quad g_{rz}^{(n,p)} = \prod_{m=1, m\neq n,p}^{N} \left(\sum_i a_{ir}^{(m)} a_{iz}^{(m)} \right)$

Gauss-Newton for CP Decomposition

- A step of Gauss-Newton requires solving a linear system with $\boldsymbol{H}$
- Cholesky of $\boldsymbol{H}$ requires $O(N^2 s^2 R^2)$ memory and cost $O(N^3 s^3 R^3)$
- Matrix-vector product with $\boldsymbol{H}$ can be computed with cost $O(N^2 s R^2)$
- Can use CG method with implicit matrix-vector product[5]
- We formulate product $\boldsymbol{u} = \boldsymbol{H}\boldsymbol{v}$ using tensor contractions as

```
u = []
for n in range(N):
  u.append(zeros((s,R)))
  for p in range(N):
    if n == p:
      u[n] += einsum("rz,kz->kr",G[n,p],v[p])
    else:
      u[n] += einsum("kz,lr,rz,lz->kr", \
                     A[n],A[p],G[n,p],v[p])
```

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

Regularization for Gauss-Newton

- $\boldsymbol{H}$ is inherently rank-deficient, so incorporate Tikhonov regularization,

$$(\boldsymbol{H} + \lambda \boldsymbol{I})\boldsymbol{s}^{(k)} = \boldsymbol{J}^T \boldsymbol{r}(\boldsymbol{x}^{(k)})$$

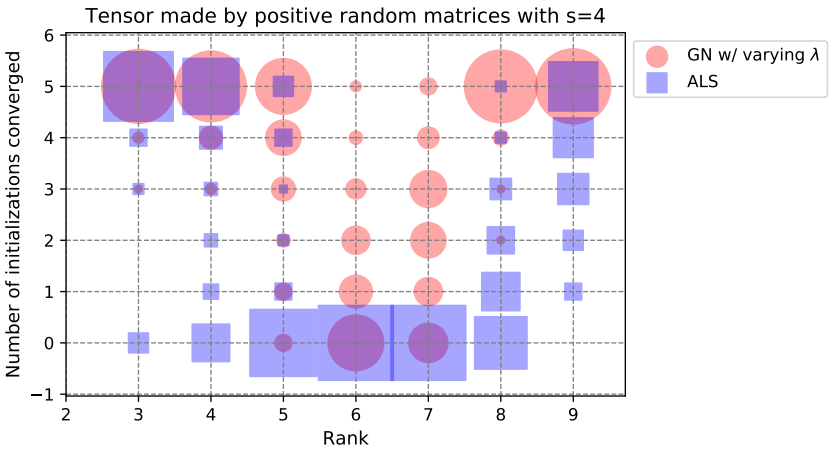
- CP Decomposition is sensitive to the λ parameter
  - initially and in swamps (regions of slow change), want large regularization to take steps toward negative gradient diretion
  - near minima, want small regularization to achieve quadratic convergence

We propose a new approach for choosing λ

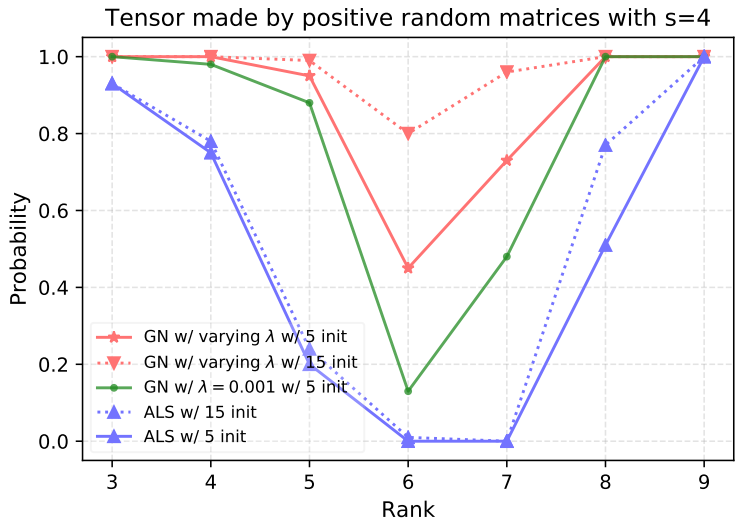
1. Initialize λ near upper threshold and choose $\mu > 1$
2. Update λ to $\lambda \leftarrow \lambda/\mu$ until lower threshold
3. Now, reverse and update $\lambda = \lambda\mu$ until upper threshold
4. Go back to (2) and repeat

Navjot Singh

Convergence results



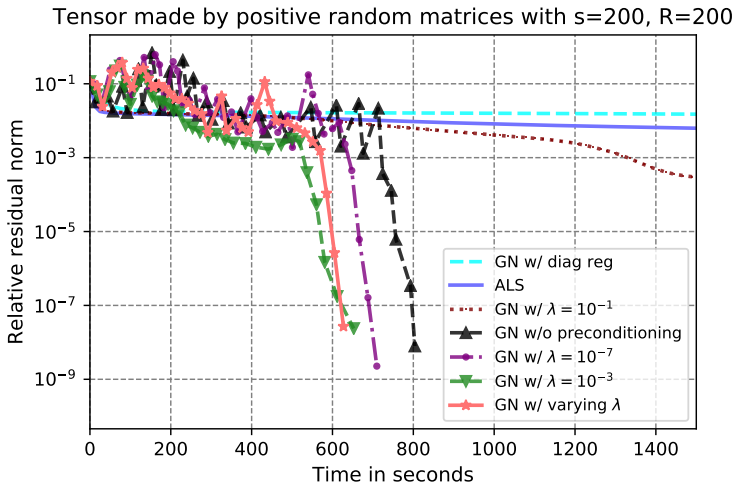
(e) Probability 1



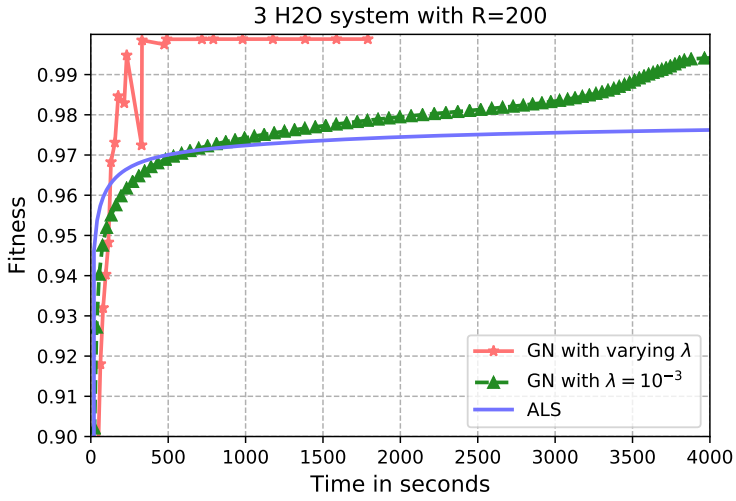
(f) Probability 2

- Probability 1: Diameter of circle/side of square corresponds to number of problems converged
- Probability 2: Each point in the plot corresponds to probability of at least one of given initializations converging

Sequential Execution Results



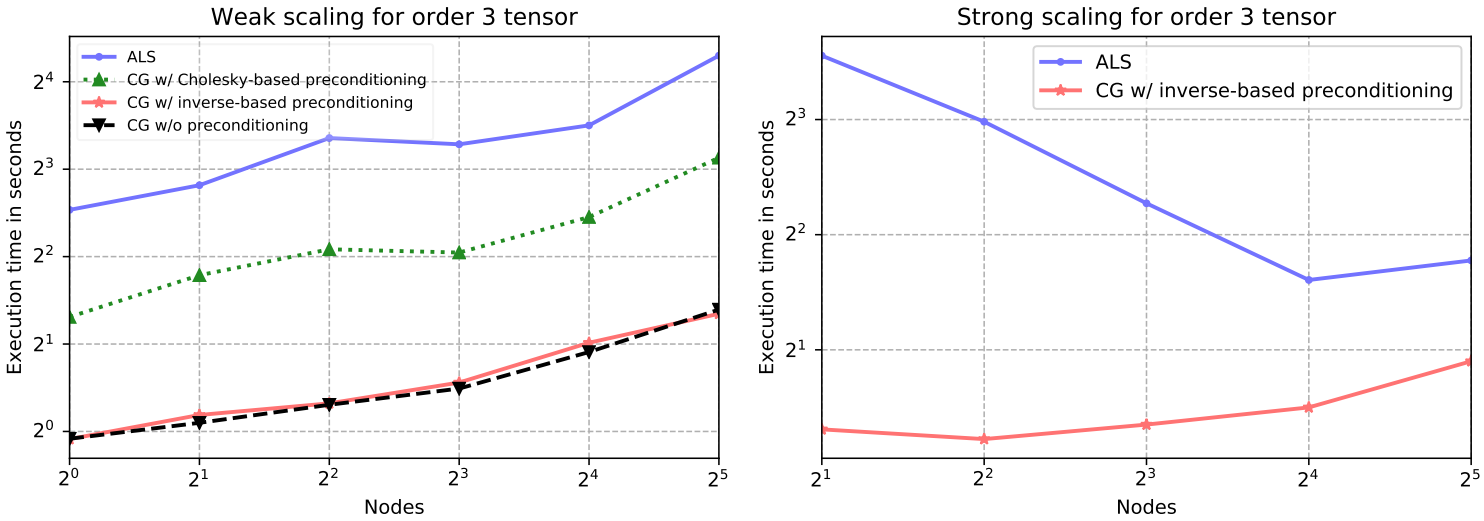
(e) Random Tensor



(f) Quantum Chemistry Tensor

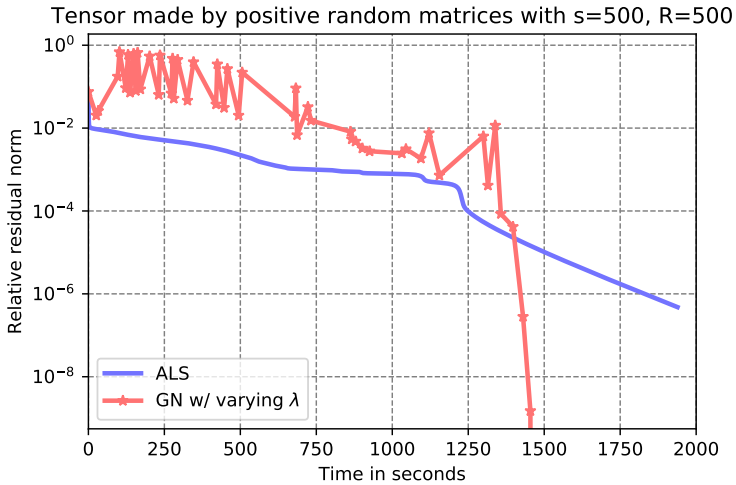
- $200 \times 200 \times 200$ random tensor with exact rank 200 CP
- $339 \times 21 \times 21$ quantum chemistry tensor dimension rank 200 CP
- Gauss-Newton with proposed regularization outperforms all the variants of Gauss-Newton and ALS

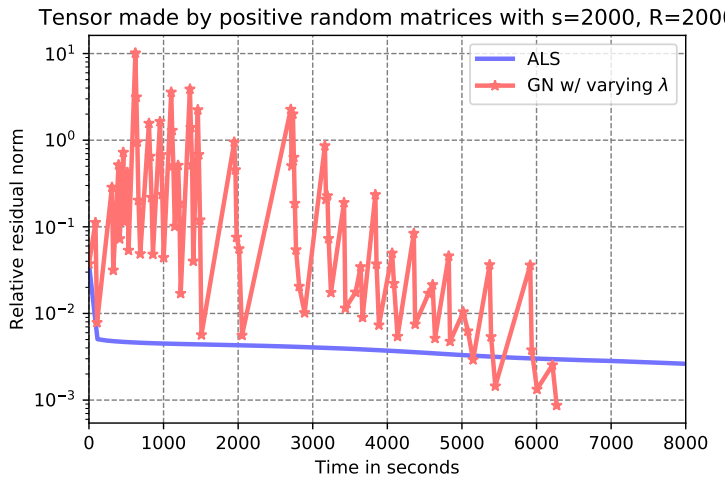
Parallel Scaling of GN and ALS



- Weak scaling with fixed memory/processor (increasing work/processor)
- Gauss-Newton with implicit CG is harder to scale than ALS

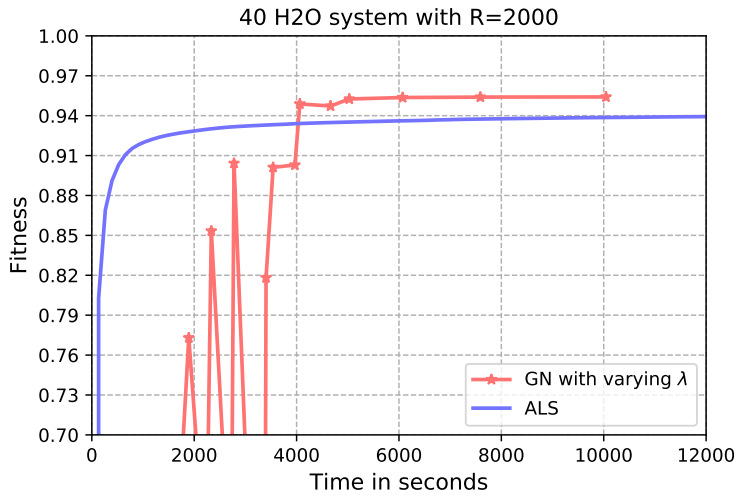
Parallel Performance of GN and ALS

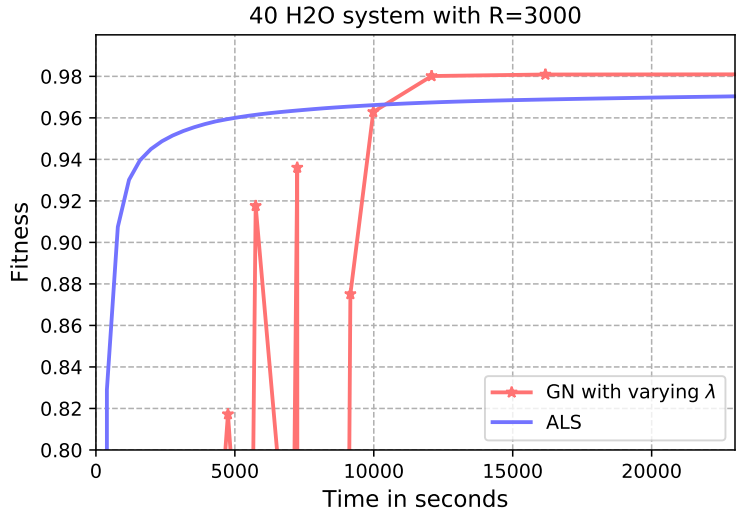




- $500 \times 500 \times 200$ random tensor with exact CP rank 500 on 4 nodes of Stampede2
- $2000 \times 2000 \times 200$ random tensor with exact CP rank 2000 on 16 nodes of Stampede2

Parallel Performance of GN and ALS





- $4520 \times 280 \times 280$ quantum chemistry tensor with $R = 2000$ CP
- $4520 \times 280 \times 280$ quantum chemistry tensor with $R = 3000$ CP

Finding Fast Matrix Multiplication Algorithms

Dimensions	Rank	p_{GN}	t_{GN}	n_{ALS}	p_{ALS}	t_{ALS}
(2,2,3)	11	9	1.54	5897	7	14.48
(2,3,3)	15	7	2.77	3619	7	9.23
(2,2,4)	14	9	2.53	3600	7	9.33
(2,2,5)	18	9	4.53	3602	2	12.45
(2,3,4)	20	4	4.89	4638	3	10.39
(3,3,3)	23	5	2.1	3685	2	8.82

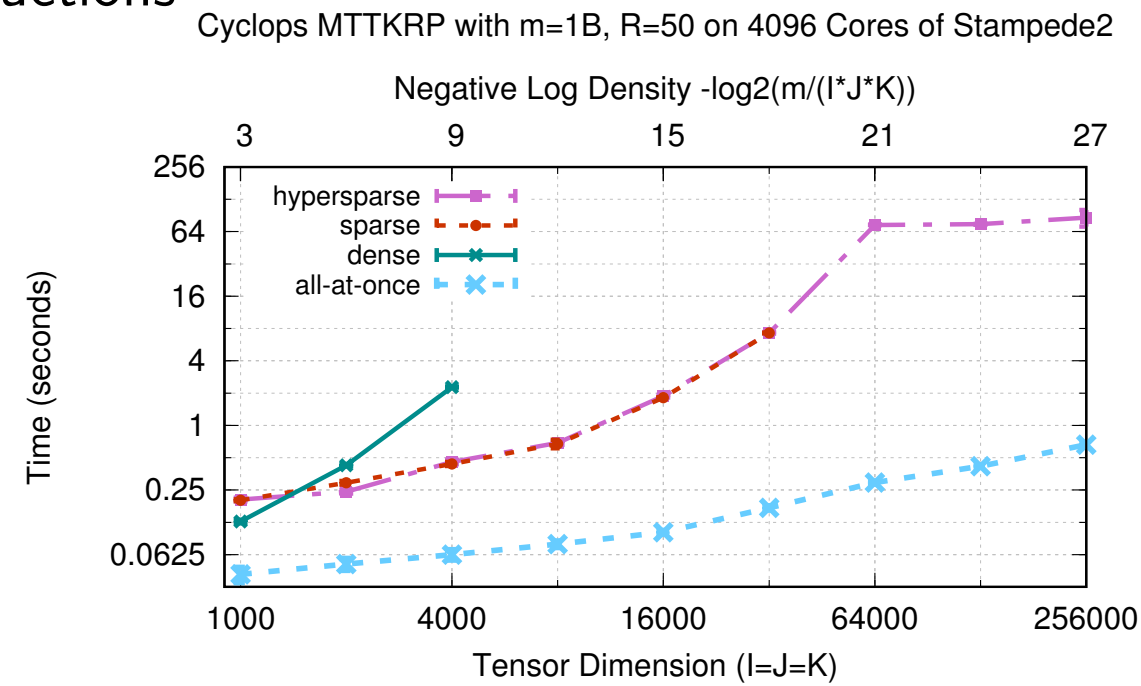
- GN initialized with a few steps of ALS
- p_{GN} and p_{ALS} is number out of 10 initializations that converge
- t_{GN} and t_{ALS} is average time in seconds to converge (for initializations that do)
- n_{ALS} is average number of ALS iterations to converge (for initializations that do)

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 with pairwise tensor contractions



Tensor Completion

- Letting $\langle \cdot, \cdot, \cdot \rangle$ denote a trilinear inner product, CP decomposition is

$$t_{ijk} = \langle \boldsymbol{u}_i, \boldsymbol{v}_j, \boldsymbol{w}_k \rangle.$$

- Tensor completion with CP decomposition is given a set of oberved entries Ω and seeks to minimize the objective function $f(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}) =$

$$\underbrace{\sum_{(i,j,k)\in\Omega} \left(t_{ijk} - \langle \boldsymbol{u}_i, \boldsymbol{v}_j, \boldsymbol{w}_k \rangle\right)^2}_{\text{Frobenius norm error on observed entries}} + \underbrace{\lambda(||\boldsymbol{U}||_F^2 + ||\boldsymbol{V}||_F^2 + ||\boldsymbol{W}||_F^2)}_{\text{regularization to prevent overfitting}}.$$

- ALS right-hand sides look as before, but equations harder to form,

$$\boldsymbol{u}_i^{(\text{new})}(\boldsymbol{G}^{(i)} + \lambda \boldsymbol{I}) = \sum_{(j,k)\in\Omega_i} (\boldsymbol{v}_j \odot \boldsymbol{w}_k) t_{ijk},$$

$$\text{where} \quad \boldsymbol{G}^{(i)} = \sum_{(j,k)\in\Omega_i} (\boldsymbol{v}_j \odot \boldsymbol{w}_k)^T (\boldsymbol{v}_j \odot \boldsymbol{w}_k).$$

Tensor Completion ALS with Implicit CG

We avoid the bottleneck of forming each $\boldsymbol{G}^{(i)}$ by using a new implicit conjugate gradient method, which computes batches of matrix vector products, $\boldsymbol{y}^{(i)} = \boldsymbol{G}^{(i)}\boldsymbol{x}^{(i)}$ via

$$y_r^{(i)} = \sum_{j,k,s} v_{jr} w_{kr} \hat{\Omega}_{ijk} v_{js} w_{ks} x_s^{(i)}.$$

where $\hat{\Omega}_{ijk} = 1$ if $(i,j,k) \in \Omega$

Can rewrite each iteration as

$$z_{ijk} = \underbrace{\hat{\Omega}_{ijk} \sum_s v_{js} w_{ks} x_s^{(i)}}_{\text{TTTP}}, \quad y_r^{(i)} = \underbrace{\sum_{j,k} v_{jr} w_{kr} z_{ijk}}_{\text{MTTKRP}}.$$

Tensor Times Tensor Product (TTTP)

- TTTP generalized SDDMM (latter motivated by matrix completion),

$$x_{i_1 \ldots i_N} = s_{i_1 \ldots i_N} \sum_{r=1}^{R} \prod_{j=1}^{N} a_{i_j r}^{(j)} = s_{i_1 \ldots i_N} \langle \boldsymbol{a}_{i_1}^{(1)}, \ldots, \boldsymbol{a}_{i_N}^{(N)} \rangle$$

- Cannot be done efficiently by pairwise tensor contraction
- Permits efficient calculation of tensor completion residual

$$r_{ijk} = t_{ijk} - \hat{\Omega}_{ijk} \sum_{r=1}^{R} u_{ir} v_{jr} w_{kr}$$

- Can also be used to accelerate calculation of standard sparse CP decomposition residual

Parallel TTTP

All-at-once TTTP can be parallelized over sparse tensor entries, similar to MTTKRP

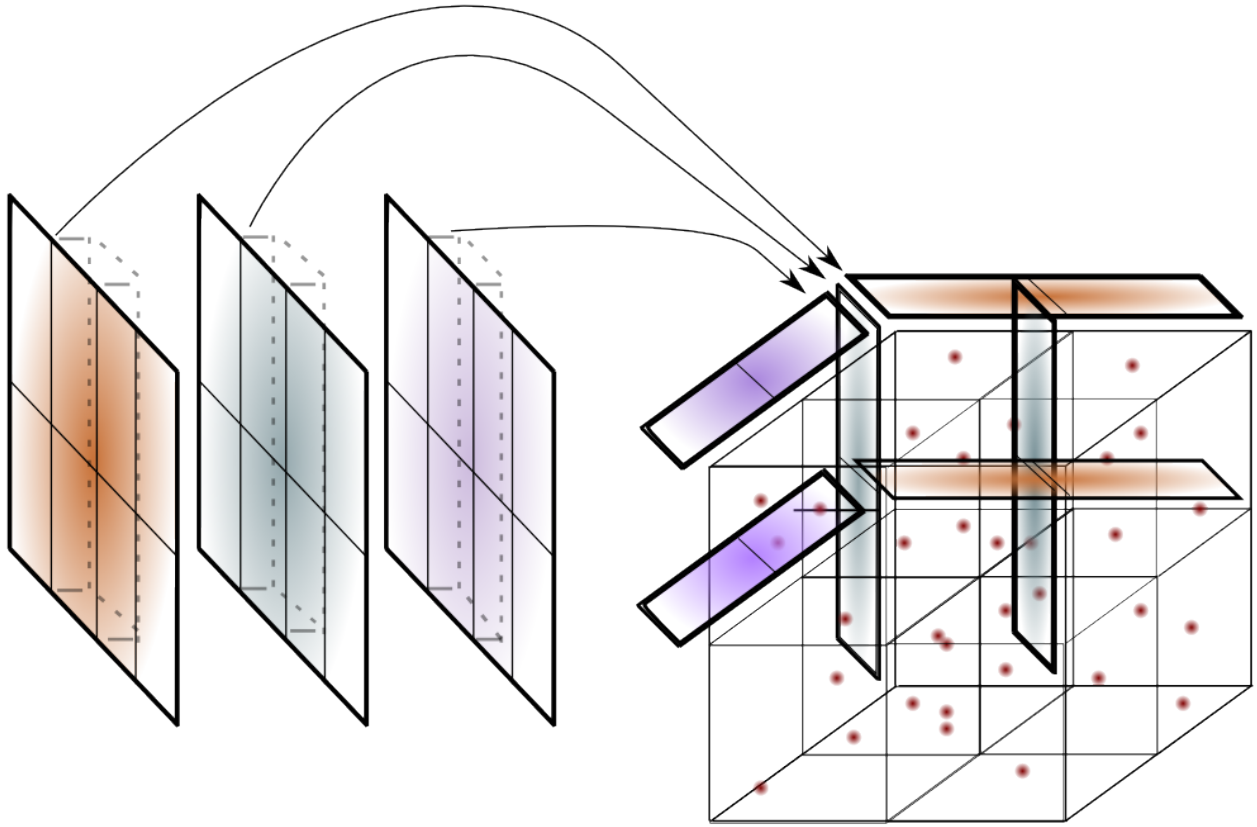
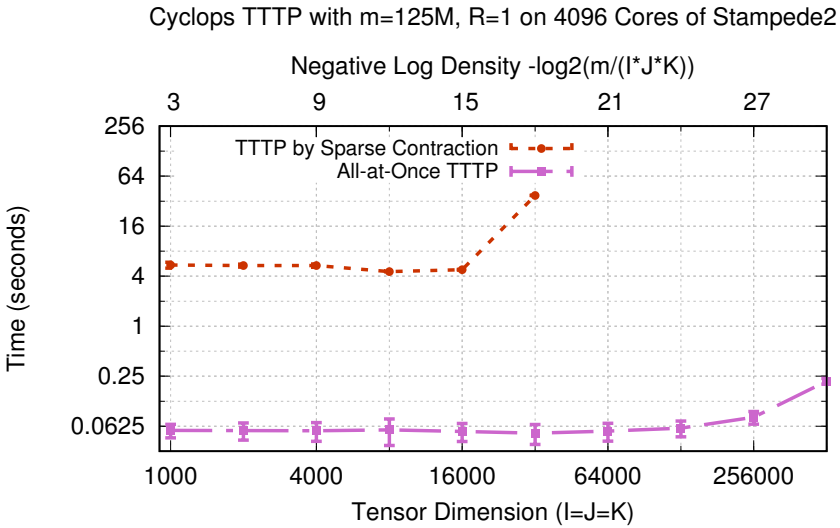
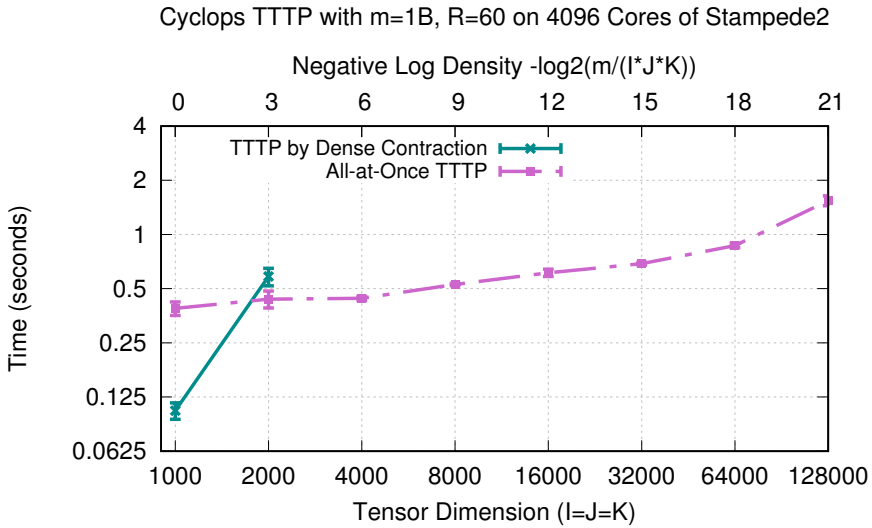


Figure: Depiction of 8 processor parallelization of TTTP computing one of four smaller TTTP substeps.

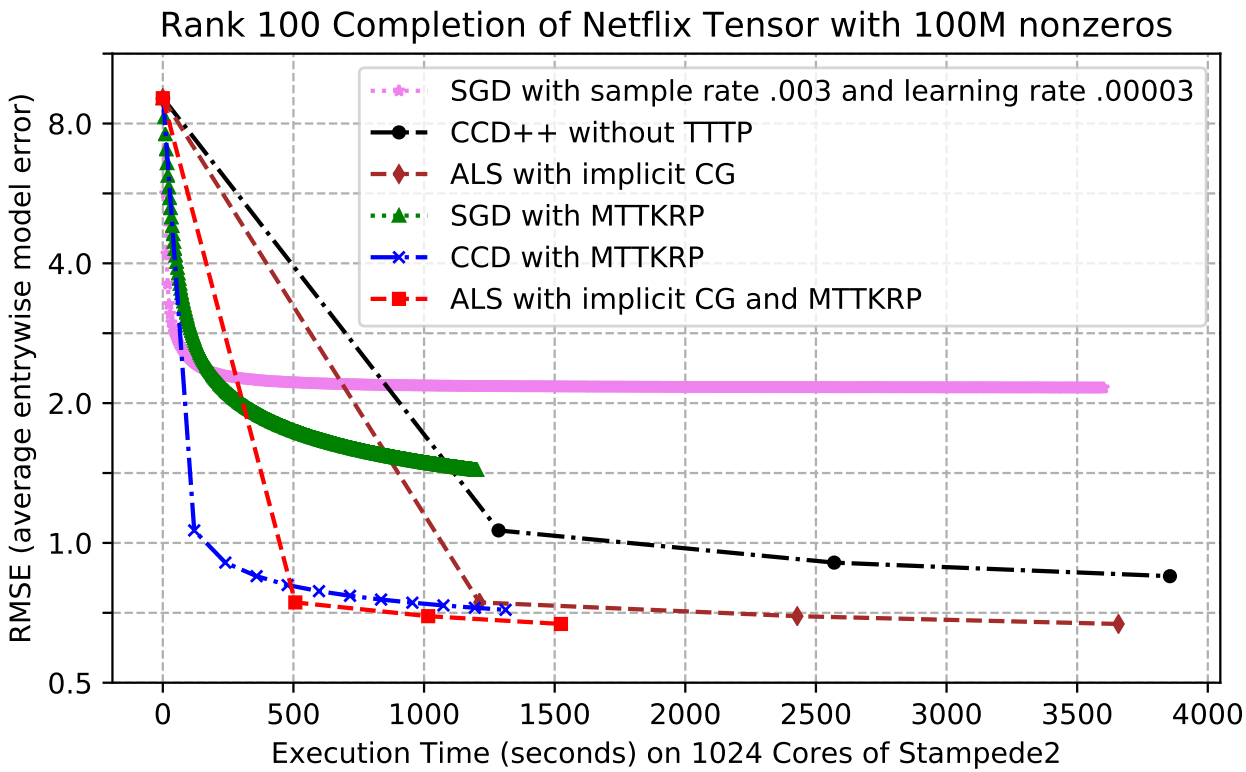
Parallel TTTP Performance





Execution time of the described TTTP kernel (all-at-once TTTP) and implementations based on pairwise tensor contraction, with $R = 1$ and $R = 60$ tensor products.

Parallel Tensor Completion Performance

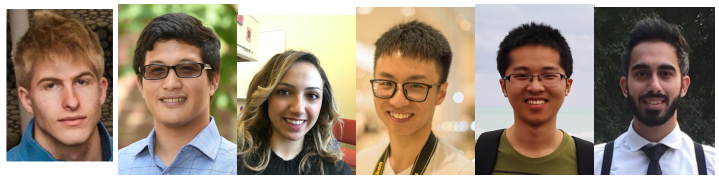


- CCD++ (optimizing for a column of a factor matrix at a time, avoiding solving linear systems)
- SGD can be done by selecting a random subsample of entries and running gradient descent
- (All-at-once) MTTKRP greatly accelerates performance

Conclusion

- Pairwise perturbation method for CP-ALS
  - Linjian Ma and E.S. arXiv:1811.10573
- Study of Gauss-Newton method for CP
  - Navjot Singh, Linjian Ma, Hongru Yang, and E.S. arXiv:1910.12331
- Tensor completion and Cyclops sparse kernels
  - Zecheng Zhang, Xiaoxiao Wu, Naijing Zhang, Siyuan Zhang, and E.S. arXiv:1910.02371

L P N A @CS@Illinois



DOE CSGF

NSF

http://lpna.cs.illinois.edu