A Parallel Framework for Constraint-Based Bayesian Network Learning via Markov Blanket Discovery

Ankit Srivastava*

Sriram Chockalingam

Srinivas Aluru

Accepted to be presented at SC20 – November 17th,10-10:30am

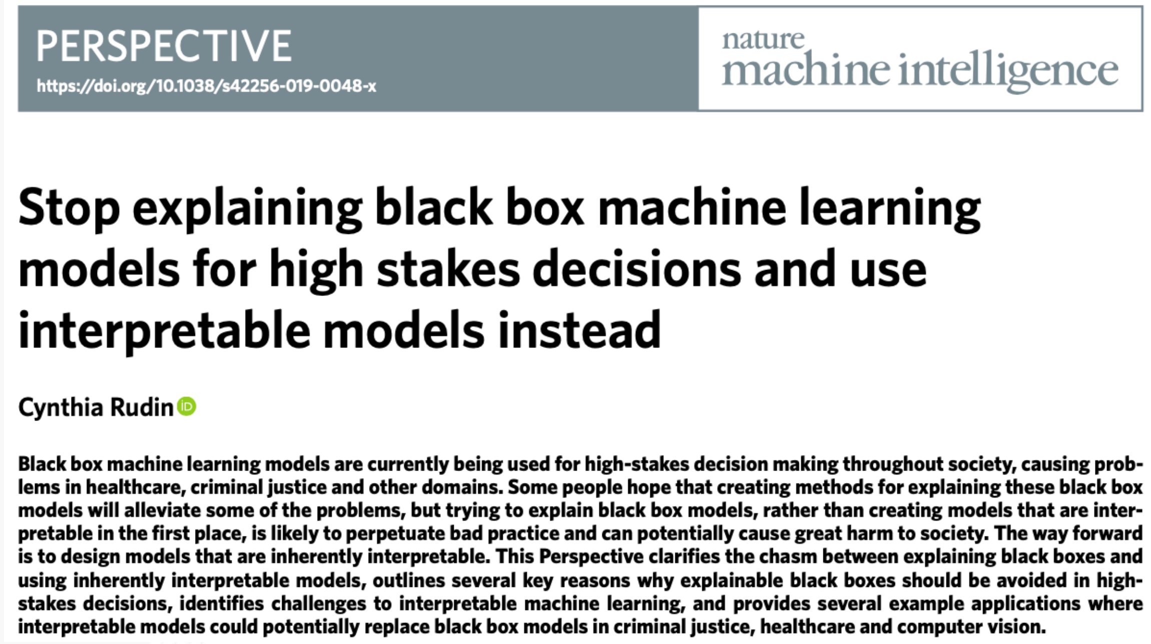
Motivation

- Machine Learning (ML) models are used for decision-making in a diverse set of fields, e.g., spam filtering, dynamic pricing, etc.
  - "Black box" models are typically used for the purpose

Image Source: *Interpretable Machine Learning — Fairness, Accountability, and Transparency in ML systems*
https://medium.com/@ODSC/interpretable-machine-learning-fairness-accountability-and-transparency-in-ml-systems-3ab45d961fbc

Motivation

- Machine Learning (ML) models are used for decision-making in a diverse set of fields, e.g., spam filtering, dynamic pricing, etc.
  - “Black box” models are typically used for the purpose
- Increasingly, ML is being used in high human-impact areas, e.g., healthcare, criminal justice, etc.
  - Explainable ML is the need of the hour



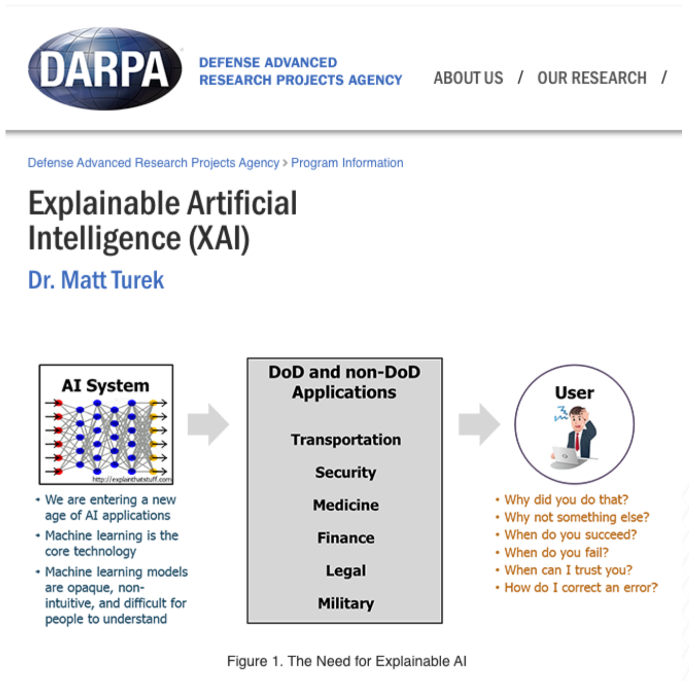
PERSPECTIVE
https://doi.org/10.1038/s42256-019-0048-x

nature machine intelligence

Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead

Cynthia Rudin

Black box machine learning models are currently being used for high-stakes decision making throughout society, causing problems in healthcare, criminal justice and other domains. Some people hope that creating methods for explaining these black box models will alleviate some of the problems, but trying to explain black box models, rather than creating models that are interpretable in the first place, is likely to perpetuate bad practice and can potentially cause great harm to society. The way forward is to design models that are inherently interpretable. This Perspective clarifies the chasm between explaining black boxes and using inherently interpretable models, outlines several key reasons why explainable black boxes should be avoided in high-stakes decisions, identifies challenges to interpretable machine learning, and provides several example applications where interpretable models could potentially replace black box models in criminal justice, healthcare and computer vision.



DARPA DEFENSE ADVANCED RESEARCH PROJECTS AGENCY ABOUT US / OUR RESEARCH /

Defense Advanced Research Projects Agency › Program Information

Explainable Artificial Intelligence (XAI)

Dr. Matt Turek

- We are entering a new age of AI applications
- Machine learning is the core technology
- Machine learning models are opaque, non-intuitive, and difficult for people to understand

- Why did you do that?
- Why not something else?
- When do you succeed?
- When do you fail?
- When can I trust you?
- How do I correct an error?

Figure 1. The Need for Explainable AI

Georgia Tech
CREATING THE NEXT

Motivation

- **Bayesian networks** (**BNs**) enable probabilistic reasoning about links between the variables of interest – interpretable decisions
  - Used for medical diagnosis, legal reasoning, epidemiology, etc.

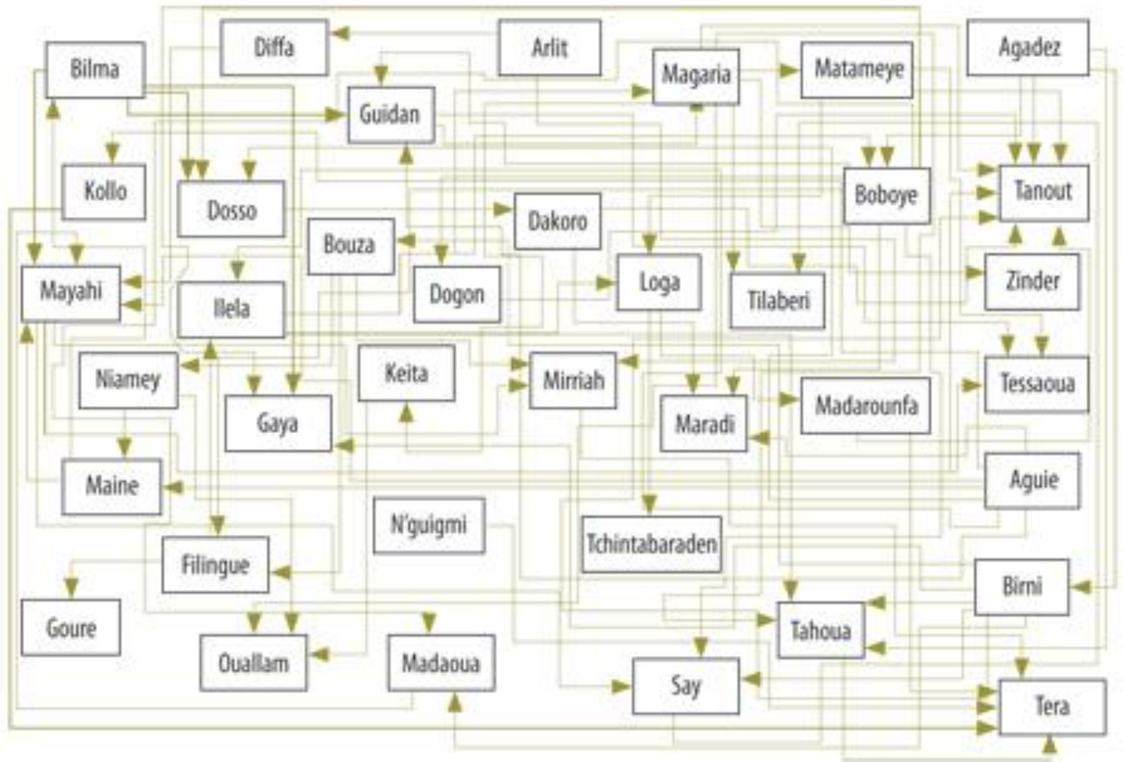


Image Source: Beresniak, A., et al. "A Bayesian network approach to the study of historical epidemiological databases: modelling meningitis outbreaks in the Niger." *Bulletin of the World Health Organization* 90 (2012): 412-417a.b45d961fbc

Motivation

- **Bayesian networks** (**BNs**) enable probabilistic reasoning about links between the variables of interest – interpretable decisions
  - Used for medical diagnosis, legal reasoning, epidemiology, etc.
- Learning structure of BNs is compute-intensive – needs parallelism
- Existing libraries for learning BNs support limited or no parallelism
  - e.g., *bnlearn*, *pcalg*, *Tetrad*
- Different parallelization strategies have been proposed for various learning algorithms – difficult to integrate different strategies
  - Single parallel library with support for multiple algorithms is desirable

Related Works

- Exactly learning the structure of BNs is NP-hard
  - Efficient parallel solutions can only learn very small networks (<50 variables)

- Heuristic methods are used for learning bigger networks
  - *Score-based* methods rely on a scoring function to choose the structure
  - *Constraint-based* methods rely on conditional independence (CI) tests

- Multiple parallelization approaches have been proposed for *score-based* methods
  - Misra et al. (2014) developed an approach that can construct a 15,216 variable BN in less than 172 seconds using 1.57 million cores of Tianhe-2

Related Works

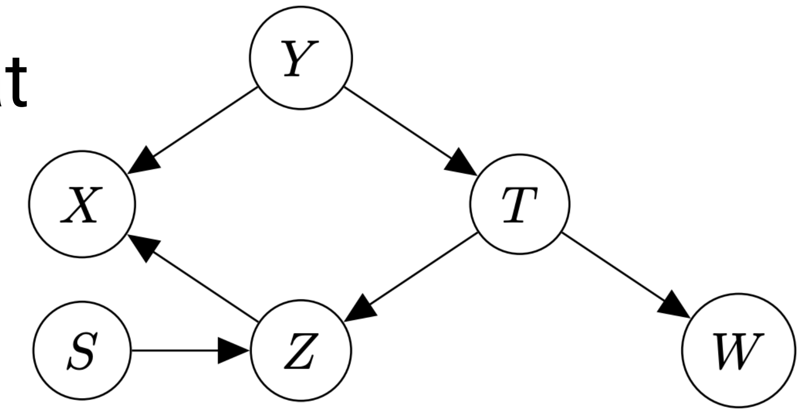
- *Constraint-based* methods have received comparatively little attention
  - Most studies in the space have focused on the *stable-PC* algorithm

- Nikolova et al. (2011) parallelized the *MMHC* (Tsamardinos et al., 2006) and the *PCMB* (Pena et al., 2007) algorithms
  - Scales well up to 512 cores for learning neighborhoods of 1,000 variables
  - Scaling deteriorates as the number of variables are increased – work distribution strategy is suboptimal

Background

- BN is a graphical representation of a joint probability distribution of a set of variables ($\mathcal{X}$)
  - Decomposes into probabilities of variables conditioned on their parents

- PC set of a variable consists of the variables that are dependent on it, given any conditioning set
  - i.e., $X \in PC(T) \Leftrightarrow \neg I(X, T|\mathcal{S}) \forall \mathcal{S} \subseteq \mathcal{X} \setminus \{X, T\}$


$$\begin{tikzcd} & & Y \arrow[dll] \arrow[dr] & & \\ X & & & T \arrow[dl] \arrow[dr] & \\ S \arrow[r] & Z \arrow[ul] & & & W \end{tikzcd}$$

$P(\mathcal{X}) = P(S)P(T|Y)P(W|T)P(X|\{Y,Z\})P(Y)P(Z|\{S,T\})$

- Markov blanket (MB) of a variable consists of the variables that render the variable independent of other variables
  - i.e., $I(X, T|MB(T)) \forall X \in \mathcal{X} \setminus (\{T\} \cup MB(T))$
  - Assuming *faithfulness*, $MB(T) = PC(T) \cup (Parents(X) \forall X \in PC(T))$

Background

- **Constraint-based algorithms** learn BN by conducting repeated CI tests using given data set of m observations for the n variables
  - Statistical tests, e.g., G^2 test for discrete data

- **Local-to-global algorithms** learn PC or MB of the variables separately and then combine them to get the BN skeleton

- **Blanket learning algorithms** learn MB sets of the variables first
  - Grow-Shrink (***GS***) (Margaritis & Thrun, 2000)
  - Incremental Association MB (***IAMB***) (Tsamardinos et al., 2003)
  - Interleaved IAMB (***Inter-IAMB***) (Tsamardinos et al., 2003)

Blanket Learning Algorithms

- All the algorithms use variations of the *Grow-Shrink* scheme
  - *Grow* phase: Add a variable to the candidate MB set
  - *Shrink* phase: Remove false positive variables from the candidate MB set
- The algorithms differ in the specifics of how the scheme is iterated
  - Choosing variables to be added in *Grow* phase
    - *IAMB* & *Inter-IAMB* pick the "most dependent" variable given the current candidate MB set; *GS* picks the first dependent variable
  - Order of *Grow* & *Shrink* phases
    - *GS* & *IAMB* execute multiple iterations of *Grow* phase followed by one *Shrink* phase; *Inter-IAMB* interleaves executes *Grow* and *Shrink* phases in every iteration
- *Symmetry correction* is performed for MB sets ($X \in MB(T) \Leftrightarrow T \in MB(X)$)
- PC sets are learned from the MB sets (PC ⊆ MB)

Parallel Framework

- Key design considerations:
  - Variables have different MB set sizes – distributing variables is suboptimal
    - Consider variable pairs in parallel instead
  - Computations for CI tests account for more than 94% of sequential run-time
    - Conduct CI tests with similar conditioning set sizes in parallel

Parallel Framework

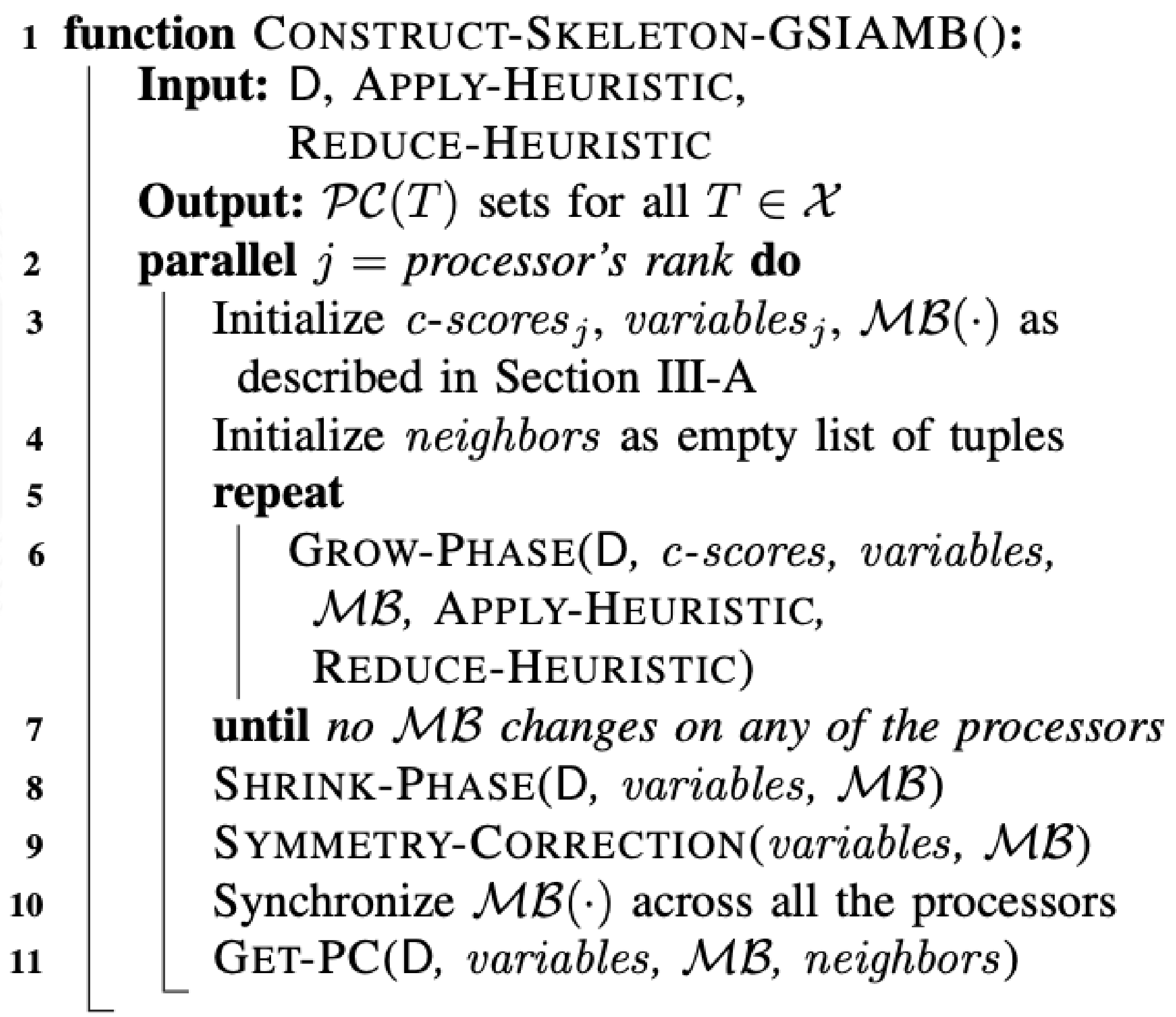
- Primary data structures:
  - $variables$ is the set of variables for which MB sets are to be computed
    - Typically initialized to $\mathcal{X}$ for learning BNs
  - $c\text{–}scores$ is a list of tuples $< X, Y, \theta_{XY} >$ such that $X \in variables, Y \in \mathcal{X} \setminus \{X\}$
    - Tuples with the same X are contiguously arranged in the list
    - θ_{XY} is the score of Y for addition to the MB set of X

- Distributed data structures in parallel:
  - $c\text{–}scores$ is block-distributed across processors – $c\text{–}scores_i$ on processor i
  - $variables_i$ is the set for which MB sets are computed on processor i

Parallel Framework

- Parallel *Grow* phase on processor i
    - Update θ_{XY} for all the tuples $\in c\text{–}scores_i$
    - Add Y to the MB of X corresponding to the best θ_{XY}
        - Best θ_{XY} is dependent on the algorithm
        - Can be identified using two segmented parallel prefix operations for all the variables
- Parallel *Shrink* phase on processor i
    - Complete MB sets are available for all elements of $variables_i$
    - *Shrink* can be performed locally on every processor
- Parallel *Symmetry Correction* using algorithm by Nikolova et al. (2011)
- Parallel PC from MB for $variables_i$ on processor i

Parallel Framework



```
function CONSTRUCT-SKELETON-GSIAMB():
    Input: D, APPLY-HEURISTIC,
           REDUCE-HEURISTIC
    Output: $\mathcal{PC}(T)$ sets for all $T \in \mathcal{X}$
    parallel $j$ = processor's rank do
        Initialize $c\text{-}scores_j$, $variables_j$, $\mathcal{MB}(\cdot)$ as
         described in Section III-A
        Initialize neighbors as empty list of tuples
        repeat
            GROW-PHASE(D, c-scores, variables,
             $\mathcal{MB}$, APPLY-HEURISTIC,
             REDUCE-HEURISTIC)
        until no $\mathcal{MB}$ changes on any of the processors
        SHRINK-PHASE(D, variables, $\mathcal{MB}$)
        SYMMETRY-CORRECTION(variables, $\mathcal{MB}$)
        Synchronize $\mathcal{MB}(\cdot)$ across all the processors
        GET-PC(D, variables, $\mathcal{MB}$, neighbors)
```

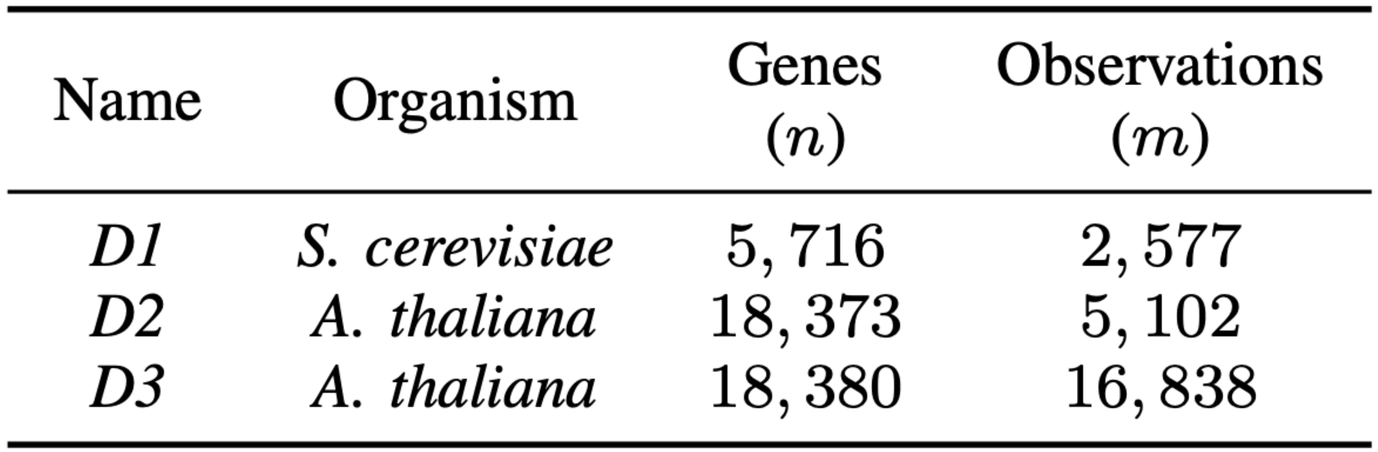
Implementation

- Implemented using *C++* and *MPI* (conforms to *C++14* and *MPI 3.1*)
  https://github.com/asrivast28/ramBLe

- Optimizations for fast execution in practice
  - Algorithm specific optimizations - *GS*
  - Experimented with different statistic computation strategies for CI tests
  - Dynamic load balancing scheme

Experiments & Results

- Experimental setup
  - 64 nodes of the *Hive* cluster, 16 MPI processes per node
  - *RHEL* 7.6, *gcc* v9.2.0, *MVAPICH2* v2.3.3
- Used real gene-expression data sets to learn gene networks



Name	Organism	Genes (n)	Observations (m)
D1	*S. cerevisiae*	$5,716$	$2,577$
D2	*A. thaliana*	$18,373$	$5,102$
D3	*A. thaliana*	$18,380$	$16,838$

- Used three simulated data sets ($S1$, $S2$, and $S3$) to show scalability
  - $n = 30{,}000$; $m = 10{,}000$; edge addition probabilities: $5e-5$, $1e-4$, and $5e-4$

Experiments & Results

- Sequential comparison with *bnlearn*
  - Popular library for learning BNs; *C* implementation interfaces with *R*

Algorithm	Data set	Run-time (s)		Speedup
		bnlearn	*Ours*	
GS	*D1*	8 720.0	240.1	36.3
	D2	×	6 760.3	N/A
	D3	×	18 695.0	N/A
IAMB	*D1*	975.9	624.6	1.6
	D2	40 605.7	14 529.8	2.8
	D3	84 403.1	46 603.2	1.8
Inter-IAMB	*D1*	992.0	624.1	1.6
	D2	40 819.0	14 559.0	2.8
	D3	89 839.7	48 442.4	1.9

Experiments & Results

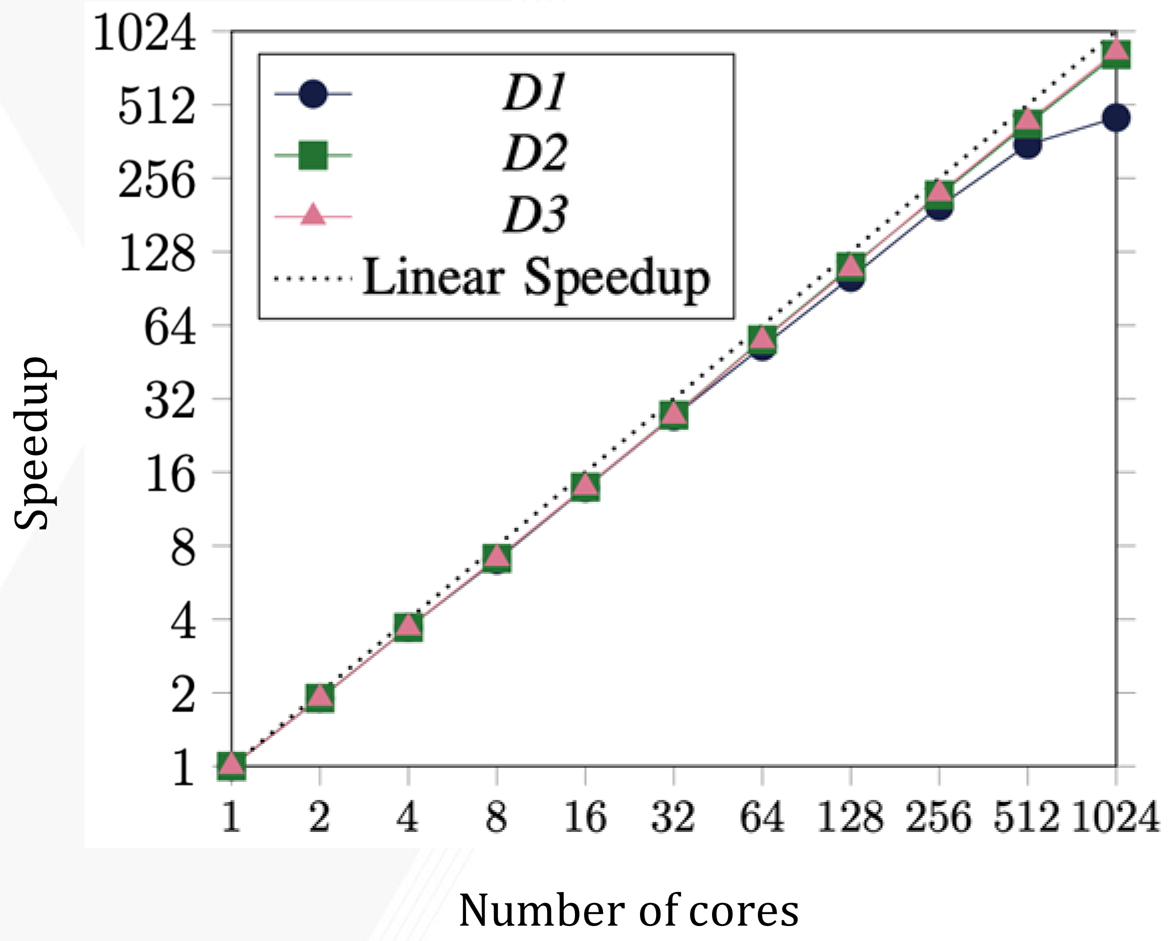
- Sequential comparison with *bnlearn*
  - Popular library for learning BNs; *C* implementation interfaces with *R*

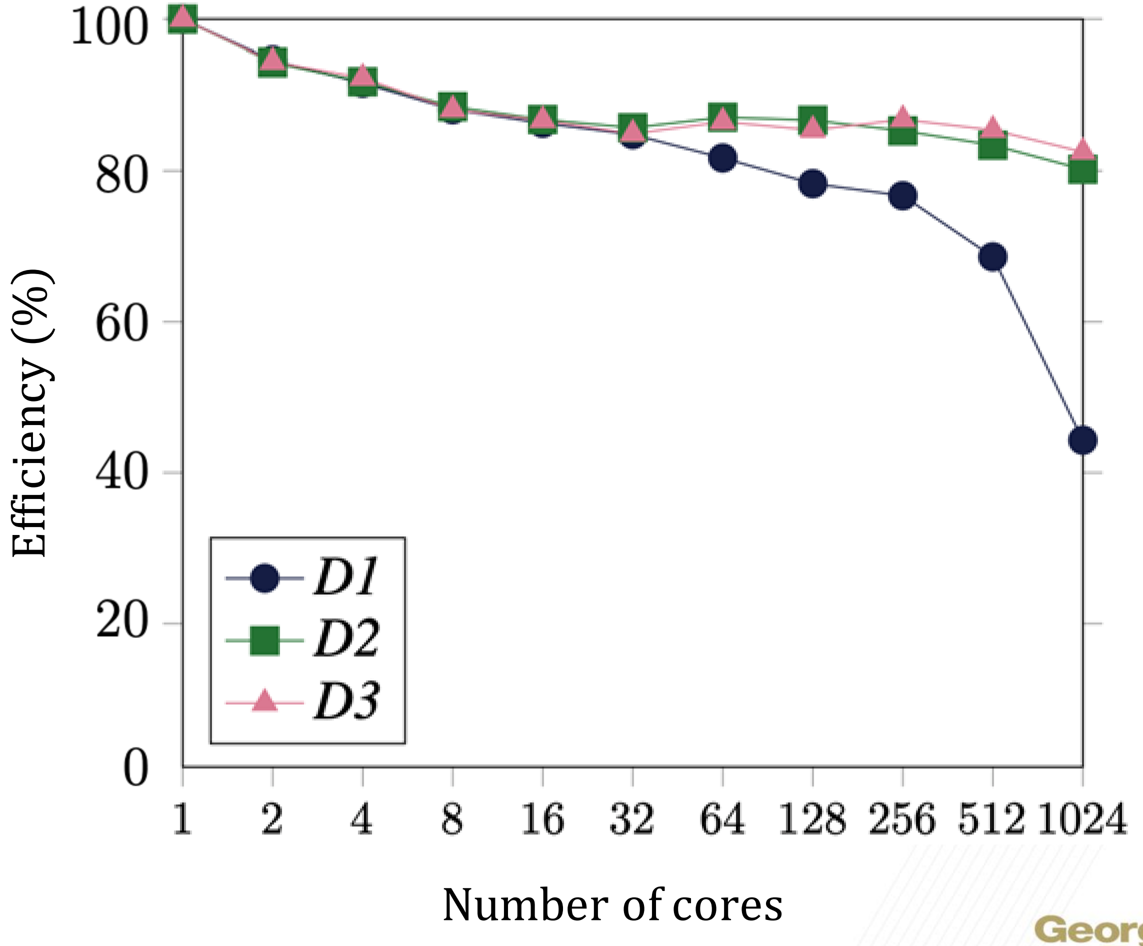
- BNs learned by our implementations are similar to those by *bnlearn*
  - Recalled 99.84% edges with a precision of 99.92% for $D1$ data set
  - Changes in the ordering of the variables caused these differences

- Parallelism in *bnlearn* yields diminishing returns beyond a single node
  - e.g., *IAMB* shows a self-speedup of 3.4X on 16 cores for $D3$ data set while the self-speedup using 64 cores on four nodes is 3.9X

Experiments & Results

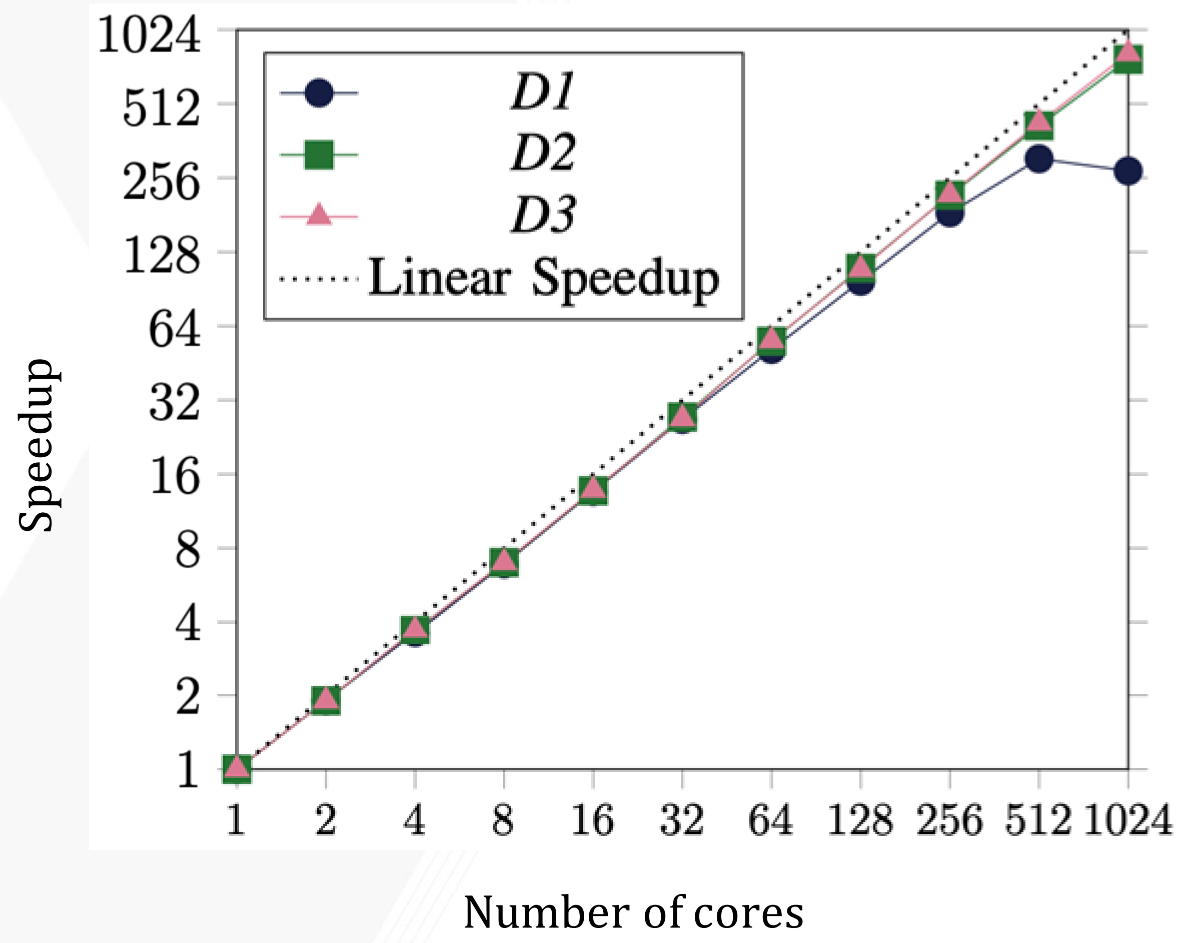
- Strong scaling of our framework – *IAMB*

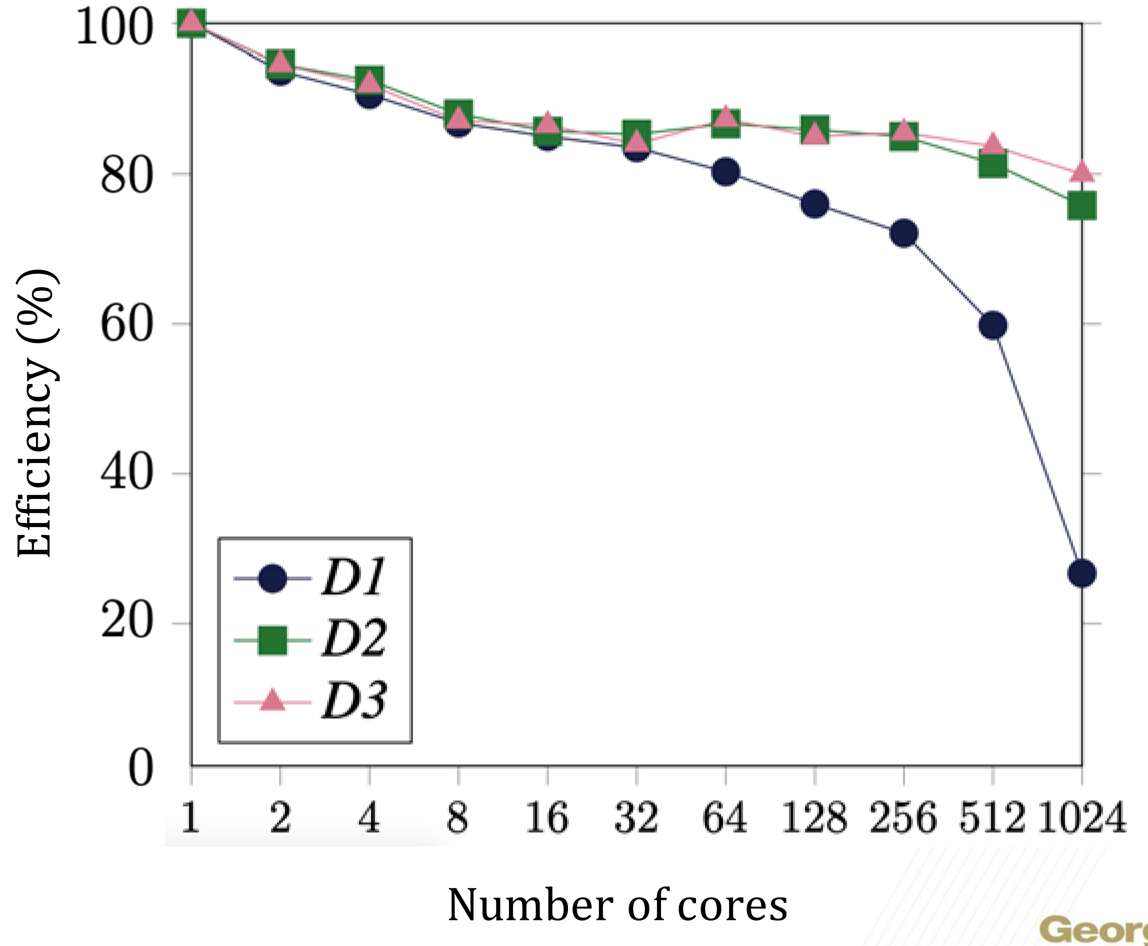




Experiments & Results

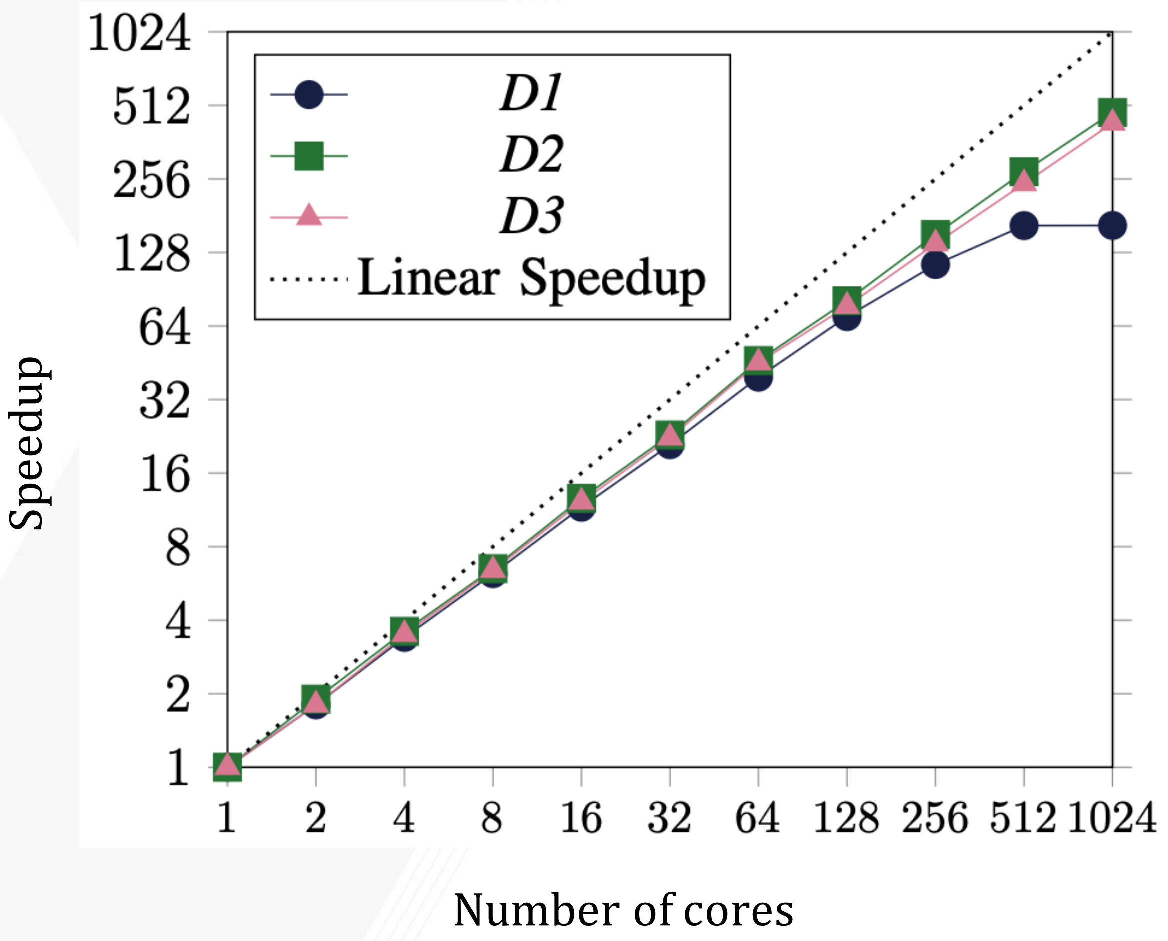
- Strong scaling of our framework – *Inter-IAMB*

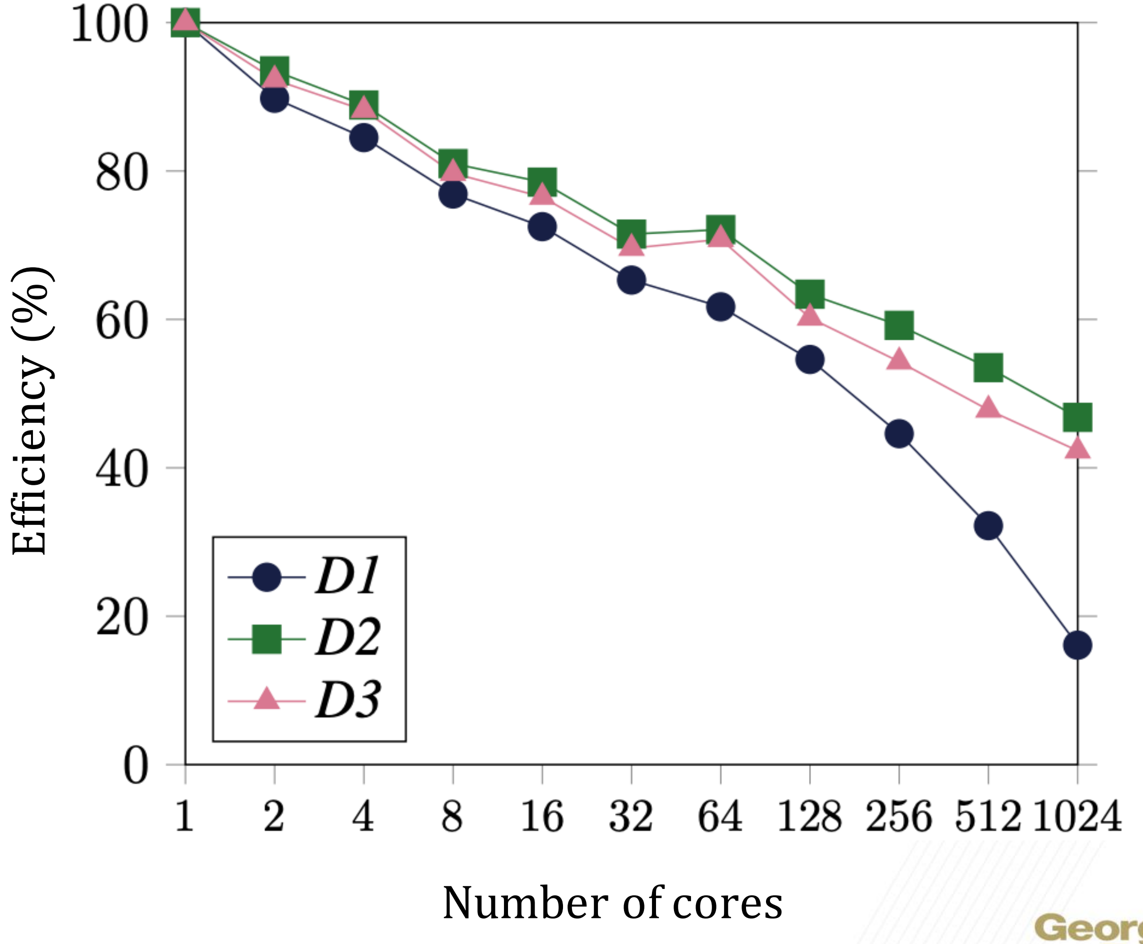




Experiments & Results

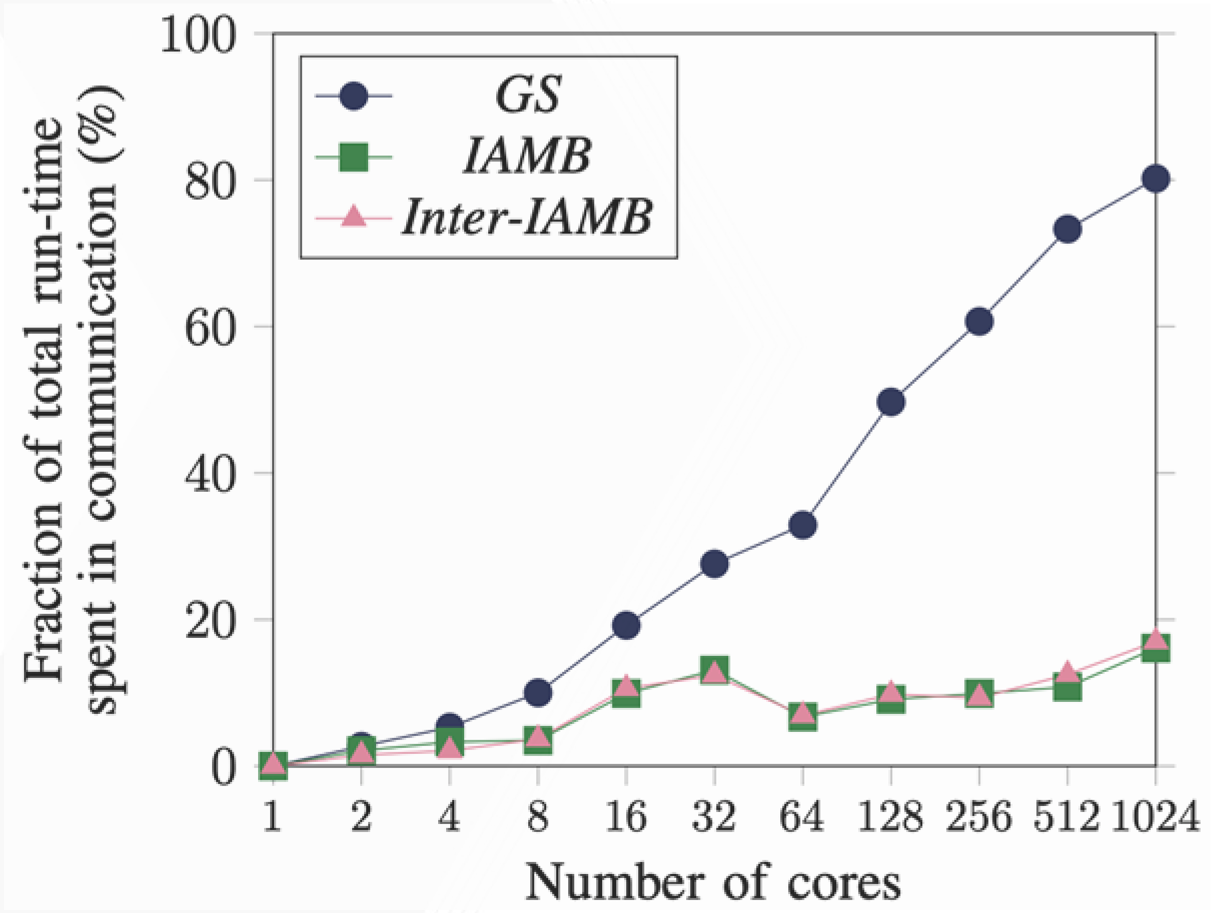
- Strong scaling of our framework – *GS*

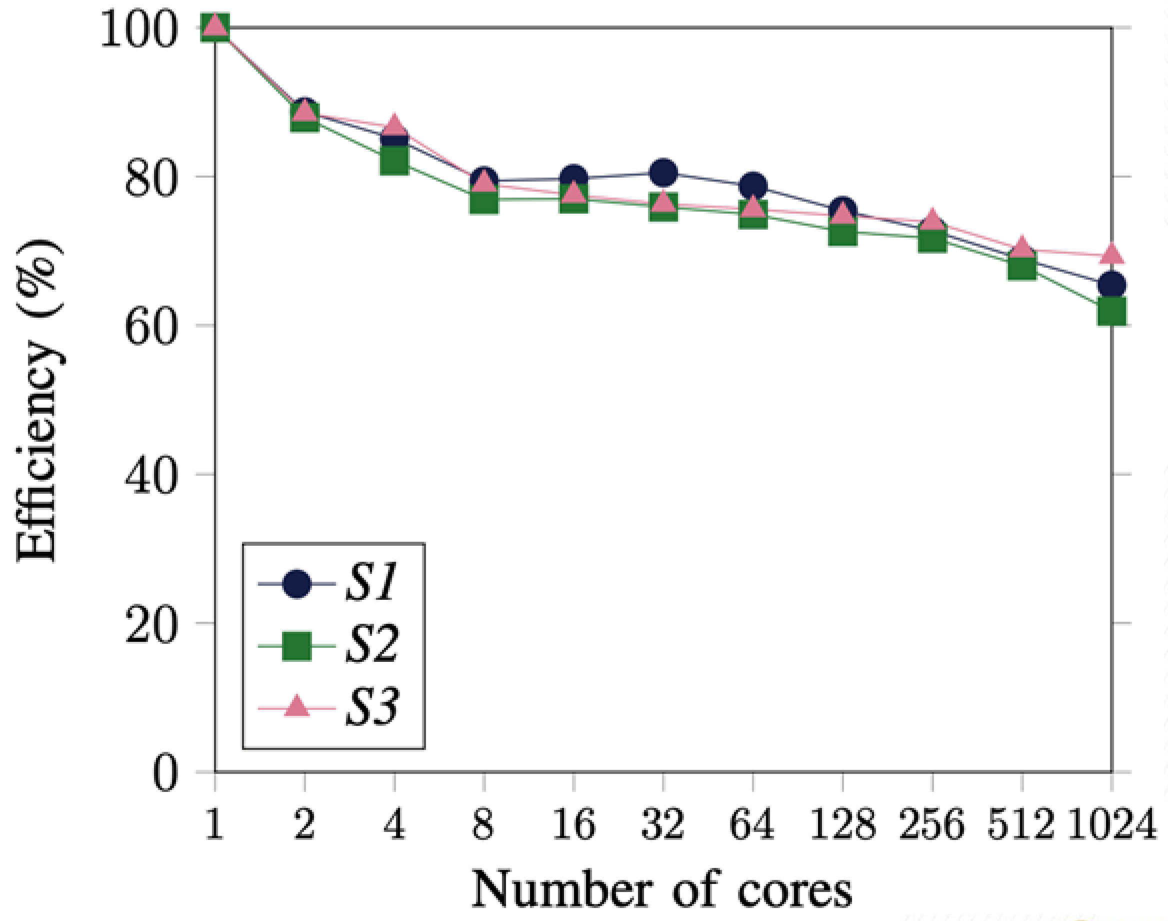




Experiments & Results

- Investigating the scaling performance of *GS*





Experiments & Results

- The algorithms implemented using our proposed parallel framework can learn genome-scale gene networks in less than a minute
  - Maximum speedup of 844.8X and 82.5% scaling efficiency on 1024 cores
  - *IAMB* and *Inter-IAMB* show a sustained efficiency of >75% for $D2$ and $D3$
- Learning BNs from simulated data sets takes less than two minutes on 1024 cores, as compared to more than a day sequentially
  - Maximum speedup of 845X and 82.5% scaling efficiency on 1024 cores
  - GS shows an improved efficiency of >60% for all the data sets

Conclusions

- Proposed a framework for parallelizing multiple BN structure learning algorithms that rely on MB discovery as an intermediate step
  - The algorithms implemented using the framework show good scalability
- Able to learn gene networks from large real data sets in <1 minute
  - Beneficial for biologists who want to iteratively search for the best BN
- Showed good scaling for learning BNs from even larger simulated data sets
  - Potential for use in previously unexplored application areas of BNs

Future Works

- Extend the framework to support other classes of algorithms
- Implement different CI tests for discrete as well as continuous data
- Enable working with distributed data sets

Thank you!

Link: https://github.com/asrivast28/ramBLe

Email: asrivast@gatech.edu