A Parallel Algorithm for the Construction of Partial Information Decomposition-based Genome-scale Networks from Single-Cell Datasets

SC '26 @ Chicago, USA

Sriram P Chockalingam

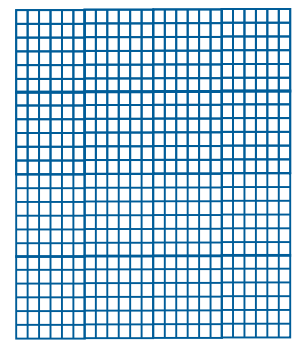
Georgia Institute of Technology

Joint work with Maneesha Aluru and Srinivas Aluru

Georgia Institute of Technology

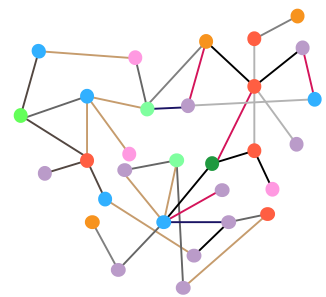
Construction of Gene Regulatory Networks (GRN)

- **Goal :** Learn gene interactions from gene expression data.
- **Gene Expression Data:** Snapshots of gene activity across different samples or conditions.
- Single-cell gene expression datasets:
  - Large datasets with millions of cells.
  - Sparse, has missing values and noisy.
  - Understand cell-specific interactions.



Gene Expression Data

Network Construction



Gene Network

Problem Definition and Key Contributions

- **Problem Definition:** Given $\mathsf{D} \in \mathbb{R}^{m \times n}$, m observations of n genes,
  - Construct $\mathsf{G} \in \mathbb{R}^{n \times n}$, adjacency matrix representing a network of n genes.
- **PIDC**, a gene network construction method tailored of single-cell data
  - Handles single-cell data characteristics via. partial-information decomposition.
  - **PIDC** takes $O(n^3)$ time
- Key Contributions
  - $O(n^2 \log n)$ sequential algorithm, a factor of $n/\log n$ improvement.
  - $O(n^2/p + \mathsf{T}_{\texttt{sort}}(n^2, p))$ distributed parallel algorithm on p processors.
  - Achieves 158X speedup over the existing PIDC implementation.
  -

Background : Random variables and MI

- Notations
  - $\mathcal{X}, |\mathcal{X}| = n$: Set of discrete random variables, each representing a gene.
  - Uppercase $X, Y, Z, \ldots$ for random variables
  - $P(X, Y)$ joint distribution of two genes represented by X and Y.
  - $p(x, y)$: Probability $P(X = x, Y = y)$
  - Lower case $x \in X$ also denote the distinct values, random variable X can take.
- Mutual Information:

$$I(X, Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right)$$

Background : Redundancy and Unique Contribution

- Specific Information: Given $X \in \mathcal{X}, y \in Y$

$$I_s(y; X) = \sum_{x \in X} \frac{p(x, y)}{p(y)} \log \left(\frac{p(x, y)}{p(x) \cdot p(y)} \right)$$

- Redundancy

$$\rho((X, Y); Z) = \sum_{z \in Z} p(z) \min(I_s(z; X), I_s(z; Y))$$

- Partial Information Decomposition
  - Unique Contribution, $\nu((X, Z); Y)) = I(X, Y) - \rho((X, Z); Y)$

Background : PUC

- Unique Contribution, $\nu((X,Z);Y)) = I(X,Y) - \rho((X,Z);Y)$
- PUC, $\gamma(X,Y)$, is formally defined as,

$$\begin{aligned}
\gamma(X,Y) &= \sum_{Z \in \mathcal{T}} \frac{\nu((X,Z);Y) + \nu((Y,Z);X)}{I(X,Y)} \\
&= \sum_{Z \in \mathcal{T}} \frac{2I(X,Y) - \rho((X,Z);Y) - \rho((Y,Z);X)}{I(X,Y)} \\
&= \frac{1}{I(X,Y)} \left(2(n-2)I(X,Y) - \sum_{Z \in \mathcal{T}} \rho((X,Z);Y) + \rho((Y,Z);X) \right)
\end{aligned}$$

- Bottleneck: $\sum_{Z \in \mathcal{T}} \rho((X,Z);Y) = \sum_Z \sum_y p(y) \min(I_s(y;X), I_s(y;Z))$

Algorithm : Key Insight

$$\sum_Z \sum_y p(y) \min(I_s(y;X), I_s(y;Z))$$

$n-2$

$\cdot$	$I_s(y;X)$
$y=1$	$\cdot$
$y=2$	$\cdot$
$y=3$	$\cdot$

$\cdot$	$I_s(y;Z)$
$y=1$	$\cdot$
$y=2$	$\cdot$
$y=3$	$\cdot$

$\cdots \quad \cdots$

$$\sum_y \sum_Z \min(p(y)I_s(y;X), p(y)I_s(y;Z))$$

$$\sum_y \sum_Z \min(\mu(y;X), \mu(y;Z))$$

$\cdot$	$\mu(y=1;\cdot)$
F	1.37
D	1.12
X	1.86
$\vdots$	$\vdots$
W	2.24
$\vdots$	$\vdots$

$\cdot$	$S_{y=1}$	$P_{y=1}$
D	1.12	0.00
F	1.37	1.12
X	1.86	2.49
W	2.24	4.35
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$

$$2.49 + 1.86 \times (n-5)$$

Sequential Algorithm

	Given: $P(X), \forall X \in \mathcal{X}; P(X,Y), \forall (X,Y)$	
1	Compute Mutual Information, $I(X,Y)$	$O(n^2)$
2	Compute tables $\mu(y;X), \forall (y,X)$	$O(n^2 d) = O(n^2)$
3	$\forall y \in Y, Y \in \mathcal{X}, \mathbf{S}_y \leftarrow$ Sorted $\mu(y;X)$ table	$O(n^2 \log n)$
3	$\forall y \in Y, Y \in \mathcal{X}, \mathbf{P}_y \leftarrow$ Prefix summed $\mathbf{S}_y$ table	$O(n^2)$
4	$\forall (X,Y)$, Compute (a) $\sum_Z \nu((X,Z);Y)$, (b)$\sum_Z \nu((Y,Z);X)$	$O(n^2)$
5	$\forall (X,Y)$, Compute PUC	$O(n^2)$

Parallel Algorithm : Data Structures

- Flat arrays, uniformly distributed across p processes
  - $\mathbf{S}$, concatenation of all the $\mathbf{S}_y$
  - $\mathbf{P}$, concatenation of all the $\mathbf{P}_y$
  - $\mathbf{Q}$, an array of size $|\mathbf{S}|$
    - $\mathbf{Q}[i]$ holds the value y corresponding to the $\mu(y; X)$ value in $\mathbf{S}[i]$.
  - $\mathbf{R}$, an array of size $|\mathbf{S}|$
    - $\mathbf{R}[i]$ holds the variable X corresponding to the $\mu(y; X)$ value in $\mathbf{S}[i]$.
- Each array size of $n \sum_{Y \in \mathcal{X}} |Y| = O(n^2)$

Parallel Algorithm

	Given $\mathbf{S}, \mathbf{P}, \mathbf{Q}, \mathbf{R}$	
1	Let $j \leftarrow$ processor's rank	
2	Compute $\langle X, Y, \sum \nu((X, Z); Y) \rangle$ for each entry $\mathbf{S}^{(j)}, \mathbf{P}^{(j)}, \mathbf{Q}^{(j)}, \mathbf{R}^{(j)}$	$O(n^2/p)$
3	Insert $\langle X, Y, \sum \nu((X, Z); Y) \rangle$ to $\mathbf{R}^{(j)}$	$O(n^2/p)$
4	Insert $\langle X, Y, I(X, Y) \rangle$ to $\mathbf{R}^{(j)}$	$O(n^2/p)$
5	Parallel Sort the distributed $\mathbf{R}$	$\mathsf{T}_{\texttt{sort}}(n^2, p))$
6	Compute PUC from sorted $\mathbf{R}^{(j)}$	$O(n^2/p)$

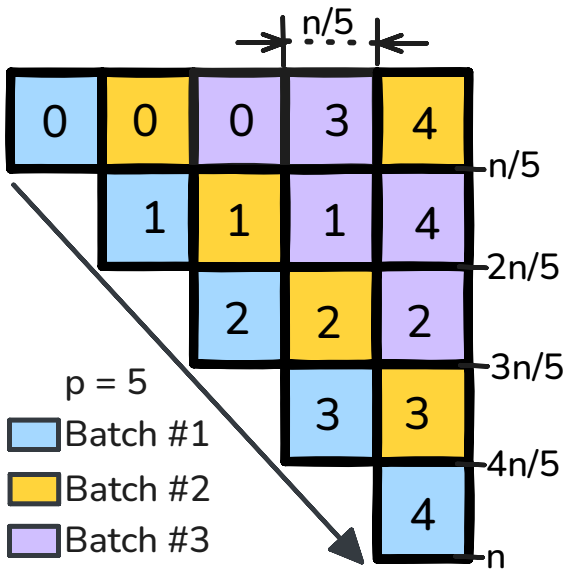
Implementation : Platform and Data Processing

- PIDC implementation in julia
  - Read input data from `HDF5` input instead of `csv` files
- Our MPI Implementation
  - Rust programming language
  - `rsmpi` : MPI 3.1 wrapper (Experiments use `OpenMPI`)
  - `sope` : Convenience library features include generic struct types, sorting etc
  - `hdf5-rs` : Updated for parallel I/O
- `anndata` I/O
  - Save D^T with genes as rows for faster parallel I/O

Implementation : Computing $P(X)$ and $P(X, Y)$

- Marginal distributions, $P(X)$
  - Block distribute n genes across p processes
  - Compute histogram with BayesianBlocks method for n/p genes
- Joint Distribution
  - Copy of $P(X), \forall X$ in every process (Only 26MB for 30K genes).
  - Each process computes $P(X, Y) \approx \binom{n}{2}$ genes.
  - Sorted order of (X, Y) enable constant time ownership id.

Implementation : Handling large datasets



- Datasets with m exceeding millions of cells.
- Batch-wise processing of pair blocks with only $O(mn/p)$ data per process.
- Row/column-wise distribution
  - Parallel I/O of non-contiguous overlapping rows of D^T.
- Diagonal distribution
  - Uniform block-distributed parallel read of the input HDF5

Implementation : Parallel Algorithm

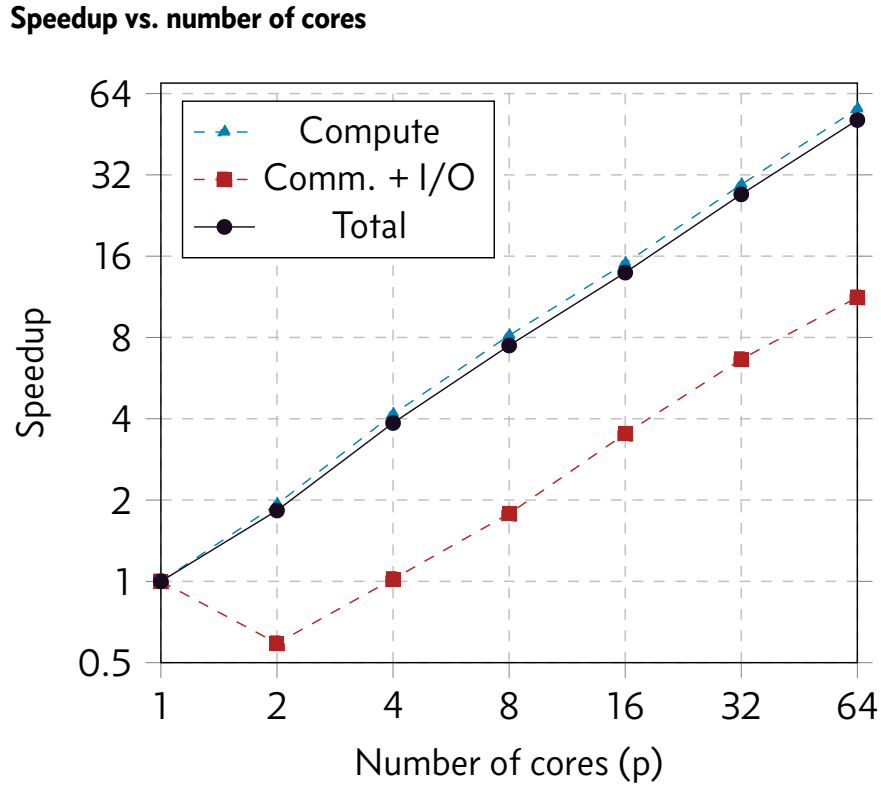
- Compute MI and $\mu(\cdot, \cdot)$
  - $I(X, Y)$
  - $\mu(x; Y) \forall x \in Y$, and $\mu(y; X), \forall y \in Y$.
  - Distribute $\mu(x; Y)$ values in the order of Y , followed by y, and then by X.
- Compute PUC, $\gamma(X, Y)$
  - $\mathbf{Q}$ and $\mathbf{R}$ are not explicitly computed
    - Derived from the ordering of X and $x \in X$
  - Radix sorting with a single `MPI_all2allv`

Experiments: Datasets

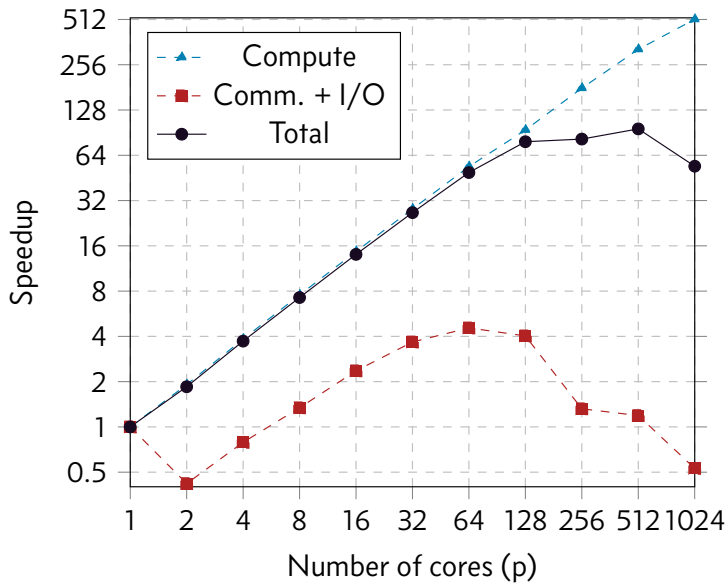
Dataset	Tissue	m	n	D (GB)	$\sum \lvert P(X,Y) \rvert$ (GB)
D1	PBMC	23,766	5,000	0.44	47.38
D2	Lung	91,068	5,000	1.82	121.10
	"		10,000	3.39	476.46
	"		15,000	5.08	1,131.42
D3	PBMC	794,170	20,000	57.63	8,828.66
	"		30,000	86.45	19,570.00

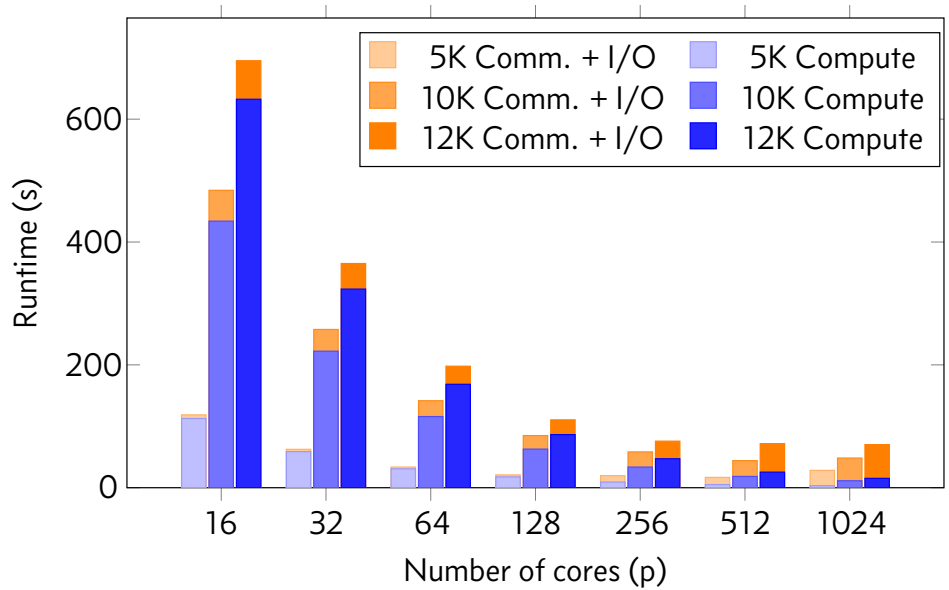
Results for dataset $D1$

- Machine: Shared memory machine with 72-core Intel® Xeon® E7-8870 2.7 GHz CPU
- Max. speedup of 52 on 64 cores
- Communication and I/O times are a small fraction (at most 11% with 64 cores)
- 158X times faster than PIDC, reducing the runtime from 10.8 hours to about 4 minutes. to about 4 minutes.



Results : Dataset $D2$ w. 5K genes

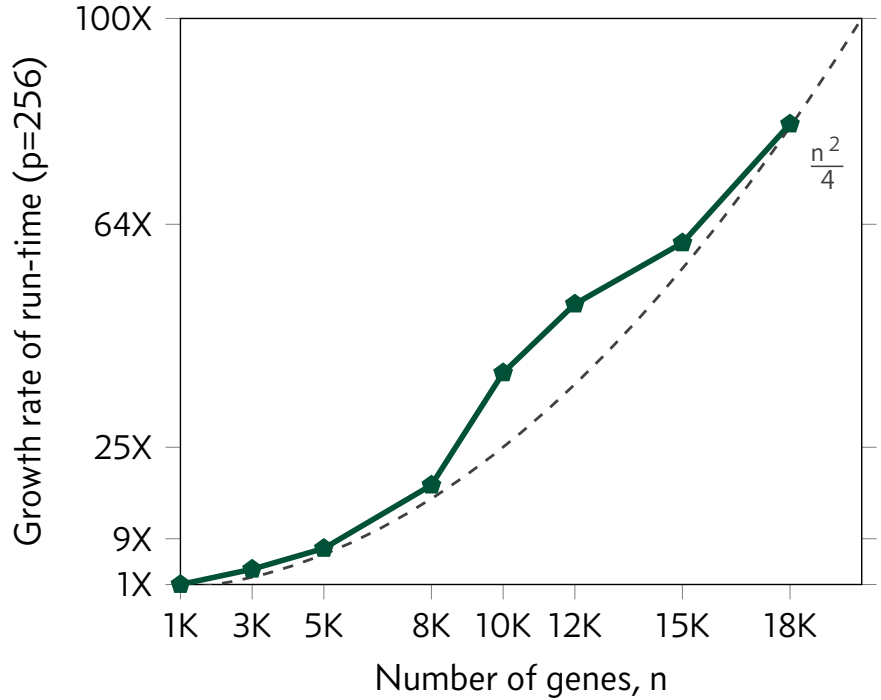




- 484-node cluster where each node has a 24-core Intel® Xeon® 6226 2.7 GHz processor
- Max speedup of speedup of 96 with 512 cores, which then drops to 55 with 1, 024 cores

Results : Runtime vs. number of gene Weak Scaling

- D2 using 256 cores, varying network size from 1, 000 genes to 18, 000 genes.
- Observed growth is close to $O(n^2)$ rather than $O(n^2 \log n)$ because parallel sort does not dominate the run-time .



Results : Big Datasets

n	20K	20K	30K	30K
p	1, 024	2, 048	1, 024	2, 048
Distributions, μ (min.)	116.89	71.08	243.39	149.36
μ →PUC (min.)	0.91	1.02	1.63	1.90
Total Runtime (min.)	117.80	72.10	245.02	151.26

- 20K-gene network took 1.9 hours and 1.2 hours, respectively, with 1, 024 cores and 2, 048 cores.
- 30K-gene network took 4.1 hours and 2.5 hours, respectively, with 1, 024 cores and 2, 048 cores.
- No linear scaling from 1, 024 to 2, 048 cores due to the increase in I/O time.

Conclusions

- Key Contributions
  - $O(n^2 \log n)$ sequential algorithm, a factor of $n/\log n$ improvement.
  - $O(n^2/p + \mathsf{T}_{\texttt{sort}}(n^2, p))$ distributed parallel algorithm on p processors.
  - To our knowledge, the 30K network is the largest network covering the complete human genome.
- Future work
  - Improve scalability of computing the distributions by better process assignment.
  - Address the cases where the gene expression values need to be treated as continuous.
  - I/O scalability via asynchronous read/writes.