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

Ankit Srivastava

SC21
St. Louis, MO | science & beyond.

Georgia Tech

Motivation

- Machine Learning (ML) models are being used for decision-making in a diverse set of fields – spam detection, recommender systems, etc.
    - “Black box” models are typically used for the purpose

Image Source: *XKCD – Machine Learning*
https://xkcd.com/1838/

Motivation

- Machine Learning (ML) models are being used for decision-making in a diverse set of fields – spam detection, recommender systems, etc.
  - “Black box” models are typically used for the purpose – NOT interpretable

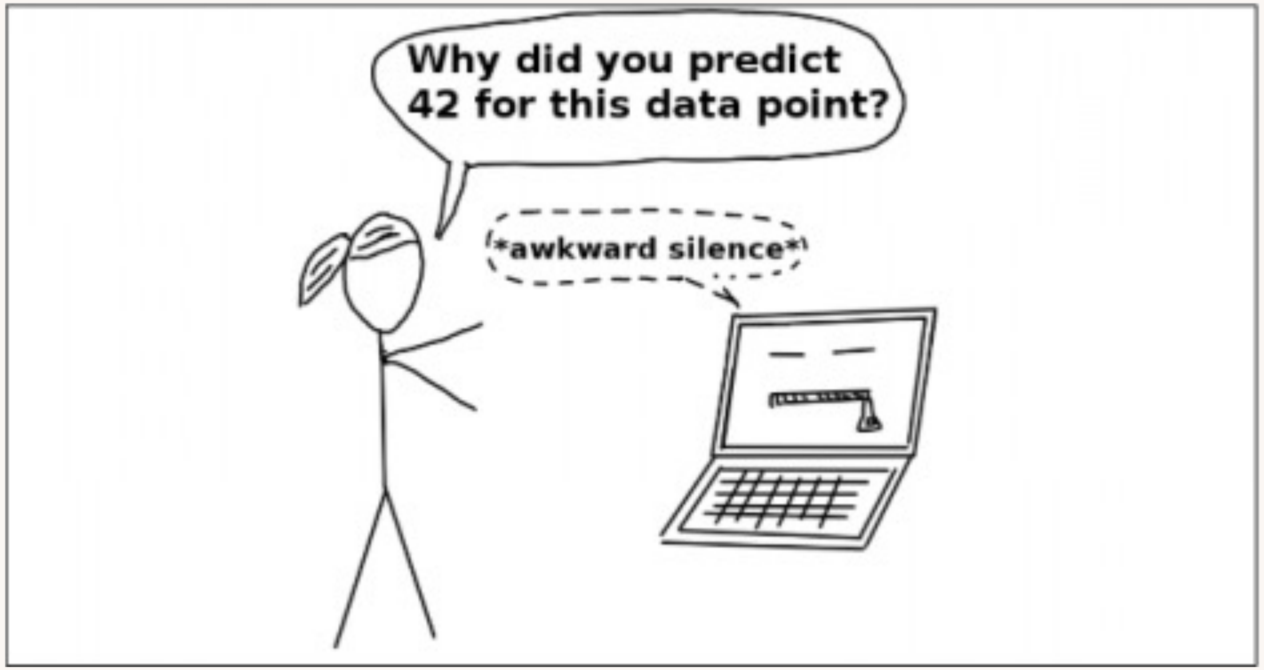


Image Source: *Interpretable Machine Learning — Fairness, Accountability, and Transparency in ML systems*
https://medium.com/@ODSC/interpretable-machine-learning-fairness-accountability-and-transparency-in-ml-systems-3ab45d961fbc

Motivation

- Machine Learning (ML) models are being used for decision-making in a diverse set of fields – spam detection, recommender systems, etc.
  - “Black box” models are typically used for the purpose – NOT interpretable
- Increasingly, ML is being used in high human-impact areas, e.g., criminal justice, healthcare, law enforcement, etc.
  - Apprehensions regarding use of black box models in these areas is growing


The New York Times

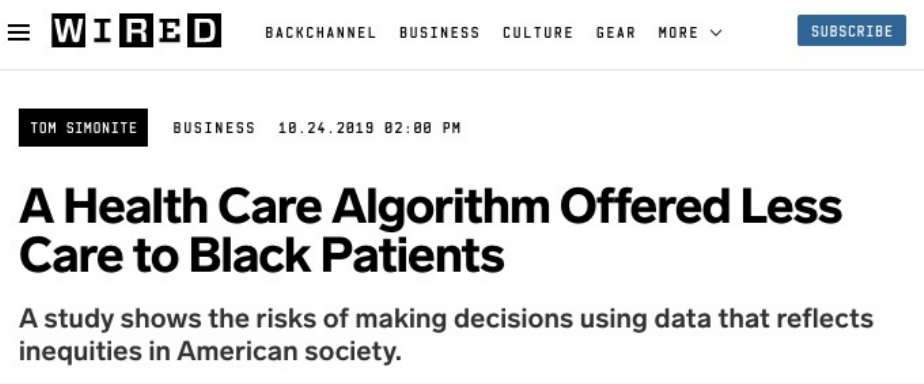
Opinion

OP-ED CONTRIBUTOR

When a Computer Program Keeps You in Jail

By Rebecca Wexler

June 13, 2017


WIRED BACKCHANNEL BUSINESS CULTURE GEAR MORE SUBSCRIBE

TOM SIMONITE BUSINESS 10.24.2019 02:00 PM

A Health Care Algorithm Offered Less Care to Black Patients

A study shows the risks of making decisions using data that reflects inequities in American society.


CBSN ORIGINALS

By ELAISHA STOKES / CBS NEWS / November 19, 2020, 7:00 AM

Wrongful arrest exposes racial bias in facial recognition technology

Motivation

- Machine Learning (ML) models are being used for decision-making in a diverse set of fields – spam detection, recommender systems, etc.
  - “Black box” models are typically used for the purpose – NOT interpretable
- Increasingly, ML is being used in high human-impact areas, e.g., criminal justice, healthcare, law enforcement, etc.
  - Interpretable ML models are the need of the hour

REUTERS

U.S. MARKETS MARCH 29, 2021 / 2:02 PM / UPDATED 4 MONTHS AGO

U.S. banking regulators seek input on how firms rely on artificial intelligence

THE NATIONAL LAW REVIEW

FTC Issues New Guidance, Warning That Bias in Artificial Intelligence Could Create Potential Liability for Enforcement Actions

Saturday, April 24, 2021

FORTUNE

TECH • ARTIFICIAL INTELLIGENCE

Europe proposes strict A.I. regulation likely to have an impact around the world

BY JEREMY KAHN

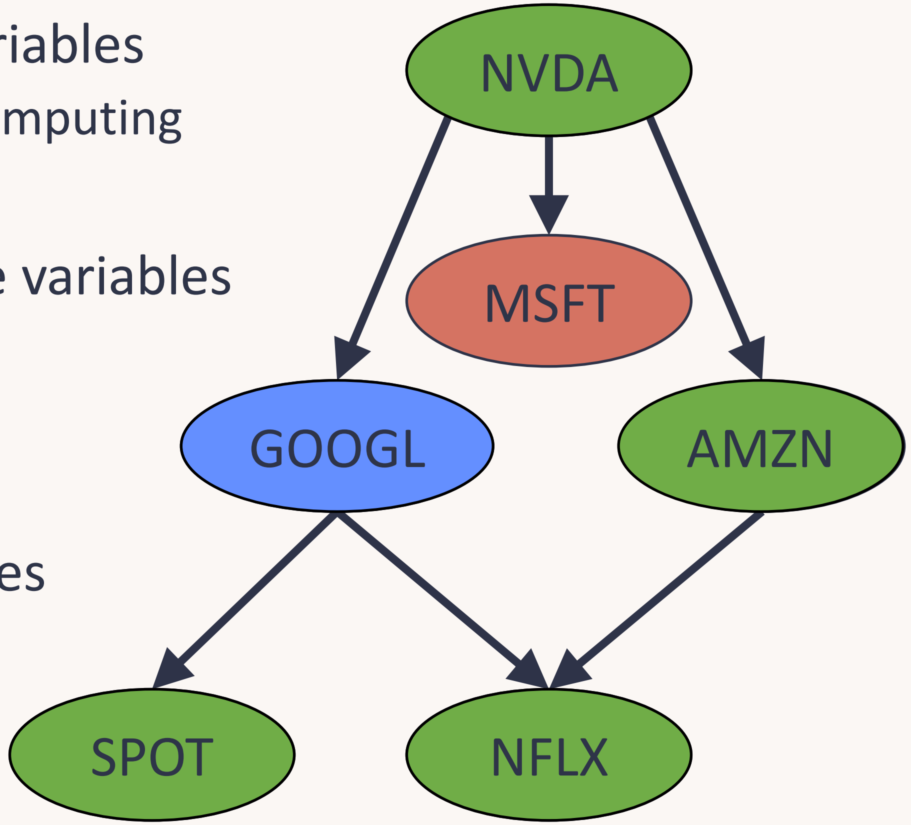
April 21, 2021 7:48 AM EDT

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*
- Parallelization strategies have been proposed for various BN learning algorithms – difficult to integrate disparate strategies
    - Parallel library with support for multiple algorithms is desirable

Background – Bayesian Networks

- BN structure represents dependence graph of a set of variables
  - Case study – Stock prices of companies related to cloud computing
- **Parents and Children (PC)** set of a variable consists of the variables that are dependent on it, given any conditioning set
  - e.g., $PC(\text{GOOGL}) = \{\text{NVDA, SPOT, NFLX}\}$
- **Markov blanket (MB)** of a variable consists of the variables that render the variable independent of other variables
  - Assuming *faithfulness*
    $MB(X) = PC(X) \cup (Parents(Y)\ \forall Y \in Children(X))$
    $\Rightarrow MB(\text{GOOGL}) = \{\text{NVDA, SPOT, NFLX, AMZN}\}$



Blanket Learning Algorithms

- ***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

- ***Blanket learning*** algorithms are *constraint-based* algorithms that first learn MB sets of all the variables separately to get the BN structure
  - *Grow-Shrink* (***GS***) (Margaritis and Thrun, 2000)
  - *Incremental Association MB* (***IAMB***) (Tsamardinos et al., 2003)
  - *Interleaved IAMB* (***Inter-IAMB***) (Tsamardinos et al., 2003)

Blanket Learning Algorithms

- Use variations of the *Grow-Shrink* scheme for learning MB sets
  - *Grow* phase: Add variables to candidate MB sets
  - *Shrink* phase: Remove false positive variables from candidate MB sets
- Differ in the specifics of how the scheme is iterated
  - Choosing variables to be added in *Grow* phase
    - *IAMB* and *Inter-IAMB* pick the “most dependent” variable given the current candidate MB set
    - *GS* picks the first dependent variable
  - Order of *Grow* and *Shrink* phases
    - *GS* and *IAMB* execute multiple iterations of *Grow* phase followed by one *Shrink* phase
    - *Inter-IAMB* interleaves the execution of *Grow* and *Shrink* phases in every iteration
- Perform *symmetry correction* for MB sets ($X \in MB(T) \Leftrightarrow T \in MB(X)$)
- Learn *PC from MB* sets (PC ⊆ MB)

Related Works

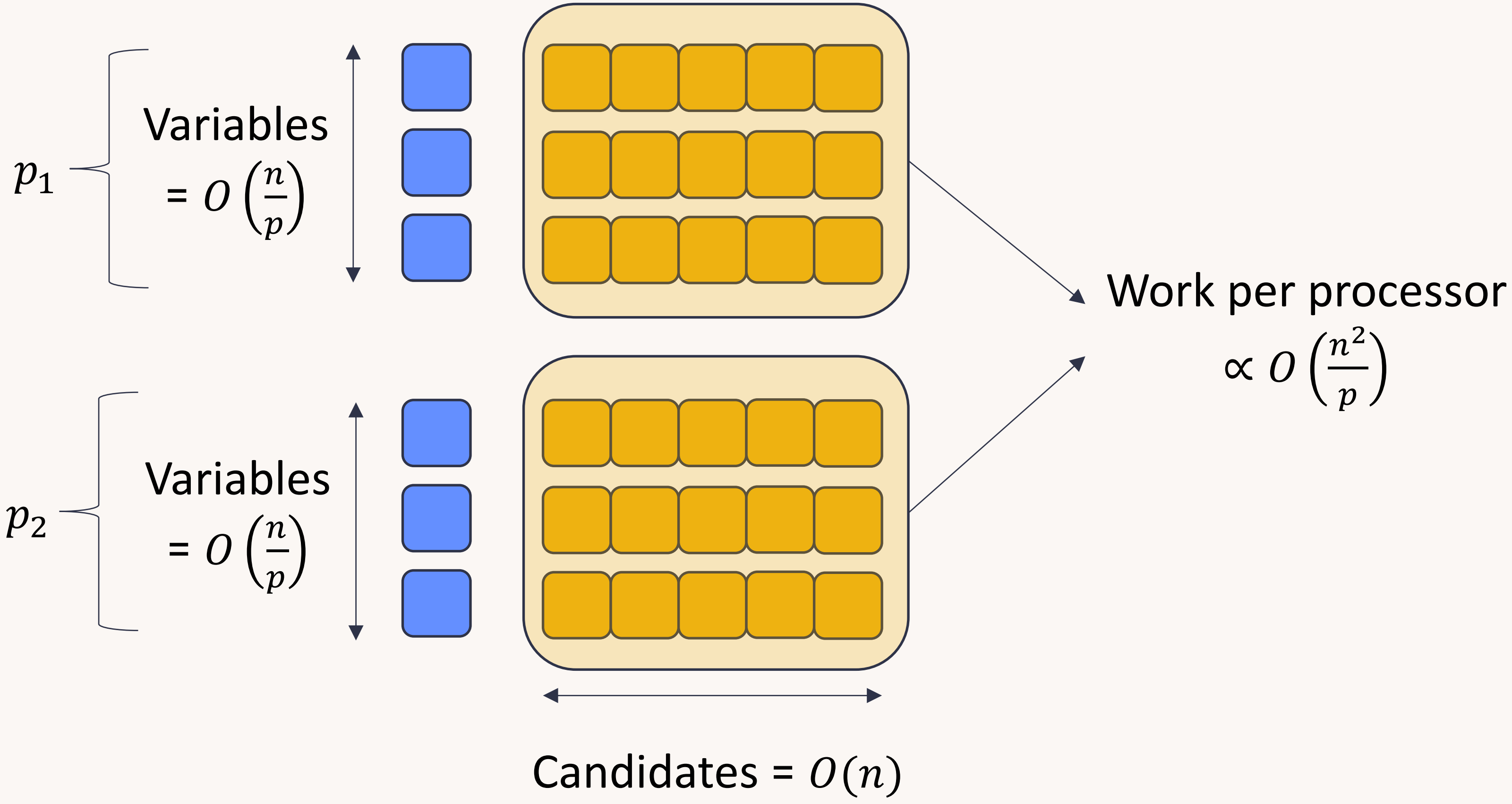
- Nikolova et al. (2011) parallelized two similar *constraint-based* algorithms: *MMPC* (Tsamardinos et al., 2006) and *GetPC* (Peña et al., 2007)
  - Scales well up to 512 cores for learning neighborhoods of 1,000 variables
  - Scaling tapers off as the number of cores or variables are increased

- *bnlearn* contains implementations of the three algorithms
  - Scutari et al. (2017) added support for parallelizing the implementations using a master-worker paradigm for small-scale parallelism

- Both these approaches distribute learning of variable neighborhoods

Parallel Framework – Key Design Ideas

- Distribute learning across processors – how?
  - Previous approaches have distributed learning neighborhoods of variables

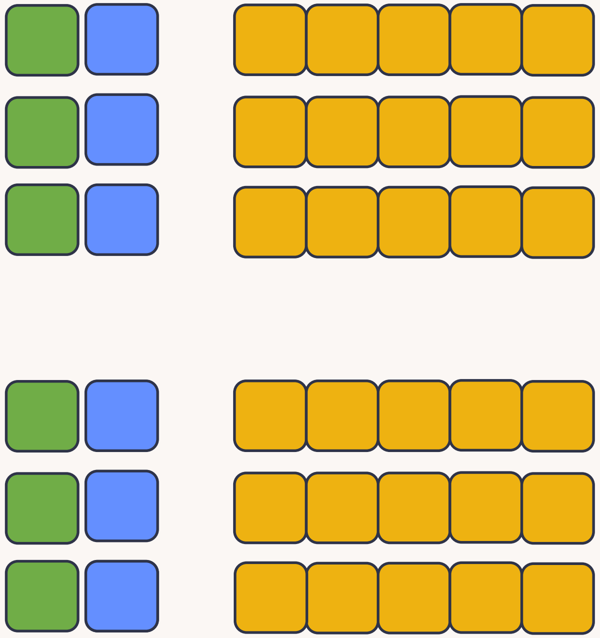


Parallel Framework – Key Design Ideas

- Distribute learning across processors – how?
  - Previous approaches have distributed learning neighborhoods of variables

p_1

p_2

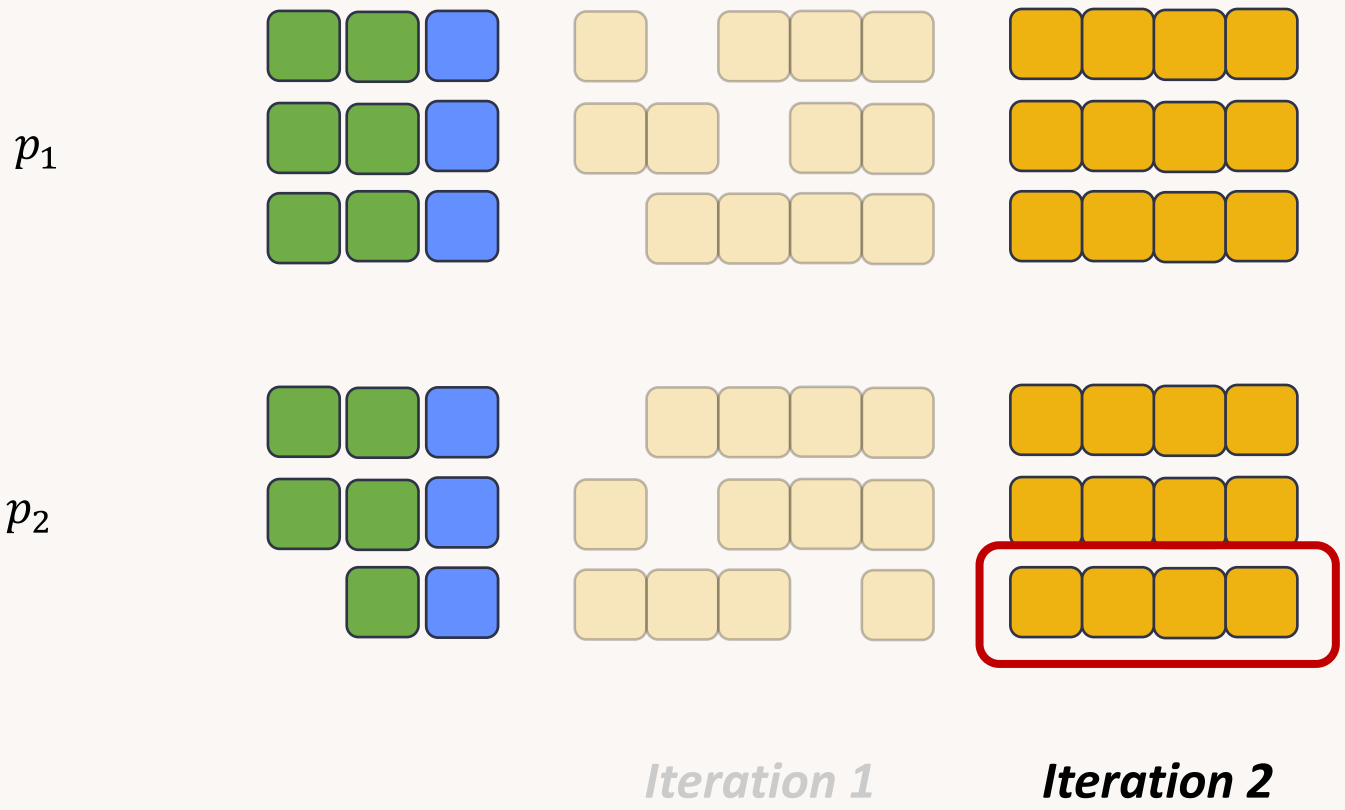


Selected Candidate Variables

Iteration 1

Parallel Framework – Key Design Ideas

- Distribute learning across processors – how?
  - Previous approaches have distributed learning neighborhoods of variables



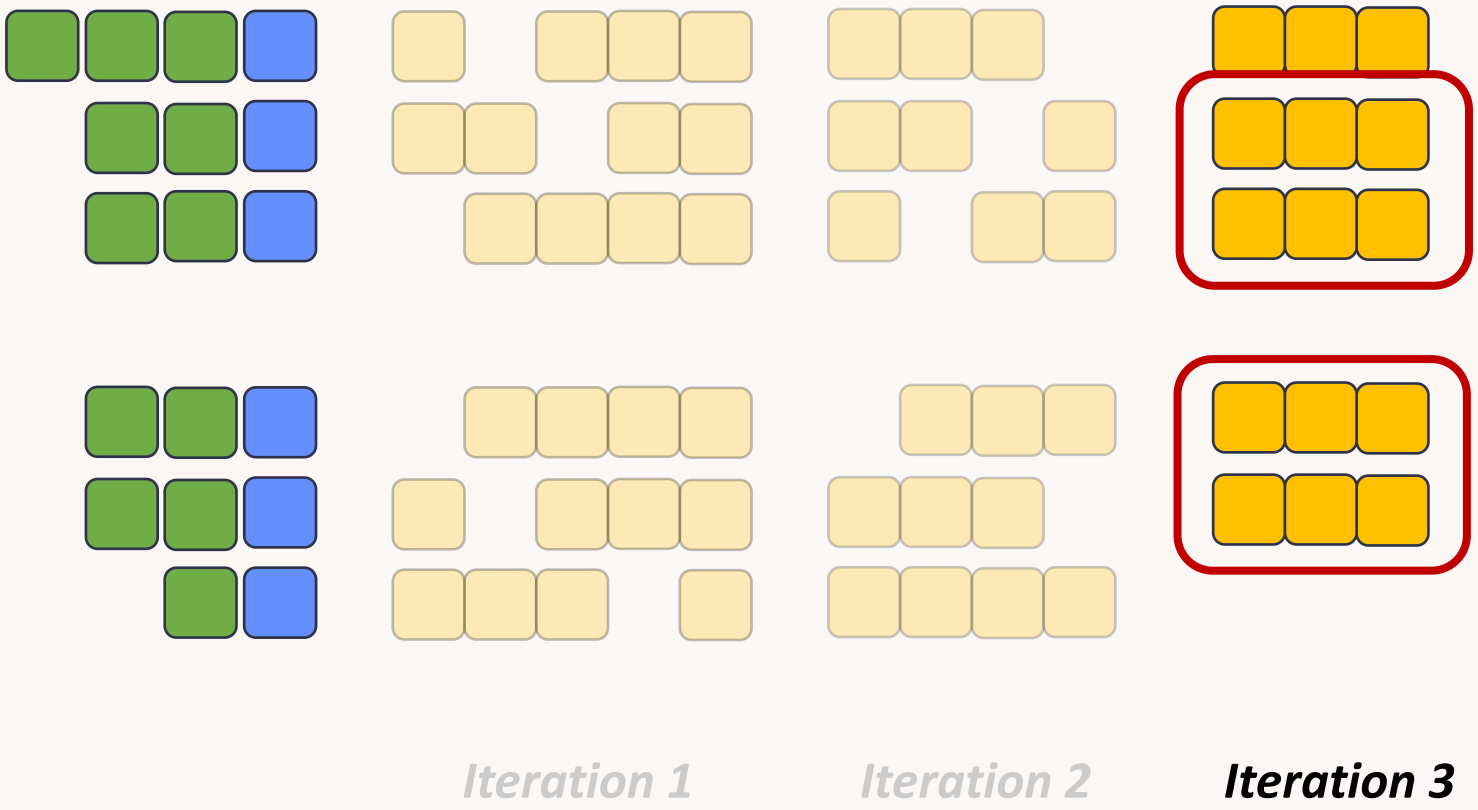
Georgia Tech

Parallel Framework – Key Design Ideas

- Distribute learning across processors – how?
  - Previous approaches have distributed learning neighborhoods of variables

p_1

p_2

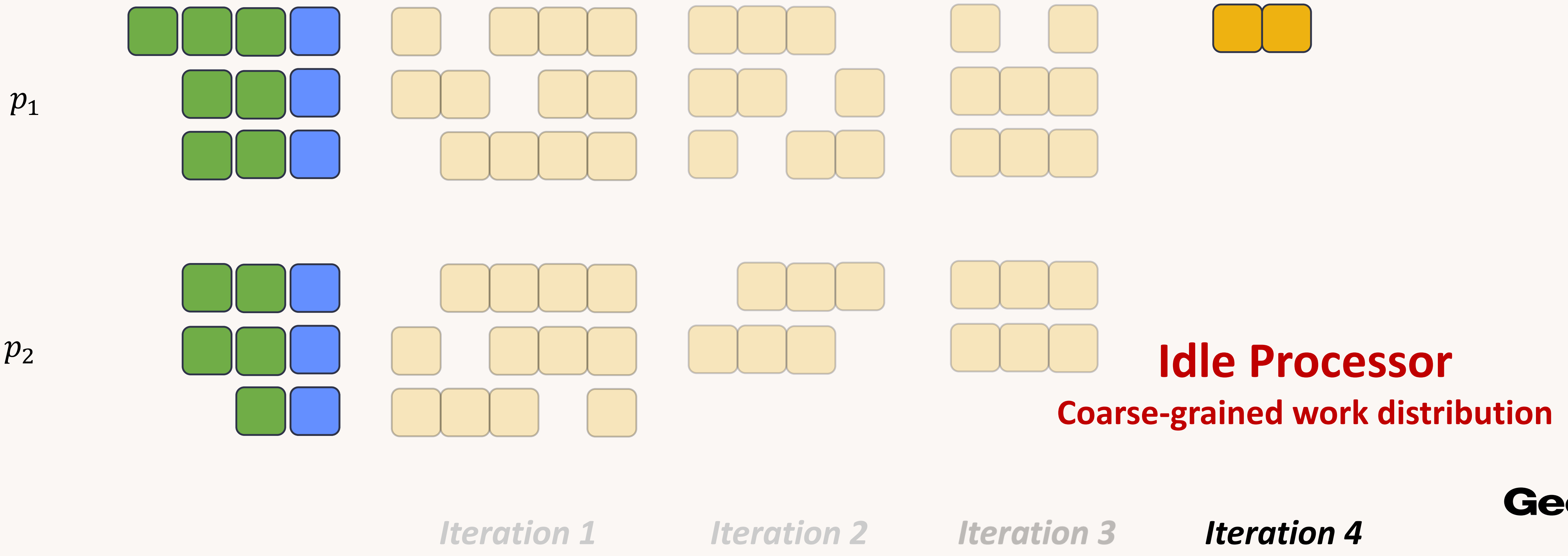


Load Imbalance

Unclear how to fix

Parallel Framework – Key Design Ideas

- Distribute learning across processors – how?
  - Previous approaches have distributed learning neighborhoods of variables



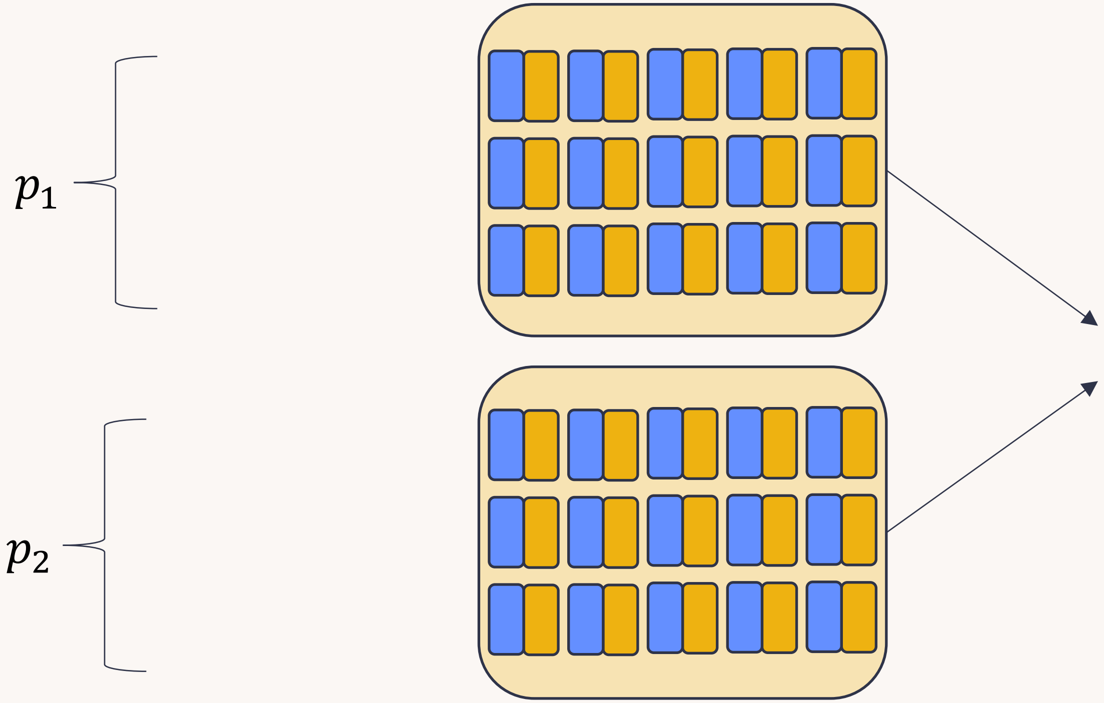
Parallel Framework – Key Design Ideas

- Distribute learning across processors – how?
  - Previous approaches have distributed learning neighborhoods of variables

- ***Observation***: **Variables have different neighborhood sizes – distributing variables to processors is suboptimal**
- ***Idea***: ***Distribute all the target and candidate variable pairs in parallel***

Parallel Framework – Key Design Ideas

- ***Idea*: Distribute all the target and candidate variable pairs in parallel**

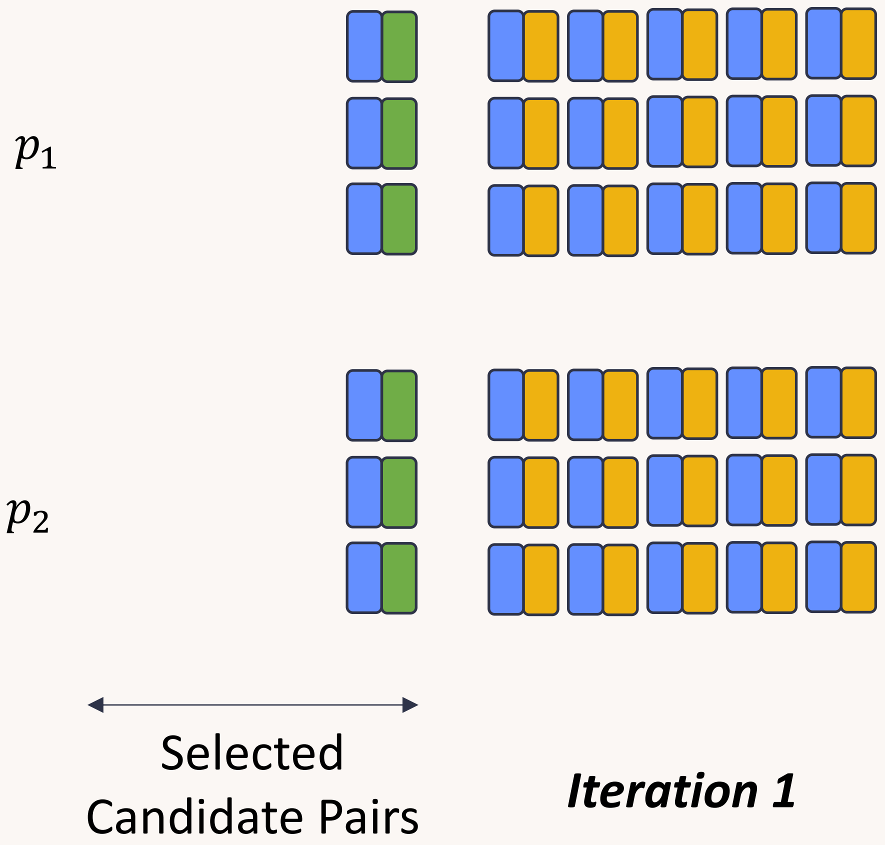


Work per processor ∝ Pairs per processor

$$= O\left(\frac{n^2}{p}\right)$$

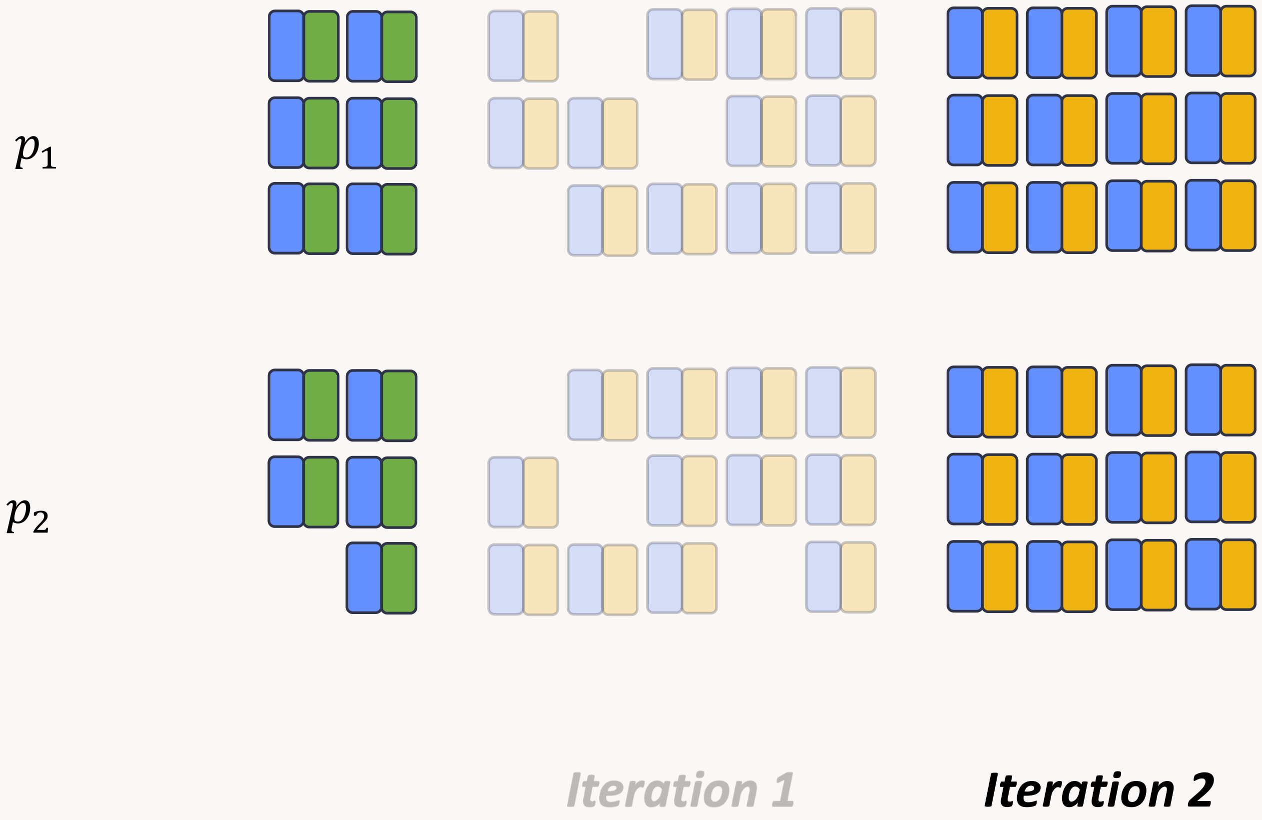
Parallel Framework – Key Design Ideas

- ***Idea*: Distribute all the target and candidate variable pairs in parallel**



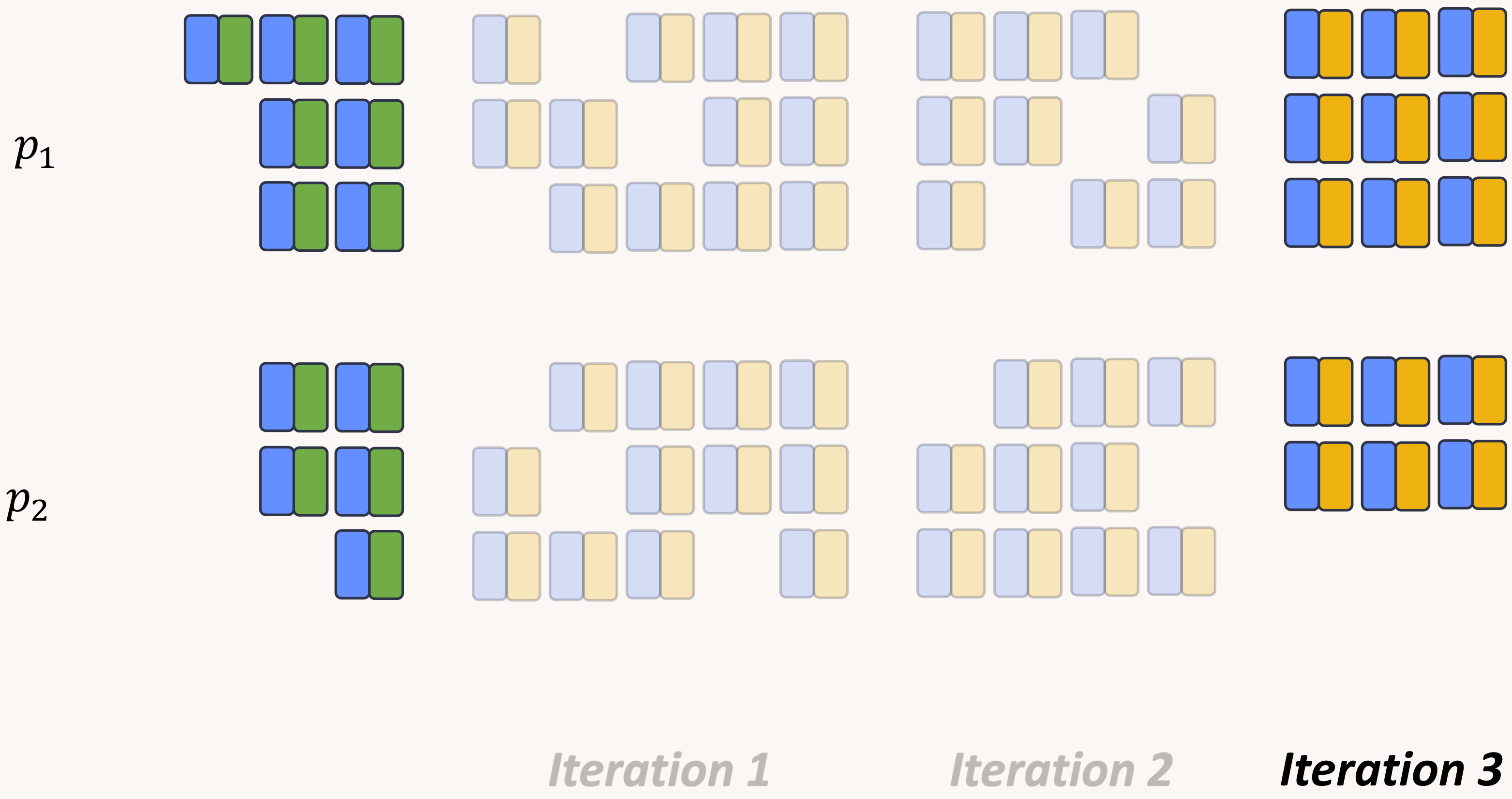
Parallel Framework – Key Design Ideas

- ***Idea*: Distribute all the target and candidate variable pairs in parallel**



Parallel Framework – Key Design Ideas

- ***Idea*: Distribute all the target and candidate variable pairs in parallel**

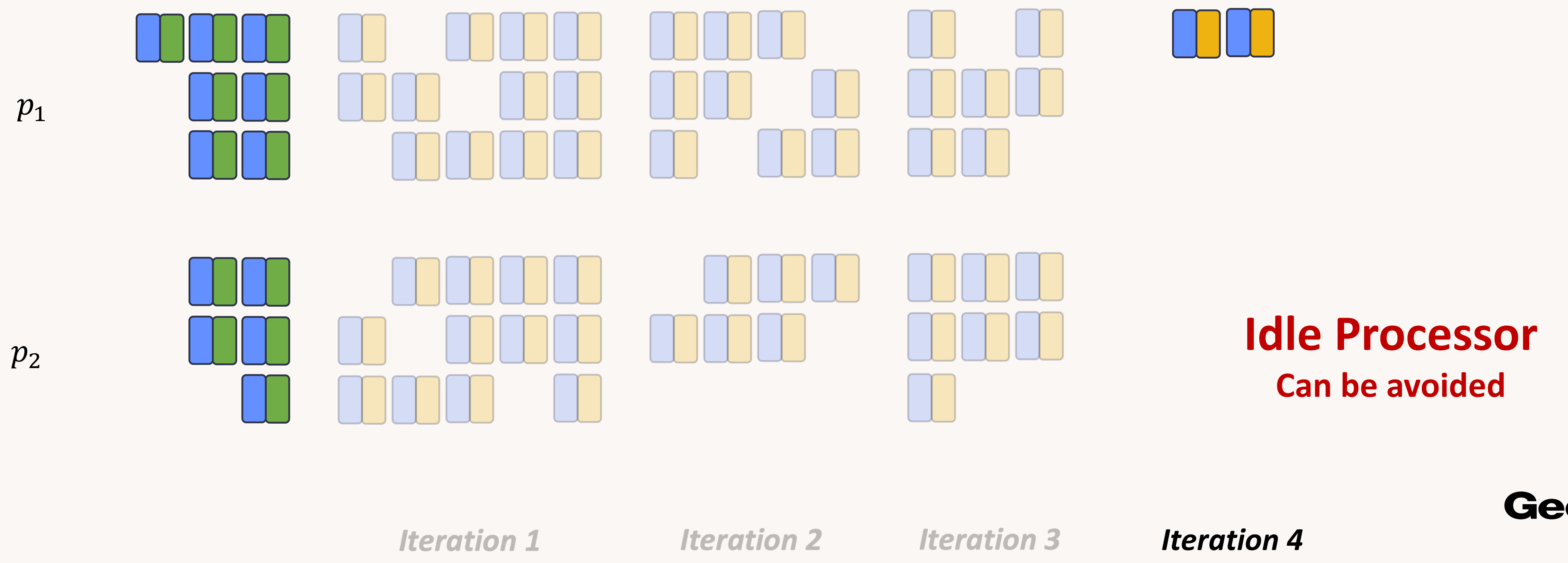


Load Imbalance

Can be alleviated

Parallel Framework – Key Design Ideas

- ***Idea*: Distribute all the target and candidate variable pairs in parallel**



Parallel Framework – Primary Data Structure

- $c\text{–}scores$ is a list of tuples $< X, Y, \theta_{XY} >$ s.t. $X \in \mathcal{X}, Y \in \mathcal{X} \setminus \{X\}$
  - θ_{XY} is the score of Y for addition to the MB set of X
  - Tuples with the same X are contiguously arranged in the list

- Work distribution in parallel by distributing the tuples
  - $c\text{–}scores$ is block-distributed across processors – $c\text{–}scores_j$ on processor j

Parallel Framework – Components

- Parallel *Grow* phase on processor j
  - Update θ_{XY} for all the tuples $\in c\text{–}scores_j$
    - Computation of θ_{XY} is dependent on the algorithm
  - Add Y to the MB of X corresponding to the best θ_{XY}
    - Can be identified using two segmented parallel prefix operations for all the variables
- Parallel *Shrink* phase on processor j
  - Candidate MBs are available for local target variables – no communication
- Parallel *Symmetry Correction* using algorithm by Nikolova et al. (2011)
- Parallel *PC from MB* for local target variables on processor j

Parallel Skeleton – Blanket Learning

function PARALLEL-SKELETON-INTERIAMB():
  Input: $\mathcal{X}$, D, APPLY-HEURISTIC, REDUCE-HEURISTIC
  Output: $\mathcal{PC}(T)$ sets for all $T \in \mathcal{X}$
  parallel j = *processor's rank* **do**
    Initialize *c-scores*$_j$, *variables*$_j$, $\mathcal{MB}(\cdot)$ as described in subsection 3.2.1
    Initialize *neighbors* as empty list of tuples
    repeat
      GROW-PHASE(D, *c-scores*, *variables*, $\mathcal{MB}$, APPLY-HEURISTIC, REDUCE-HEURISTIC)
      + SHRINK-PHASE(D, *variables*, $\mathcal{MB}$)
    until *no* $\mathcal{MB}(X)$ *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
    CONSTRUCT-PC(D, *variables*, $\mathcal{MB}$, *neighbors*)

Implementation

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

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

Experiments and Results

- Experimental setup
  - 64 nodes of the *Hive* cluster, 16 MPI processes per node – **1024 processes**
  - *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 and Results

- Sequential comparison with prior state-of-the-art – *bnlearn*
  - Popular library for learning BNs; *C* implementation interfaces with *R*

Algorithm	Data set	Run-time (s) *bnlearn*	Run-time (s) *Ours*	Speedup
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 and Results

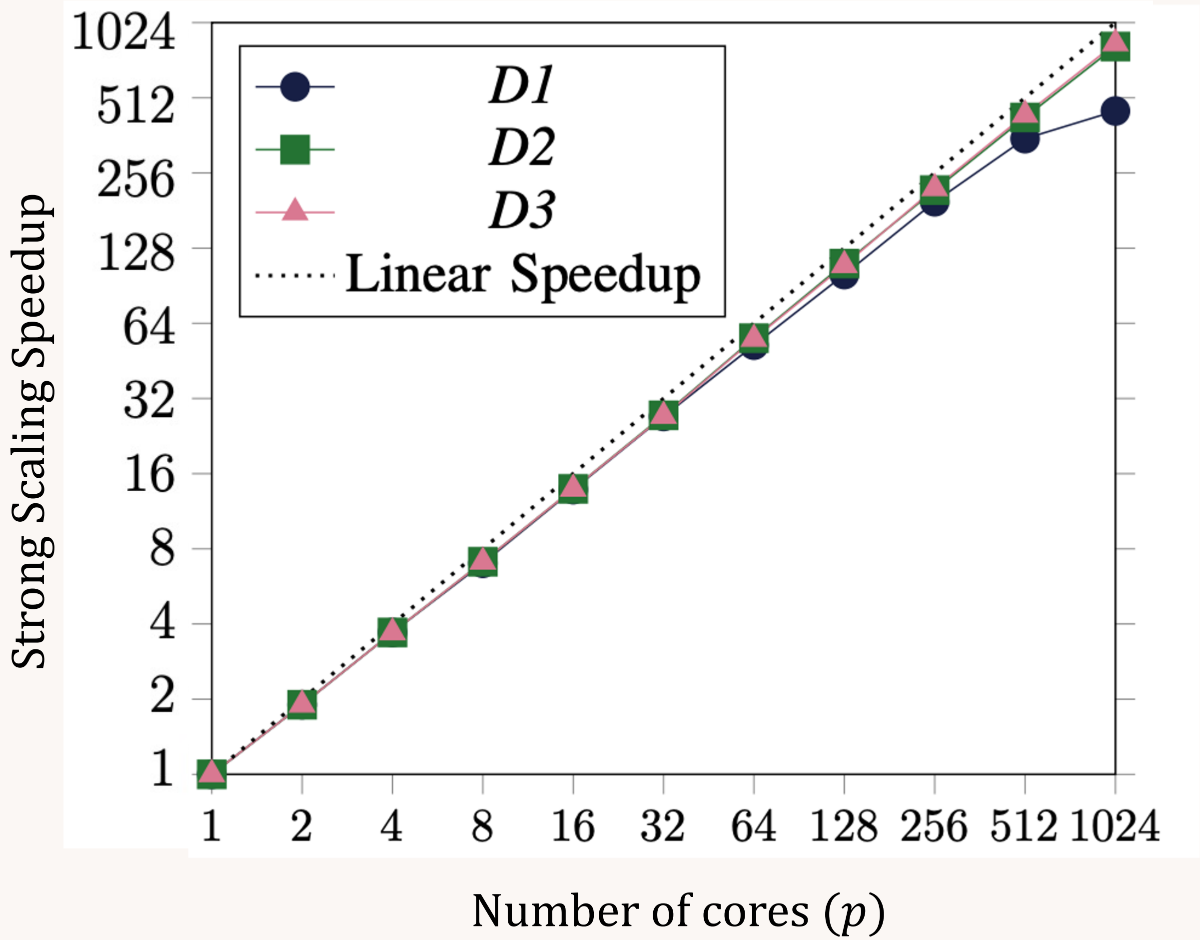
- Sequential comparison with prior state-of-the-art – *bnlearn*
  - Popular library for learning BNs; *C* implementation interfaces with *R*
- BNs learned by our implementations are similar to those by *bnlearn*
  - Recalled 99.84% edges with a precision of 99.92% for $D1$ data set
  - Changes in the ordering of the variables caused the 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

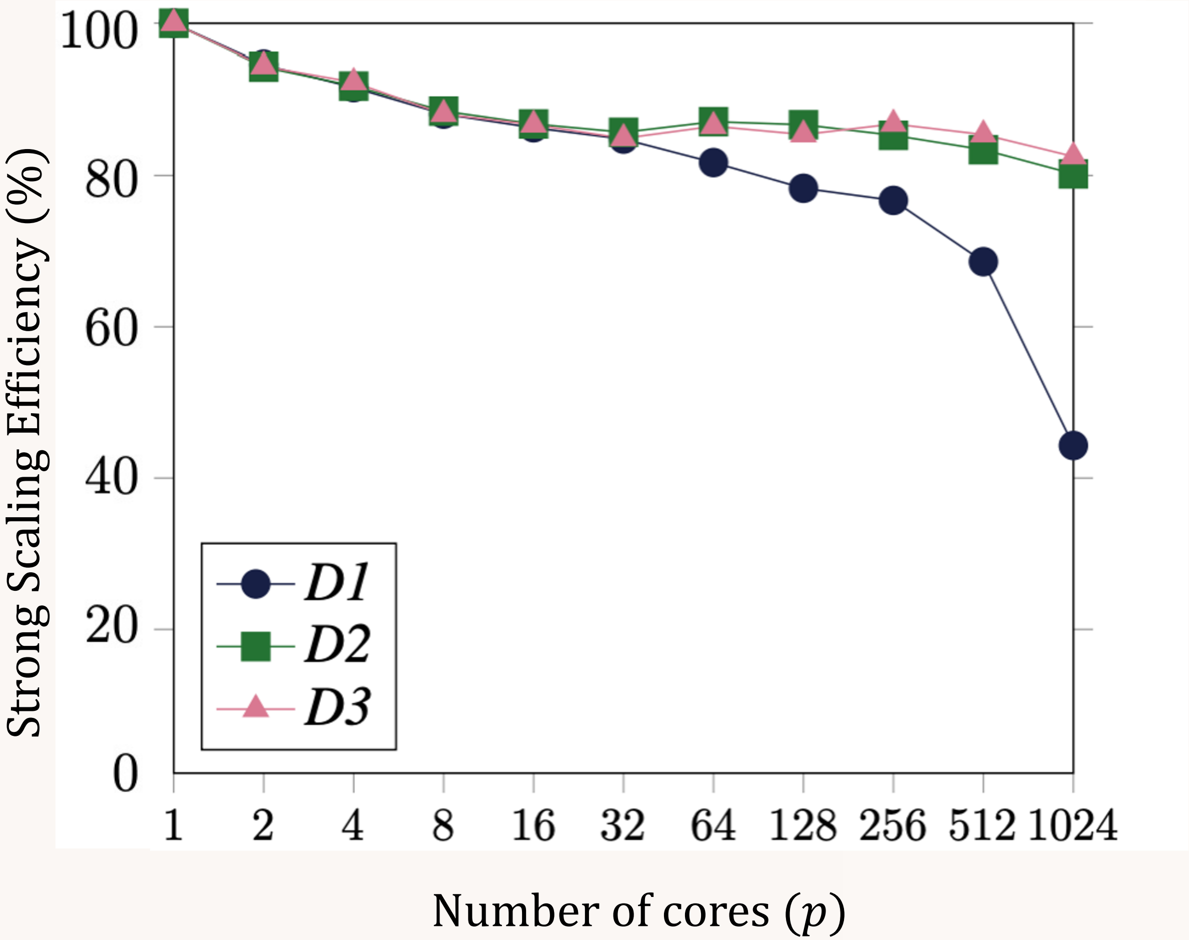
Experiments and Results

- Parallel performance of our framework – notions of scalability
- **Strong Scaling**
  - Fixed total work; how does the run-time scale with increasing parallelism? (n is kept constant as p increases)
- **Weak Scaling**
  - Fixed work per processor; how does the run-time scale with increasing parallelism? (n is increased as p increases)
- Speedup and efficiency are measured
  - Perfect parallel algorithm shows **linear speedup** and **100% efficiency**

Experiments and Results

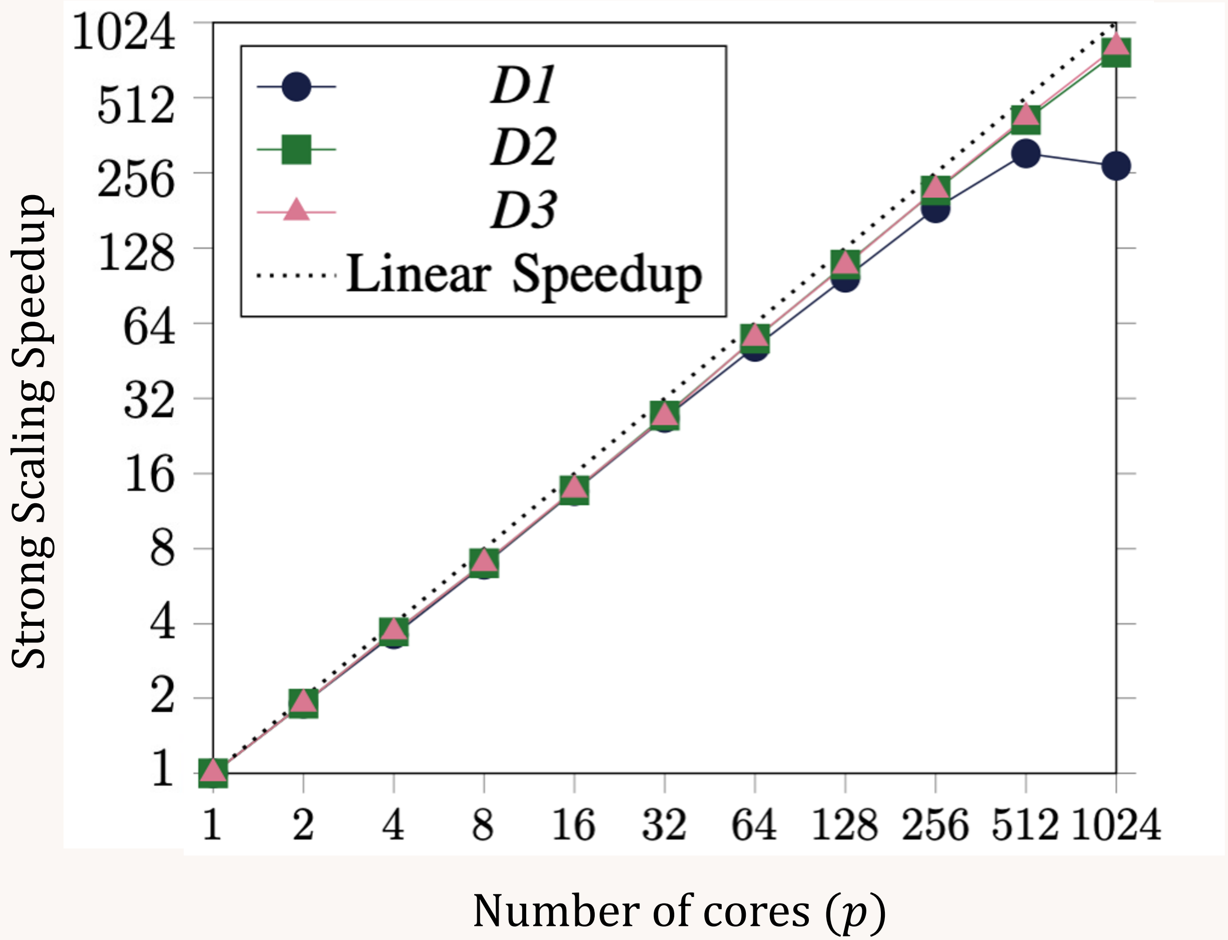
- Strong scaling of our framework – *IAMB*

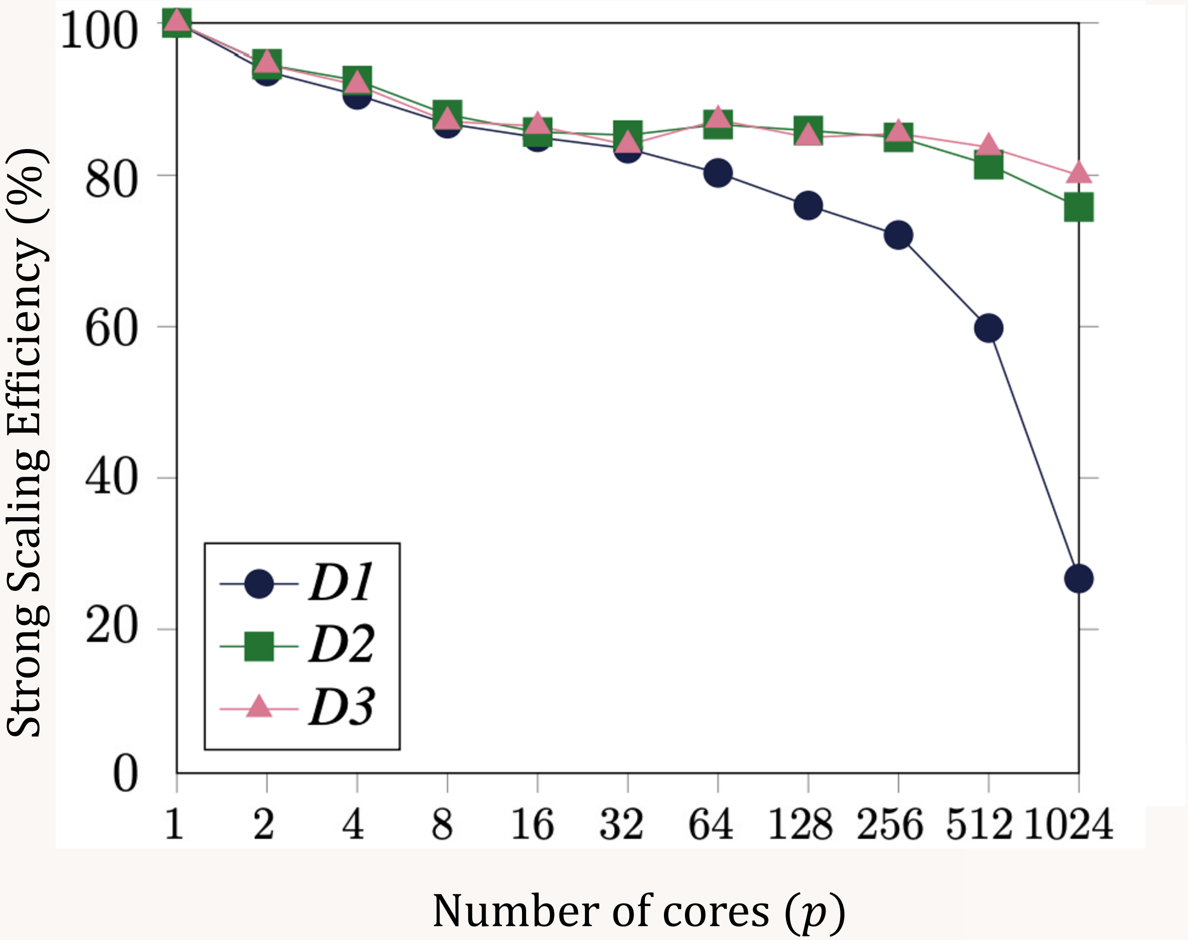




Experiments and Results

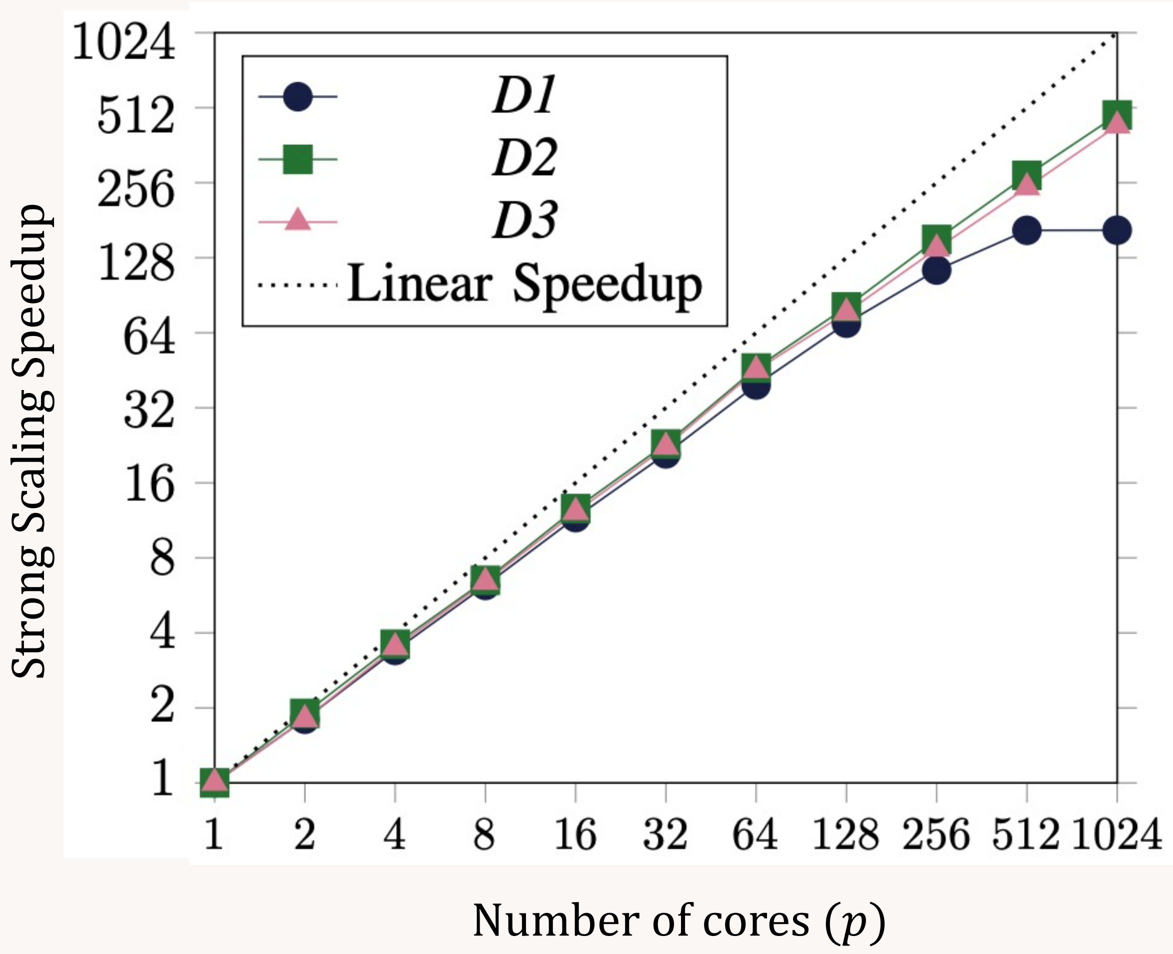
- Strong scaling of our framework – *Inter-IAMB*

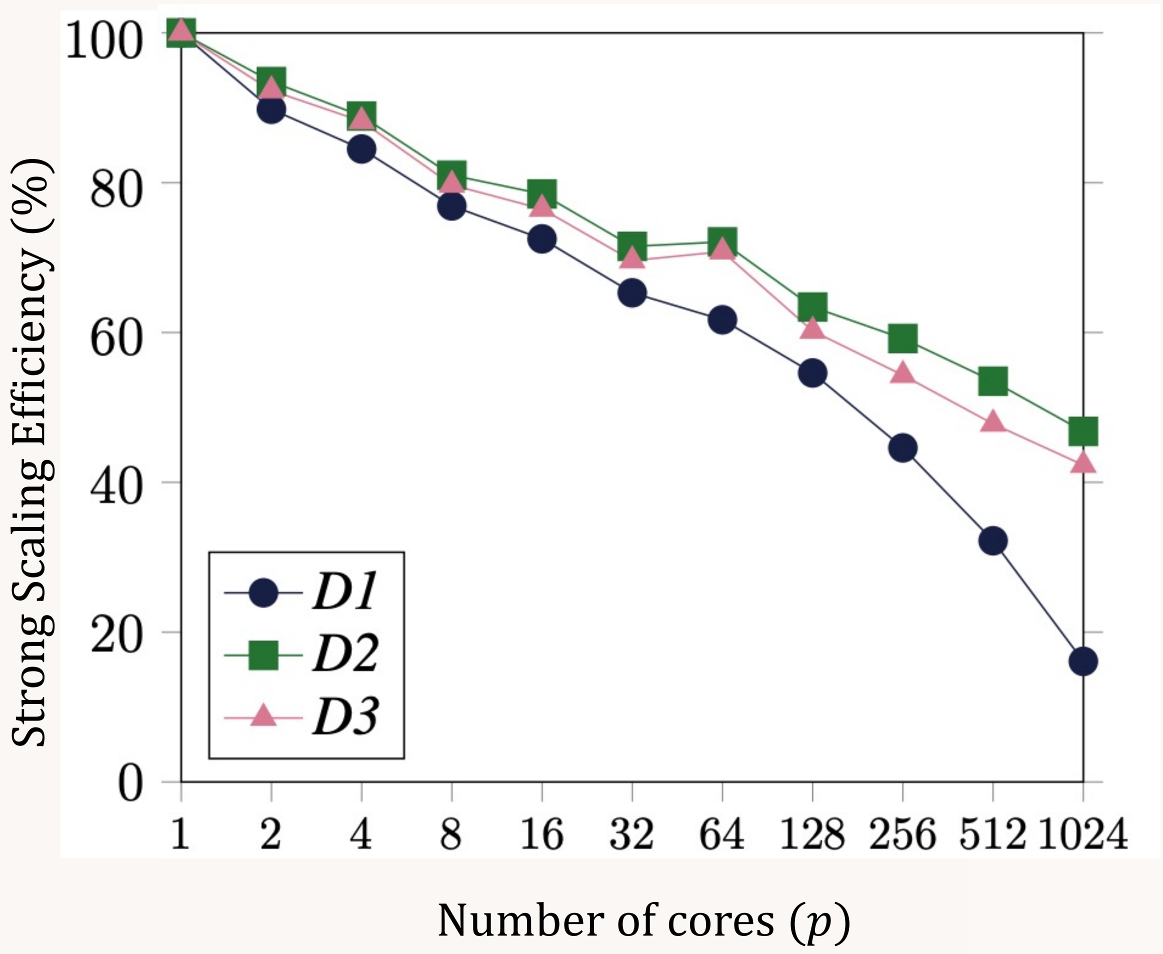




Experiments and Results

- Strong scaling of our framework – *GS*

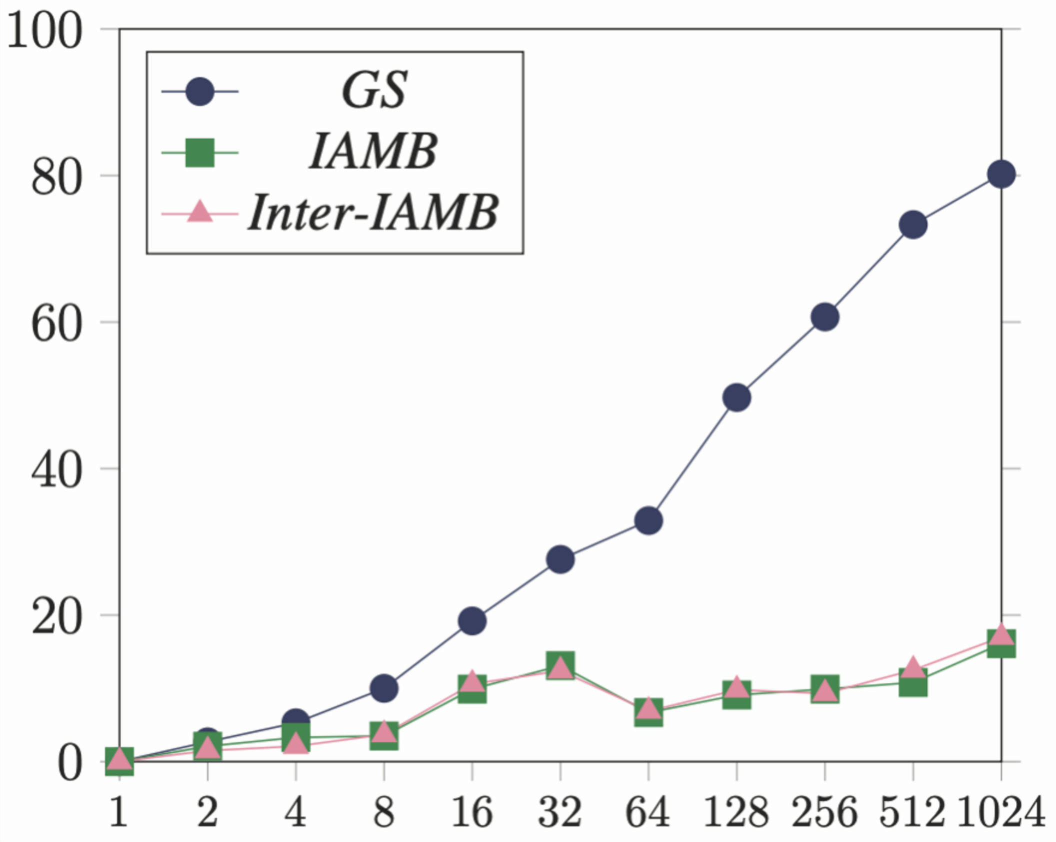




Experiments and Results

- Investigating the scaling performance of *GS*
  - High communication overhead due to lower total work?

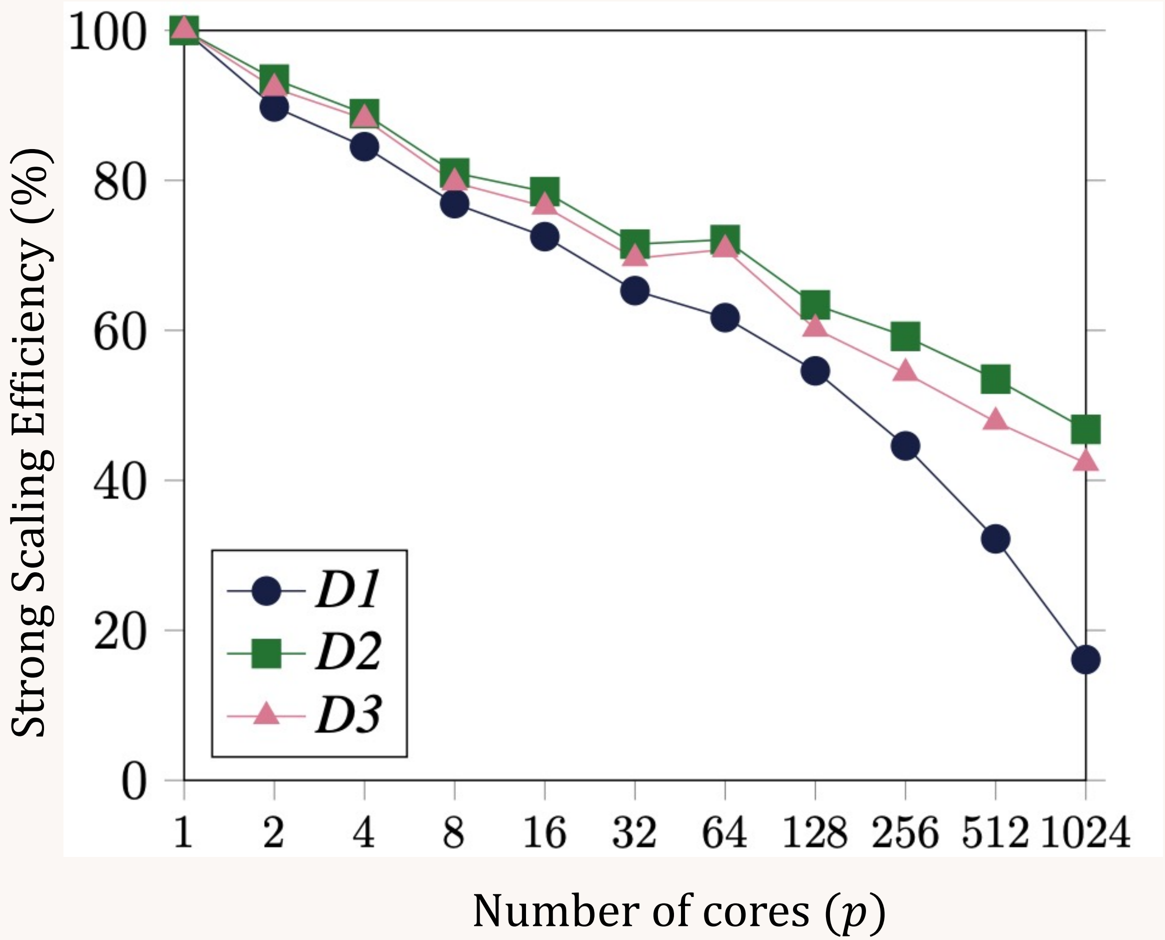
Fraction of total run – time spent in communication (%)

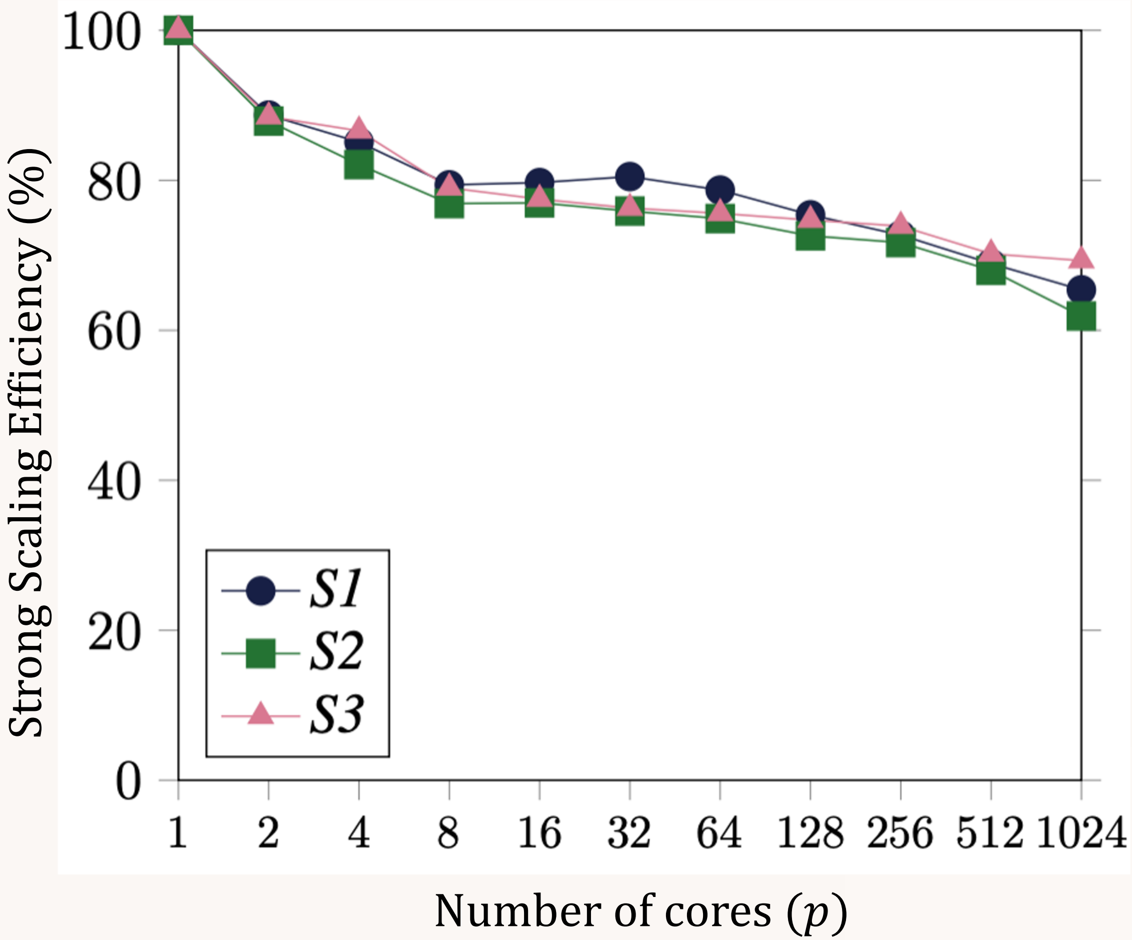


Number of cores (p)

Experiments and Results

- Scaling performance of *GS* – real data versus simulated data



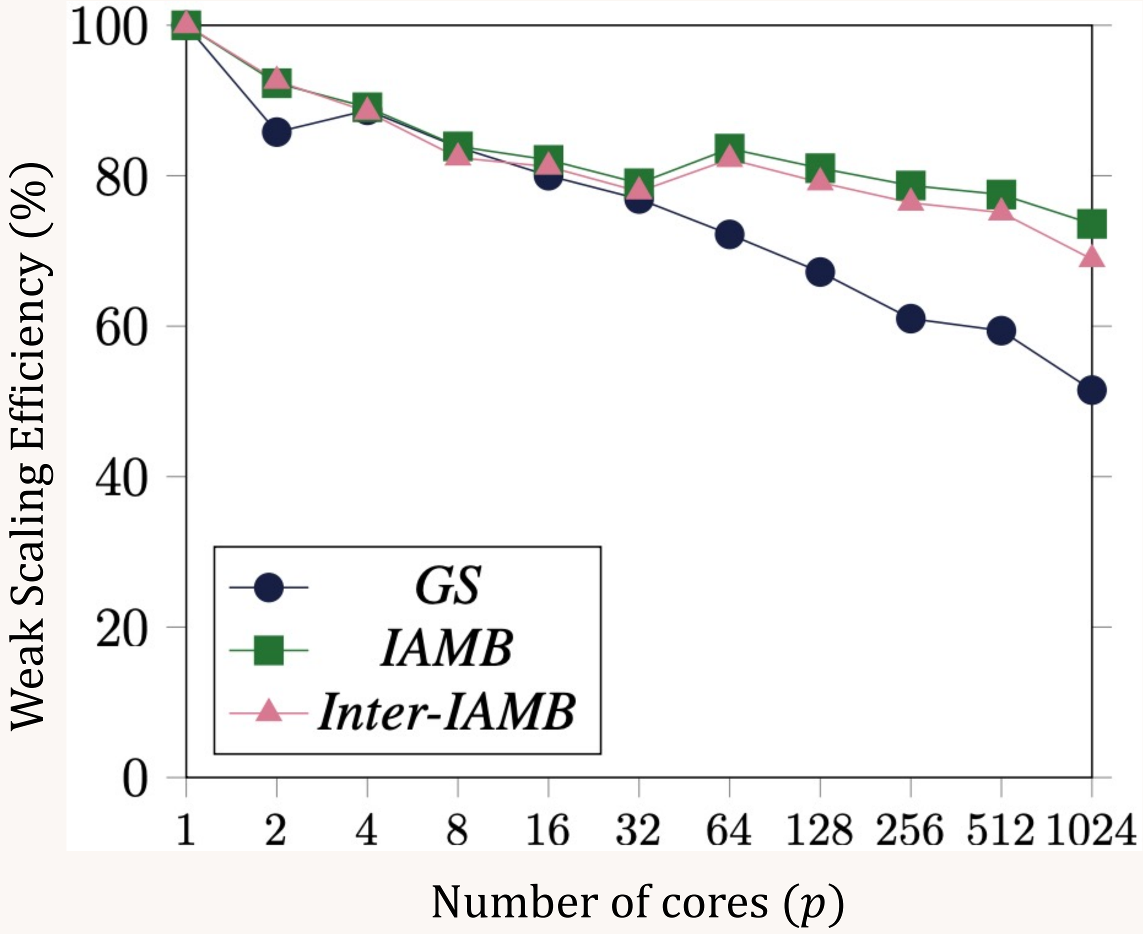


Experiments and Results

- Weak scaling of our framework
- Fixed work per processor – how do we vary n with increasing p?
  - Choose all the variables when using the largest p, a subset of variables for smaller p
- Estimated work per processor = n^2/p
  - Chosen number of variables scale as $\sqrt{p}$, i.e., $n_p = n\sqrt{p/p_{max}}$
  - We chose the first n_p variables in the data sets for our experiments

Experiments and Results

- Weak scaling of our framework – *D2*



Experiments and Results

- Our parallel algorithms learn genome-scale BNs in < 1 minute on 1024 cores, down from more than 13 hours sequentially
  - 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 < 2 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

Thanks! Questions?