CS 267
Lecture 17b: Parallel Machine Learning, Part 2

(Unsupervised Learning)

Aydin Buluc

https://sites.google.com/lbl.gov/cs267-spr2019/

Outline of the lecture

Tuesday: Part 1, Intro and Supervised Learning

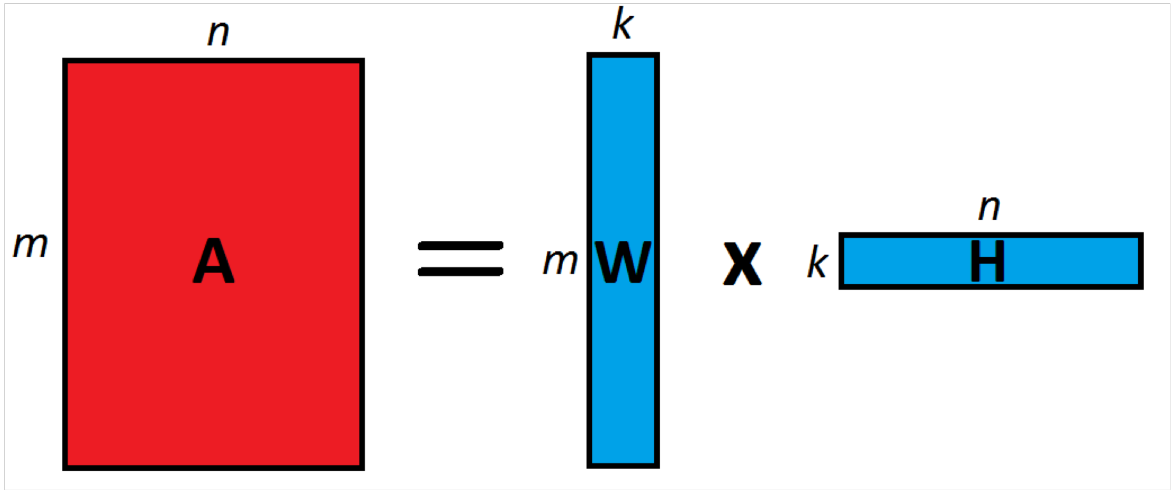
- Machine Learning & Parallelism Intro
- Neural Network Basics
- Scaling Deep Neural Network Training
- Support Vector Machines

Today: Part 2, Unsupervised Learning

- Non-Negative Matrix Factorization
- Spectral and Markov Clustering
- Sparse Inverse Covariance Matrix Estimation

Non-negative matrix factorization (NMF)

$$\min_{W \geq 0, H \geq 0} f(W,H) = \frac{1}{2} \|A - WH\|_F^2$$



- Dimensionality reduction with non-negativity constraints
- The name "factorization" is a misnomer; NMF is just a low-rank approximation as exact factorization is NP-hard
- NMF is a family of methods, not just one algorithm

The Alternating Updates Framework

Initialize H

Repeat until convergence:

1. For fixed H, solve $\min_{W \geq 0,} \left\| H^T W^T - A^T \right\|_F^2$
2. For fixed W, solve $\min_{H \geq 0,} \left\| WH - A \right\|_F^2$

Lots of algorithms fall into this framework.

- Multiplicative update (MU)
- Alternating least squares (ALS)
- Alternating non-negative least squares (ANLS)

J. Kim and H. Park. "Fast nonnegative matrix factorization: An active-set-like method and comparisons." SIAM Journal on Scientific Computing, 2011

Caveat emptor: This is not the only method for finding an NMF

Gemulla, Rainer, et al. "Large-scale matrix factorization with distributed stochastic gradient descent." KDD, 2011

The Alternating Updates Framework

Main computation is large-scale matrix multiplications:

1. $\mathbf{HH}^T$ and $\mathbf{AH}^T$ for updating **W**, given a fixed **H**
2. $\mathbf{W}^T\mathbf{W}$ and $\mathbf{W}^T\mathbf{A}$ for updating **H**, given a fixed **W**

- In general **W** and **H** are dense, but short-fat or tall-skinny
- **A** is often sparse but can be dense depending on application
- For increased interpretability, **H** or **W** can also be sparse

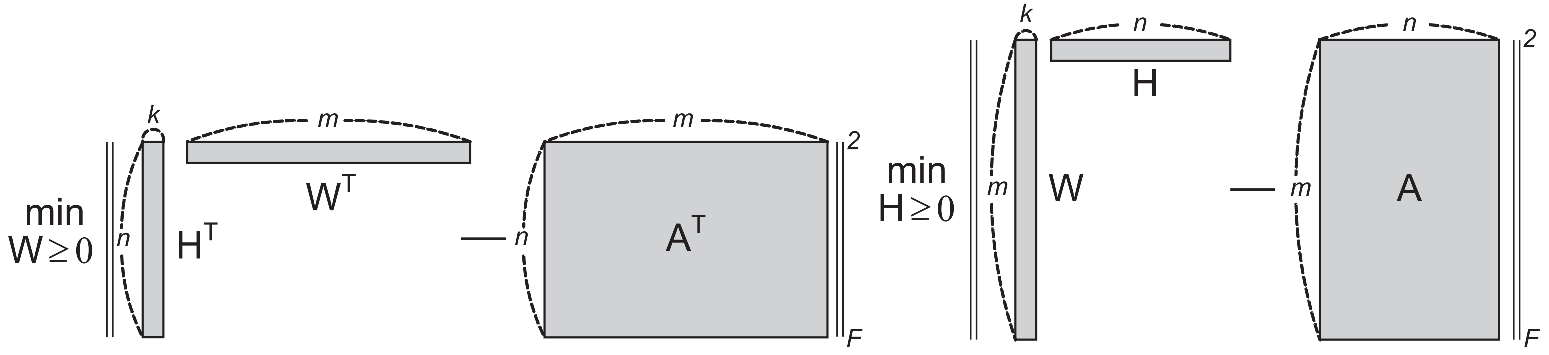


Figure: Kim and Park

The Alternating Updates Framework

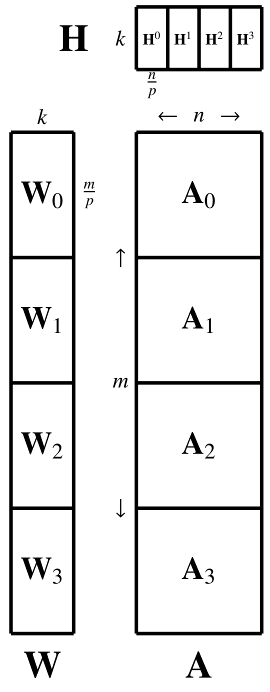
Main computation is large-scale matrix multiplications

Choose the best distribution and algorithm depending on:

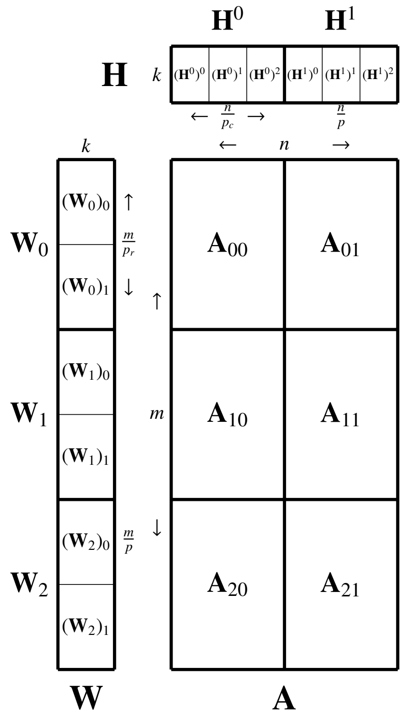
1- the relative sizes of the dimensions of the matrices

2- the number of processors

This work: Never communicate A, because it is asymptotically larger than H and W



(a) 1D Distribution with $p = p_r = 4$ and $p_c = 1$.



(b) 2D Distribution with $p_r = 3$ and $p_c = 2$.

Kannan, Ballard, Park. “MPI-FAUN: An MPI-Based Framework for Alternating-Updating Nonnegative Matrix Factorization”. 2016.

Outline of the lecture

Tuesday: Part 1, Intro and Supervised Learning

- Machine Learning & Parallelism Intro
- Neural Network Basics
- Scaling Deep Neural Network Training
- Support Vector Machines

Today: Part 2, Unsupervised Learning

- Non-Negative Matrix Factorization
- Spectral and Markov Clustering
- Sparse Inverse Covariance Matrix Estimation

Clustering

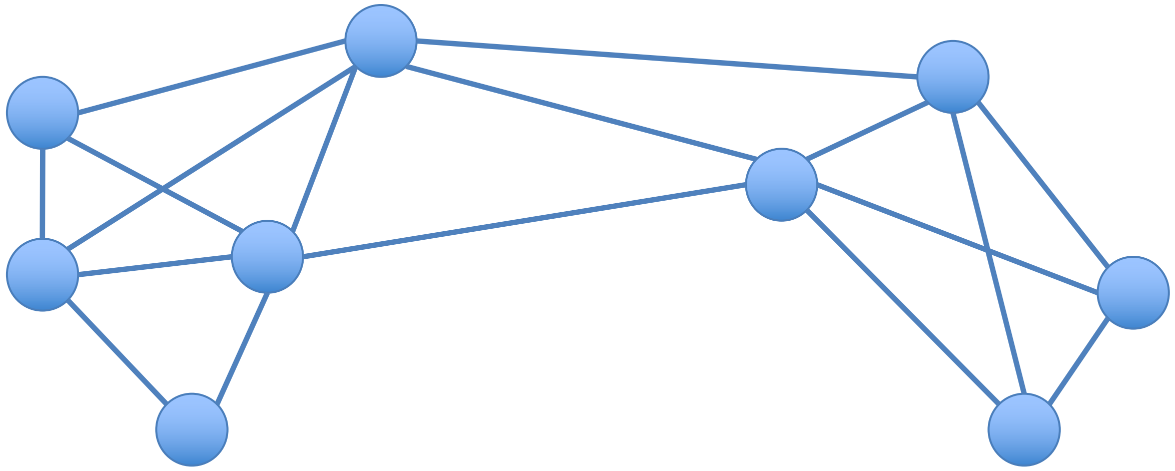
Many families of methods

- Centroid based (k-means, k-medians, and variations)
- Flow based (Markov clustering)
- Spectral methods
- Density based (DBSCAN, OPTICS)
- Agglomerative methods (single linkage clustering)
- …

- Often the right method depends on the input characteristics and require some domain knowledge.
- We will talk about parallel algorithms for two: **Spectral Clustering** and **Markov Clustering (MCL).**

Spectral Clustering

- **Input:** Similarities between data points
- Many ways to compute similarity, some are domain specific: cosine, Jaccard index, Pearson correlation, Spearman's rho, Bhattacharyya distance, LOD score, …
- We can represent the relationships between data points in a graph.
- Weight the edges by the similarity between points



Graph definitions

- ε-neighborhood graph
  - Identify a threshold value, ε, and include edges if the affinity between two points is greater than ε.
- k-nearest neighbors
  - Insert edges between a node and its k-nearest neighbors.
  - Each node will be connected to (at least) k nodes.
- Fully connected
  - Insert an edge between every pair of nodes.

Spectral Clustering Intuition

- The minimum cut of a graph identifies an optimal partitioning of the data.
- Spectral Clustering
  - Recursively partition the data set
    - Identify the minimum cut
    - Remove edges
    - Repeat until k clusters are identified
- **Problem**: Identifying a minimum cut is NP-hard.
- There are efficient approximations using linear algebra, based on the Laplacian Matrix, or **graph Laplacian**

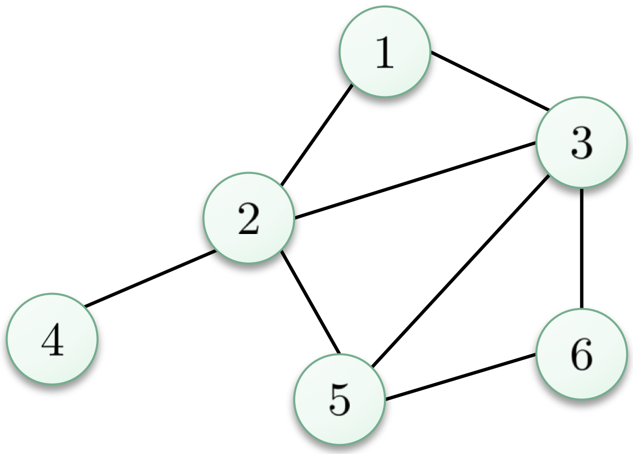
The Graph Laplacian

- Graph Laplacian
  - unnormalized graph Laplacian : $L = D - W$
  - normalized graph Laplacian

$$L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$$

$$L_{rw} = D^{-1} L = I - D^{-1} W \longleftarrow \text{related to random walk}$$

  - Example



$$L = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 & 0 \\ -1 & -1 & 4 & 0 & -1 & -1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & -1 & 0 & 3 & -1 \\ 0 & 0 & -1 & 0 & -1 & 2 \end{pmatrix}$$

Assume the weights of edges are 1.

One Spectral Clustering Algorithm

- I recommend the **normalized symmetric Laplacian**, as the numerical eigenvalue problem there is easier to solve.
- Normalized Spectral Clustering [Ng2002]

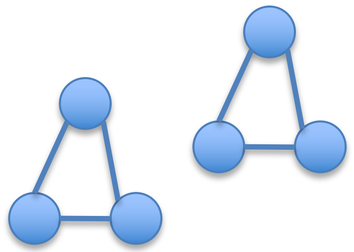
1. Construct a similarity graph and compute the normalized graph Laplacian L_{sym}.
2. Compute the k smallest eigenvectors $u_1, u_2, \cdots, u_k$ of L_{sym}.
3. Let $U = [\, u_1 \; u_2 \; \cdots \; u_k \,] \in \mathbb{R}^{n \times k}$.
4. Normalized the rows of U to norm 1.

$$U_{ij} \leftarrow \frac{U_{ij}}{(\sum_k U_{ik}^2)^{1/2}}$$

5. Let $y_i \in \mathbb{R}^k$ be the vector corresponding to the ith row of U.
6. Thinking of y_i's as points in $\mathbb{R}^k$, cluster them with k-means algorithms.

Why does it work? One intuitive explanation

- Ideal Case



$$L = D - W$$

2	-1	-1	0	0	0
-1	2	-1	0	0	0
-1	-1	2	0	0	0
0	0	0	2	-1	-1
0	0	0	-1	2	-1
0	0	0	-1	-1	2

$$Lv = \lambda v$$

1	**0**
1	0
1	0
0	1
0	1
0	1

- The multiplicity of the eigenvalue 0 gives the number of clusters (in this ideal case: the number of connected components).
- The real case is assumed to be an approximation to this situation.

How to compute those smallest Eigenvectors?

- Implementation via the Lanczos Algorithm
  - Workhorse is sparse-matrix-vector (SpMV) multiply
  - SpMV has no/minimal data reuse, bound by communication
  - To optimize sparse-matrix-vector multiply and minimize its communication, we graph partition (next lecture)

- Alternative algorithms are possible
  - Power iteration is cheaper but numerically unstable
  - LOBPCG (Locally-Optimized Block Preconditioned Conjugate Gradient) uses sparse-matrix times multiple vectors, thus has more favorable performance profile due to possible data reuse.

- In the end, you probably just want to call something existing.
  - ARPACK implements reverse communication eigensolvers: You implement the SpMV, its implements the numerical outer logic
  - PARPACK is its parallel version, The following code uses it: https://github.com/openbigdatagroup/pspectralclustering

Outline of the lecture

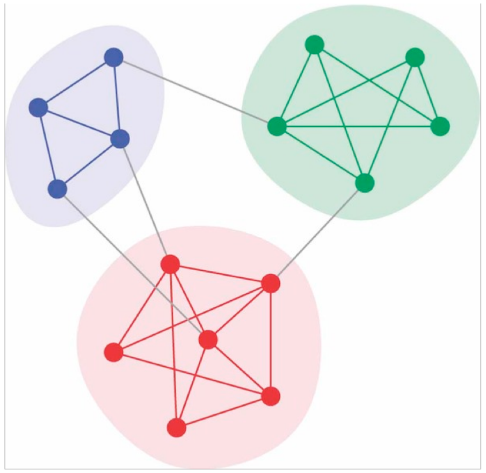
Tuesday: Part 1, Intro and Supervised Learning

- Machine Learning & Parallelism Intro
- Neural Network Basics
- Scaling Deep Neural Network Training
- Support Vector Machines

Today: Part 2, Unsupervised Learning

- Non-Negative Matrix Factorization
- Spectral and Markov Clustering
- Sparse Inverse Covariance Matrix Estimation

Philosophy of the Markov Cluster Algorithm (MCL)



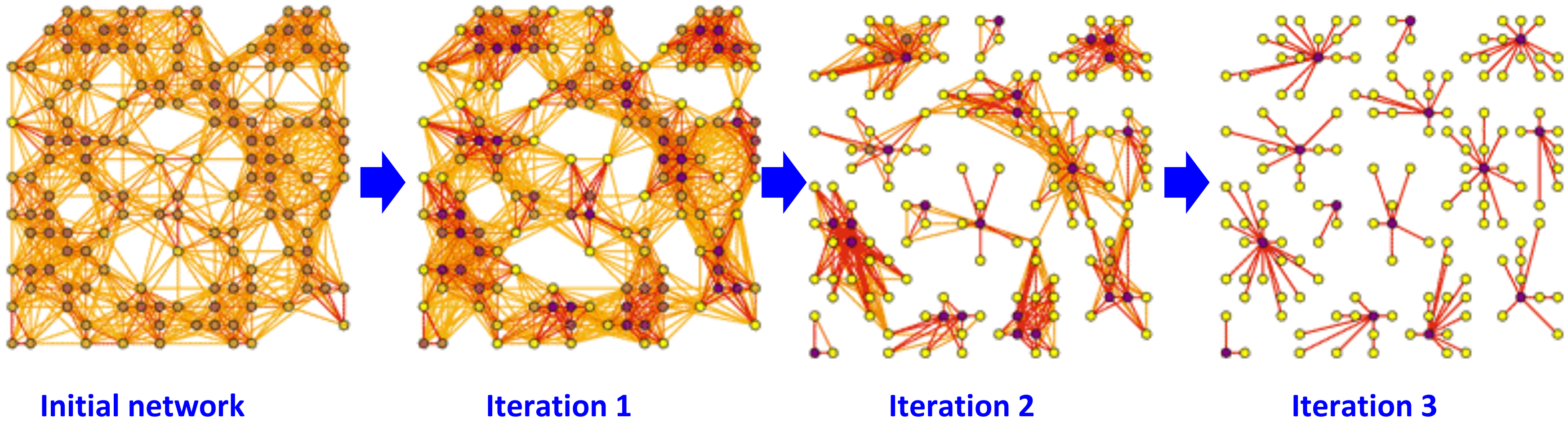
The number of **edges or higher-length paths** between two arbitrary nodes in a cluster is greater than the number of paths between nodes from different clusters

Random walks on the graph will frequently remains within a cluster

The algorithm **computes the probability** of random walks through the graph and **removes lower probability terms** to form clusters

The MCL Algorithm

Input: Adjacency matrix A (sparse & column stochastic)



At each iteration:

Step 1 (Expansion): Squaring the matrix

[corresponds to computing random walks of higher length]

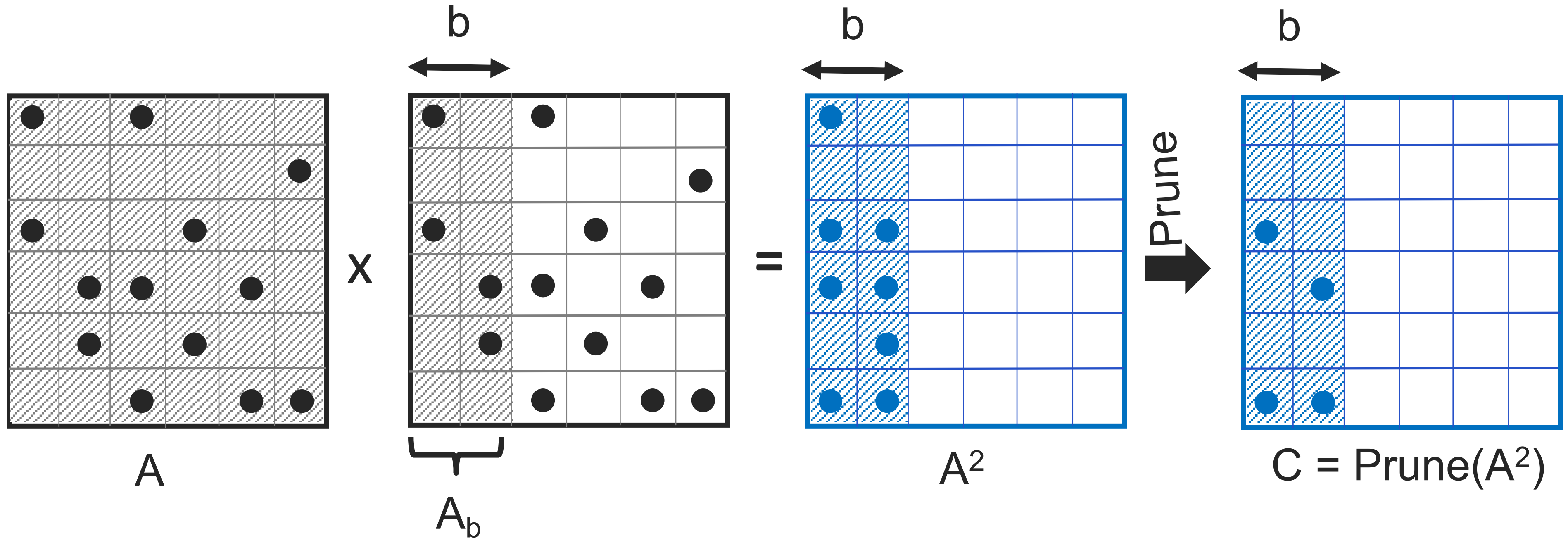
Step 2 (Inflation) : Hadamard power of a matrix (taking powers entrywise)

[boost the probabilities of intra-cluster walks and demote inter-cluster walks]

The expansion step of the MCL algorithm

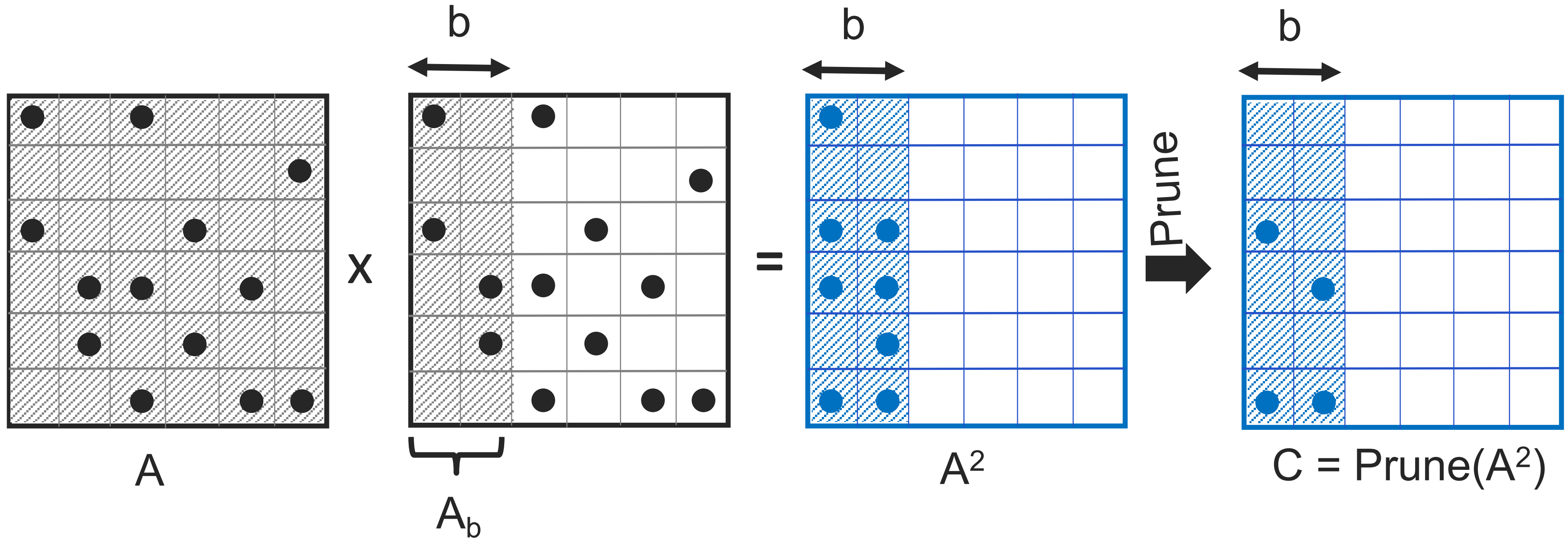
- Goal: Compute random walks of higher length
- Input: A column stochastic matrix (A)
- Algorithm
  1. Sparse matrix-sparse matrix multiplication (**SpGEMM**): A^2
  2. **Sparsify** A^2 by removing low probability terms
     - **Prune** entries in A^2 that are smaller than a threshold
     - **Recover** (if overdone pruning): Keep at least R entries (column-wise top-K selection)
     - **Selection** (if underdone pruning): Sparsify denser columns by keeping at most S entries (column-wise top-K selection)
- After sparsification at most max(R,S) (default to 1400) entries remains in each column of A^2

A combined expansion and pruning step



- b: number of columns in the output constructed at once
  - Smaller b: less parallelism, memory efficient (b=1 is equivalent to sparse matrix-sparse vector multiplication used in MCL)
  - Larger b: more parallelism, memory intensive

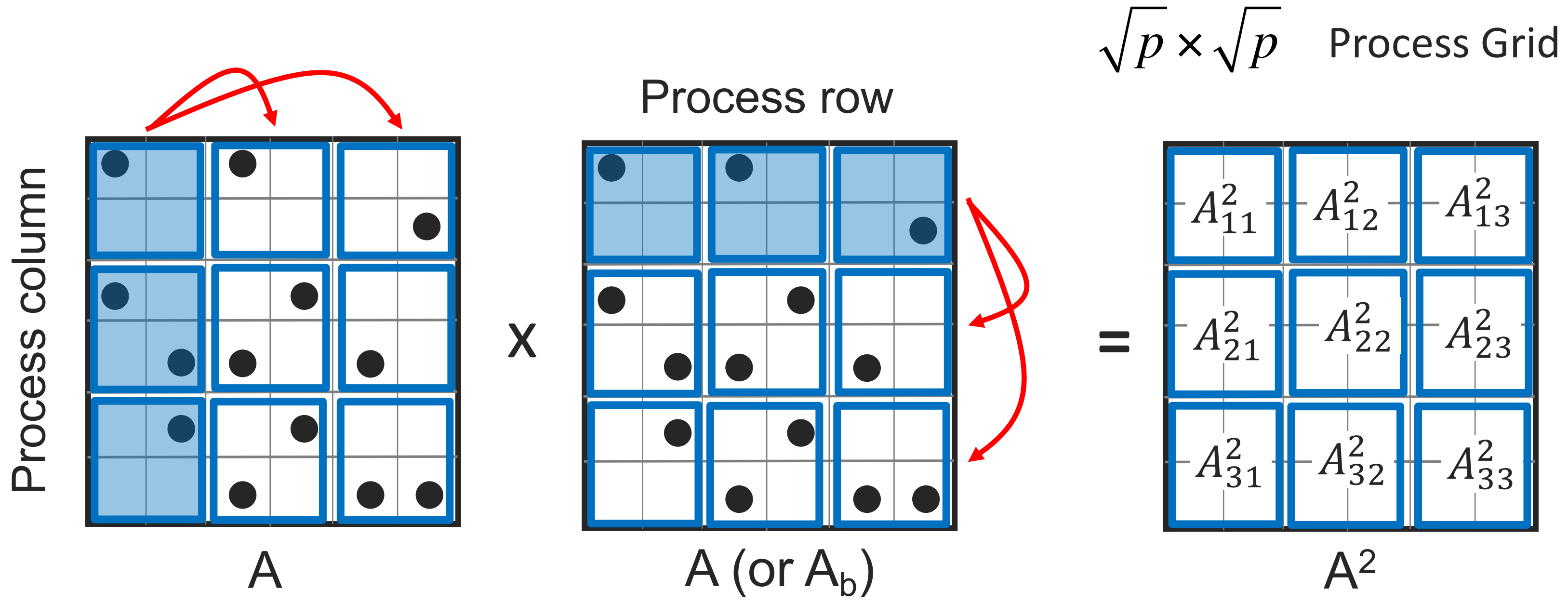
A combined expansion and pruning step



- b: number of columns in the output constructed at once
  - HipMCL selects b dynamically as permitted by the available memory
  - The algorithm works in h=N/b phases where N is the number of columns (vertices in the network) in the matrix

Current sparse matrix-matrix multiply algorithm in HipMCL

- ❑ Sparse SUMMA algorithm.
- ❑ Do this for each phase.
- ❑ Issue: repeated broadcast of A.
- ❑ Better ideas are brewing.

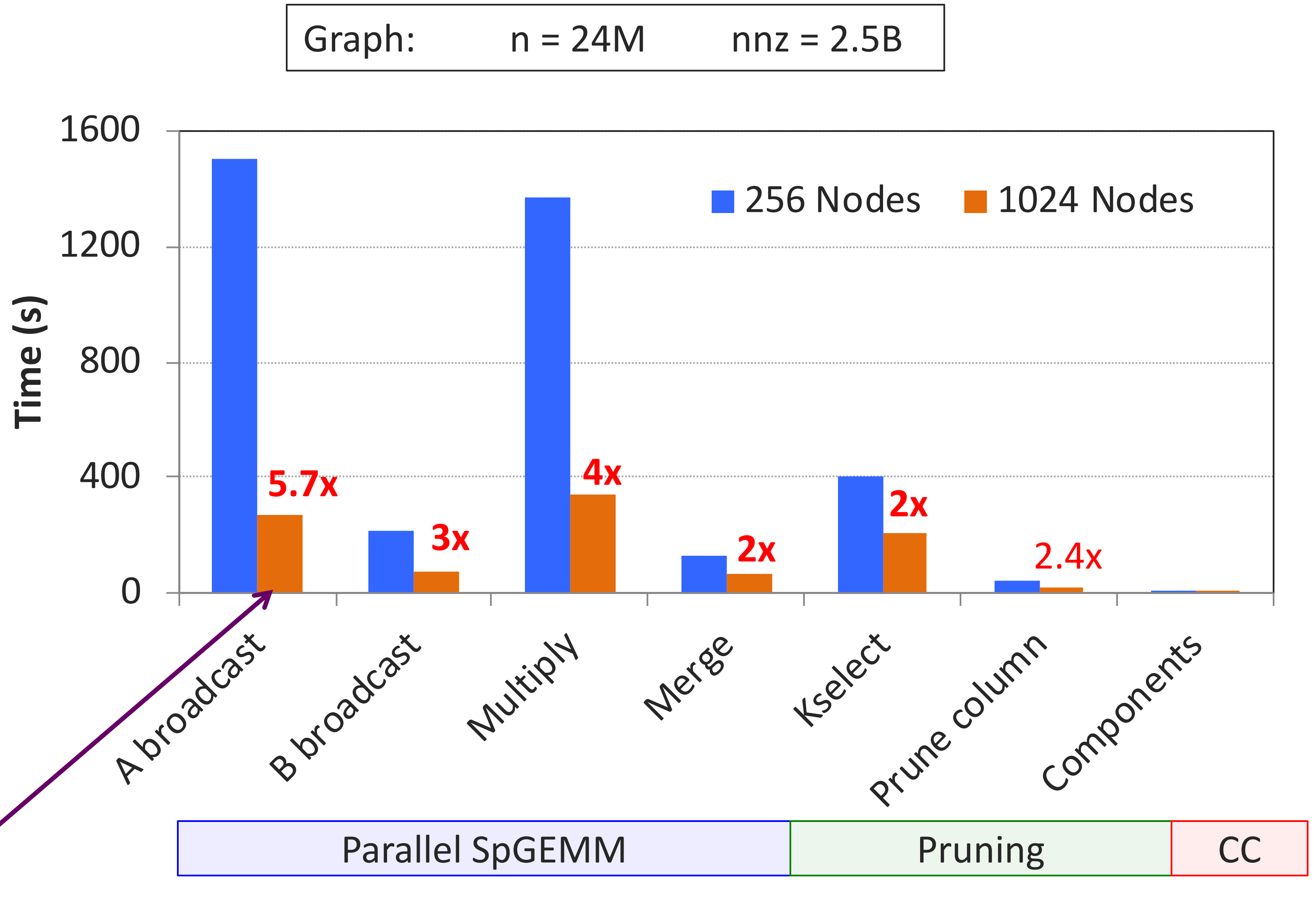


Other algorithmic steps of HipMCL

- There is more than sparse matrix multiply here.
  - Parallel k-selection algorithm for each column of the matrix (for Recovery and Selection). We will cover some algorithms in the **sorting and searching** lecture
  - Parallel pruning algorithm
  - Parallel connected component algorithm (to identify clusters after MCL is converged). Very fundamental **graph algorithm**, though we won't cover it this year.
  - Parallel file I/O. Reading terabytes of text efficiently is a challenge.

Azad, A., Pavlopoulos, G.A., Ouzounis, C.A., Kyrpides, N.C. and **Buluç, A.,** 2018. HipMCL: a high-performance parallel implementation of the Markov clustering algorithm for large-scale networks. Nucleic acids research.

Scaling of HipMCL



On 1024 nodes, we need fewer phases because of more aggregated memory

Outline of the lecture

Tuesday: Part 1, Intro and Supervised Learning

- Machine Learning & Parallelism Intro
- Neural Network Basics
- Scaling Deep Neural Network Training
- Support Vector Machines

Today: Part 2, Unsupervised Learning

- Non-Negative Matrix Factorization
- Spectral and Markov Clustering
- Sparse Inverse Covariance Matrix Estimation

Sparse Inverse Covariance Matrix Estimation

- **Precision matrix = Inverse covariance matrix**
- **Goal**: Estimating graphical model structure
- "The zeros of a precision matrix correspond to zero partial correlation, a necessary and sufficient condition for conditional independence (Lauritzen, 1996)"
- Sparsity often enforced by regularization
- One algorithm (HP-CONCORD)'s objective function:

$$\underset{\Omega \in \mathbf{R}^{p \times p}}{\text{minimize}} \quad -\log\det(\Omega_D^2) + \mathbf{tr}(\Omega S \Omega) + \lambda_1 \|\Omega_X\|_1 + \frac{\lambda_2}{2}\|\Omega\|_F^2,$$

- Ω is the sparse inverse covariance matrix we are trying to estimate

Why care? Finding Direct Associations

Partial Correlation (a.k.a. sparse inverse covariance estimation): direct association without confounders

- Gene Regulatory Network (GRN) estimation
- Joint modeling of SNPs and GRN
- Linkage Disequilibrium (LD) estimation
- Canonical Correlation Analysis (CCA)
- Genome-wide association studies (GWAS)

Data-driven hypothesis generation!

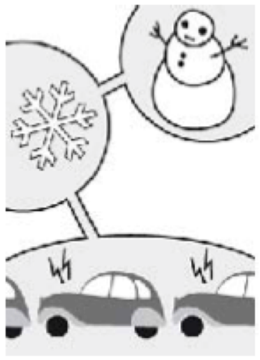


Fig. 1. Conditionally on the height of snow, the number of snowmen is independent of the intensity of traffic jams. This is represented by a two edges graph.

- ○ Computationally challenging;
- ○ **HP-CONCORD** on distributed memory increases scalability

HP-CONCORD Algorithm

Algorithm 2 The Cov variant of HP-CONCORD, for computing a sparse estimate of the inverse covariance matrix.

```
Input: data matrix $X \in \mathbf{R}^{n \times p}$; tuning parameters $\lambda_1, \lambda_2 \geq 0$; optimization tolerance $\epsilon > 0$
Output: estimate $\hat{\Omega} \in \mathbf{R}^{p \times p}$ of the underlying inverse covariance matrix $\Omega^0$
 1: $\Omega^{(0)} \leftarrow I$
 2: Compute $S \leftarrow \frac{1}{n} X^T X$          ▷ Compute (once) via a distributed dense-dense matrix multiplication
 3: Compute $W^{(0)} \leftarrow \Omega^{(0)} S$          ▷ Compute via a distributed sparse-dense matrix multiplication
 4: for $k = 0, 1, 2, \ldots$
 5:     Form $(W^{(k)})^T$          ▷ Form via a distributed matrix transpose
 6:     $G^{(k)} \leftarrow -(\Omega_D^{(k)})^{-1} + \frac{1}{2}((W^{(k)})^T + W^{(k)}) + \lambda_2 \Omega^{(k)}$          ▷ Use $W^{(k)}, (W^{(k)})^T$
 7:     $g(\Omega^{(k)}) \leftarrow -2 \sum_i \log(\Omega_{ii}^{(k)}) + \mathbf{tr}(W^{(k)} \Omega^{(k)}) + \frac{\lambda_2}{2} \|\Omega^{(k)}\|_F^2$          ▷ Use $(W^{(k)})^T$; see text for details
 8:     for $\tau = 1, \frac{1}{2}, \frac{1}{4}, \ldots$
 9:         $\Omega^{(k+1)} \leftarrow \mathcal{S}_{\tau\lambda_1}(\Omega^{(k)} - \tau G^{(k)})$          ▷ Apply the soft-thresholding operator, $\mathcal{S}_{\tau\lambda_1}$, in a distributed manner
10:         Compute $W^{(k+1)} \leftarrow \Omega^{(k+1)} S$          ▷ Compute via a distributed sparse-dense matrix multiplication
11:         $g(\Omega^{(k+1)}) \leftarrow -2 \sum_i \log(\Omega_{ii}^{(k+1)}) + \mathbf{tr}(W^{(k+1)} \Omega^{(k+1)}) + \frac{\lambda_2}{2} \|\Omega^{(k+1)}\|_F^2$          ▷ See text for details
12:     until $g(\Omega^{(k+1)}) \leq g(\Omega^{(k)}) - \mathbf{tr}((\Omega^{(k)} - \Omega^{(k+1)})^T G^{(k)}) + \frac{1}{2\tau} \|\Omega^{(k)} - \Omega^{(k+1)}\|_F^2$          ▷ See text for details
13: until a stopping criterion is satisfied, using $\epsilon$
14: return the estimate $\hat{\Omega} \leftarrow \Omega^{(k)}$
```

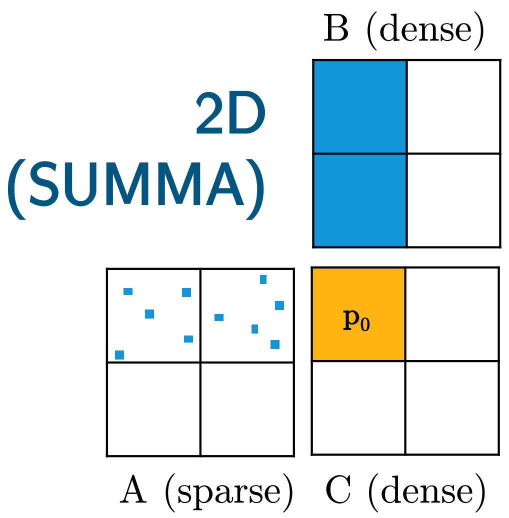
- Repeated use of sparse times dense matrix multiplication (SpDM^3)
- SpDM^3 is the bottleneck by a large margin.

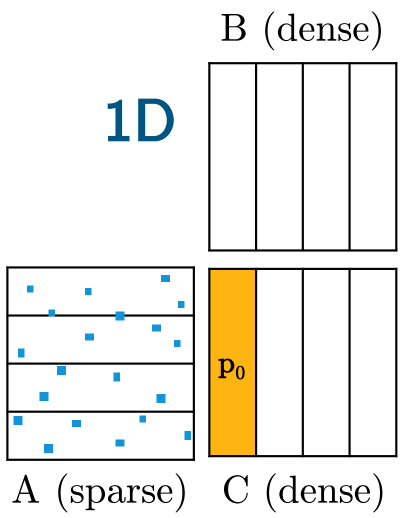
Koanantakool et al. Communication-avoiding optimization methods for distributed massive-scale sparse inverse covariance estimation. In AISTATS, 2018.

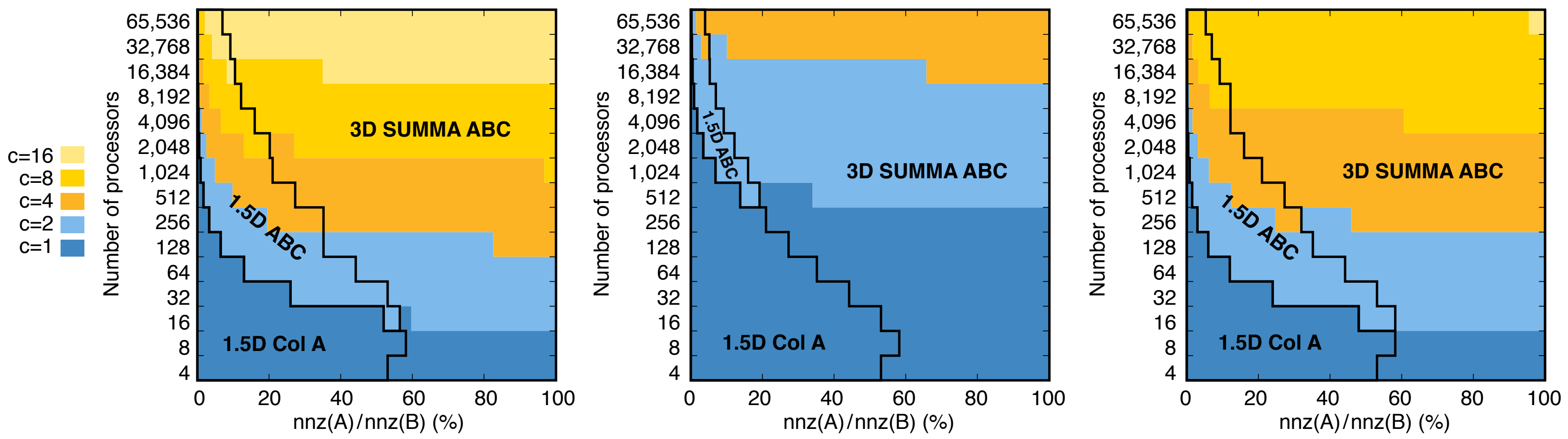
Sparse Matrix times Dense Matrix

Sometimes it pays off to communicate A instead.

How much of these ranges apply to real life NMF scenarios is open question (class project?)





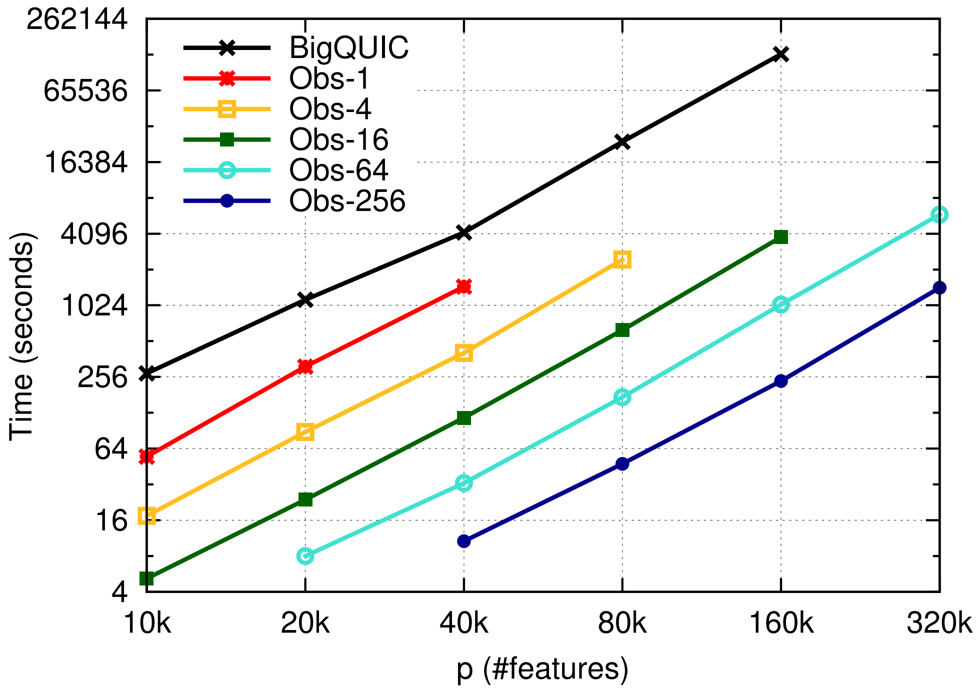


Koanantakool, Penporn, et al. "Communication-avoiding parallel sparse-dense matrix-matrix multiplication." IPDPS, 2016

HP-CONCORD Advantages

- HP-CONCORD makes fewer assumptions about the data (in particular, no Gaussianity is assumed) compared to competitors
- Thanks to **communication-avoiding matrix multiplication algorithms**, it reaches unprecedented scales

  - BigQUIC: previous state-of-the-art
  - Obs-K are the other variant of HP-CONCORD algorithm (K: number of nodes)
  - Experiment is trying to recover a random graph structure.


BigQUIC
Obs-1
Obs-4
Obs-16
Obs-64
Obs-256
Time (seconds)
10k
20k
40k
80k
160k
320k
p (#features)