Streaming Graph Challenge: Stochastic Block Partition
- draft -

Steven Smith, Edward Kao, Jeremy Kepner, Michael Hurley, Sanjeev Mohindra

LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

http://GraphChallenge.org

Outline

- Introduction
- Data Sets
- Graph Partitioning Challenge
- Metrics
- Summary

Introduction

- Previous challenges in graph exploitation, machine learning, High Performance Computing and visual analytics include
  - DIMACS (10^{th}), YOHO, MNIST, HPC Challenge, ImageNet, VAST
- Graph Challenge encourages community approaches, such as DARPA HIVE, to develop new solutions for analyzing graphs derived from social media, sensor feeds, and scientific data to enable relationships between events to be discovered as they unfold in the field
- Graph Challenge organizers will provide specifications, data sets, data generators, and serial implementations in various languages
- Graph Challenge participants are encouraged to apply innovative hardware, software, and algorithm techniques to push the envelop of power efficiency, computational efficiency, and scale
- Submissions will be in the form of full conference write ups to IEEE HPEC which will allow participants to be evaluated on their complete solution

Graph Challenge

- Graph Challenge seeks input from diverse communities to develop graph challenges that take the best of what has been learned from groundbreaking efforts such as GraphAnalysis, Graph500, FireHose, MiniTri, and GraphBLAS to create a new set of challenges to move the community forward
- Initial Graph Challenges
  - Static Graph Challenge: Sub-Graph Isomorphism

    This challenge seeks to identify a given sub-graph in a larger graph
  - Streaming Graph Challenge: Stochastic Block Partition

    This challenge seeks to identify optimal blocks (or clusters) within a graph

Static versus Streaming Mode

- Static graph processing
  – Given a large graph G
  – Evaluate $f(G)$
- Two classes of streaming: stateless and stateful
  – Stateless: process data g as it goes by $f(g)$
  – Stateful: add data g to a larger corpus G and then process corpus $f(G + g)$
- Graph processing is often in the stateful category
  – Easy to simulate by partitioning G into streaming pieces g

Labels and Filtering

- Filtering on edge/vertex labels is often used when available to reduce the search space
  - Filtering can be applied at initialization, during intermediate steps, or at the vertex level
  - Very problem dependent
- Some Graph Challenge data sets have labels and some data sets are unlabeled
  - Some participants will want to filter on labels
- Initial example implementations will work without labels
  - Labels can be used if they choose
- Graph Challenge is judged by a panel
  - Panel will value the variations that are submitted

Outline

- Introduction
- Data Sets
- Graph Partitioning Challenge
- Metrics
- Summary

Publicly Available Datasets

Initial Datasets in Blue

Stanford Large Network Dataset Collection snap.stanford.edu/data

- Social networks (10 data sets up to 4.8M vertices & 69M edges)
  - Online social networks, edges represent interactions between people
- Networks with ground-truth (6 data sets up to 66M vertices & 1.8B edges, e.g. Friendster)
  - Ground-truth network communities in social and information networks
- Communication networks (3 data sets up to 2.3M vertices & 5M edges)
  - Email communication networks with edges representing communication
- Citation networks (3 data sets up to 3.7M vertices & 16M edges)
  - Nodes represent papers, edges represent citations
- Collaboration networks (5 data sets up to 23K vertices & 198K edges)
  - Nodes represent scientists, edges represent collaborations
- Web graphs (4 data sets up to 875K vertices & 5.1M edges)
  - Nodes represent webpages and edges are hyperlinks
- Amazon networks (5 data sets up to 548K vertices & 3.4M edges)
  - Nodes represent products and edges link commonly co-purchased products
- Internet networks (9 data sets up to 62K vertices & 147K edges)
  - Nodes represent computers and edges communication
- Road networks (3 data sets up to 1.9M vertices & 2.8M edges)
  - Nodes represent intersections and edges roads connecting the intersections
- Autonomous systems (5 data sets up to 26K vertices & 106K edges)
  - Graphs of the internet
- Signed networks (10 data sets up to 4.8M vertices & 69M edges)
  - Networks with positive and negative edges (friend/foe, trust/distrust)
- Location-based online social networks (2 data sets up to 198K vertices & 950K edges)
  - Social networks with geographic check-ins
- Wikipedia networks, articles, and metadata (7 data sets up to 3.5M vertices & 250M edges)
  - Talk, editing, voting, and article data from Wikipedia
- Twitter and Memetracker (4 data sets up to 96M vertices & 476M edges)
  - Memetracker phrases, links and Tweets
- Online communities (3 data sets up to 2.3M images)
  - Data from online communities such as Reddit and Flickr
- Online reviews (6 data sets up to 34M product reviews)
  - Data from online review systems such as BeerAdvocate and Amazon

Publicly Available Datasets

Initial Datasets in Blue

AWS Public Data Sets aws.amazon.com/public-data-sets

- Astronomy (1 data set 180 GB)
  - Sloan Digital Sky Survey SQL MDF files
- Biology (4 data sets up to 200 TB)
  - Genome sequence data
- Climate (3 data sets size growing daily)
  - Satellite imagery data
- Economics (10 data sets up to 220 GB)
  - Census, transaction, and transportation data
- Encyclopedic (10 data sets up to 541 TB - Common Crawl Corpus)
  - Various online encyclopedia data
- Geographic (3 data sets up to 125 GB)
  - Street maps
- Mathematics (1 data sets 160 GB)
  - University of Florida Sparse Matrix Collection

Graph500.org Data Generator

- Used to generate world's largest power law graphs
- Can be modeled with Kronecker Product of a Recursive MATrix (R-MAT): $G^{\otimes k} = G^{\otimes k-1} \otimes G$
  - Where "⊗" denotes the Kronecker product of two matrices

Yahoo! Webscope Datasets webscope.sandbox.yahoo.com

- Advertising and Market Data (4 data sets up to 3.7 GB)
  - Yahoo!'s auction-based platform for selling advertising space
- Competition Data (3 data sets up to 1.5 GB)
  - Data challenges run by Yahoo
- Computing Systems Data (5 data sets up to 8.8 GB)
  - Computer systems log data
- Graph and Social Data (3 data sets up to 5 GB)
  - Graph data from search, groups, and webpages
- Image Data (3 data sets up to 14 GB)
  - Flickr imagery and metadata
- Language Data (29 data sets up to 166 GB - Answers browsing behavior)
  - Wide range of question/answer data sets
- Ratings and Classification Data (10 data sets up to 83 GB — 1.5 TB dataset withdrawn)
  - Community preferences and data

Tools to generate (at various scales and parameters) data sets will be provided as part of the challenge

- R-MAT: A Recursive Model for Graph Mining, Chakrabarti, Zhan & Faloutsos (2004)
- Mathematically Tractable Graph Generation and Evolution, Leskovec, Chakrabarti, Kleinberg & Faloutsos, (2005)

Graph Challenge Data Formats

Wikipedia Voting

Wikipedia Adminship

Road Network

Twitter Data

- **Public data is available in a variety of formats**
  - Linked list, tab separated, labeled/unlabeled
- **Requires parsing and standardization**
- **Proposed formats**
  - Tab separated triples in ASCII file with labels removed
  - MMIO ASCII format: math.nist.gov/MatrixMarket

Amazon cloud services will be used to host particularly large data sets

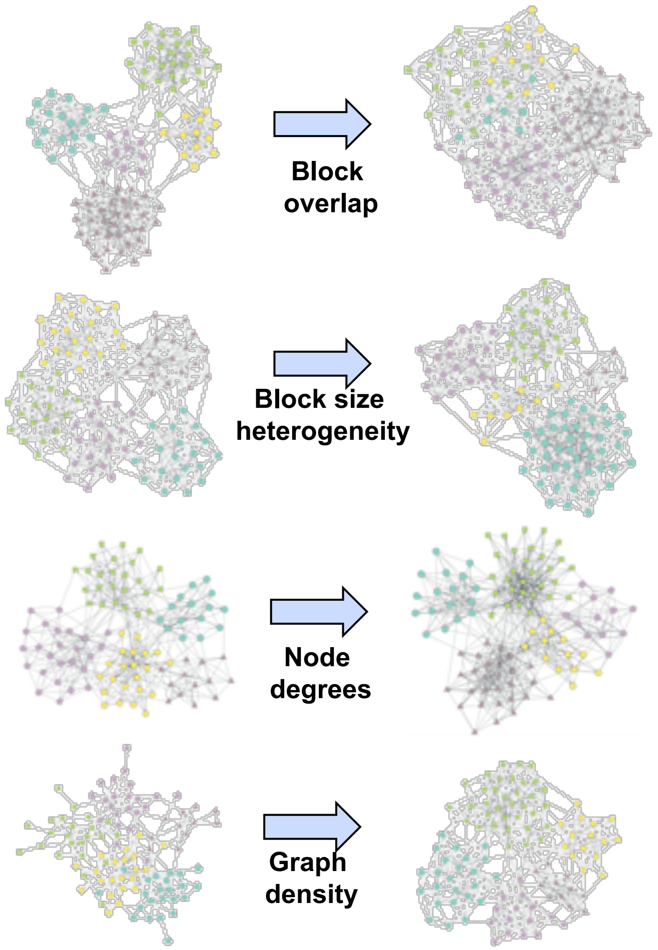
The Matrix Market Exchange Formats: Initial Design, Boisvert, Pozo & Remington (1996)

Data Set and Truth Generation for Graph Partitioning Challenge

- **Generate simulated graphs with truth on the block structure (i.e. graph partition)**
  - **Realism captured by statistical models:**

    Degree corrected blockmodel, power law degree distribution, Dirichlet-multinomial block assignment
  - **Vary parameters to capture diversity and levels of difficulty**

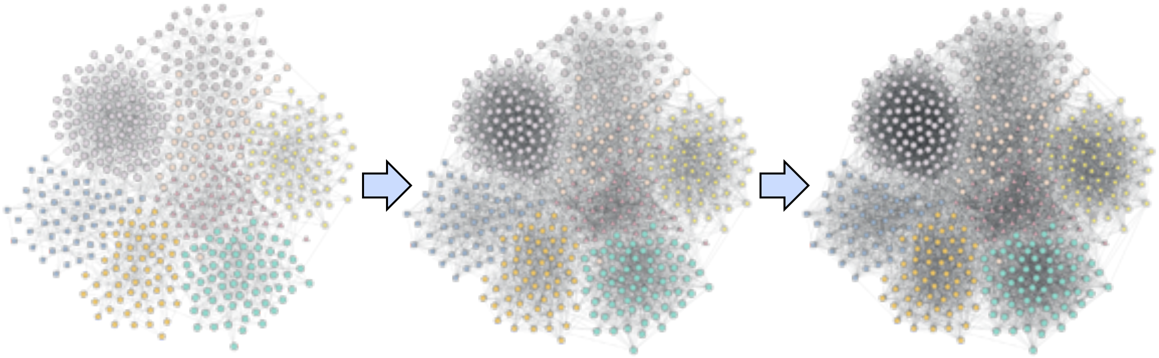
    Block overlap, block size, node degree, graph size, density, and more
- **Real world graphs (SNAP library)**
  - **However most real world graphs do not have truth**
- **Embed generated graphs into real world graphs**
  - **Partition correctness is evaluated on the generated part with truth**



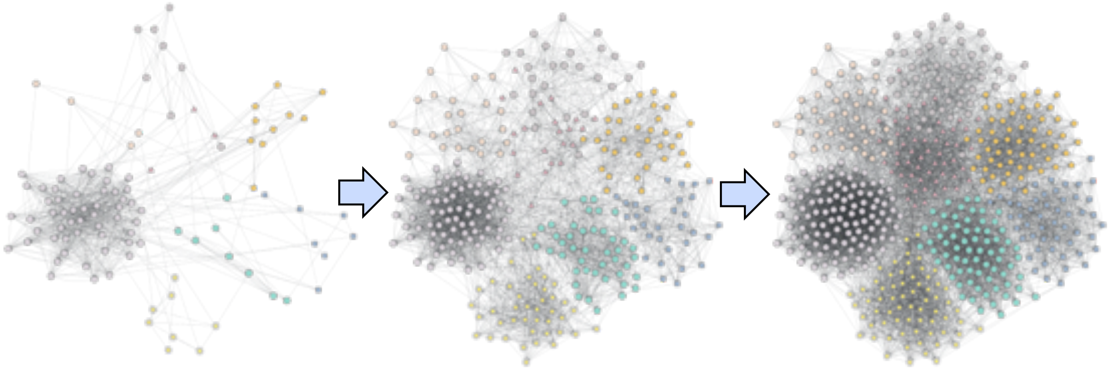
Streaming Graph Generation and Processing

- In real-world applications, graph data often arrives in streaming fashion
- Streaming graph data sets are generated in two ways:
  - Emerging edge samples: edges become available and are observed over stages
  - Snowball sampling: the graph is explored through snowball sampling from entry node(s)
- Partition is done at each stage of the streaming graph
  - Algorithm moves onto the next stage when it has partitioned the previous stage
  - Algorithm that leverages partition(s) from the previous stage(s) is encouraged
  - Performance metrics should be reported at each stage

Streaming with Emerging Edge Samples



Streaming with Snowball Sampling

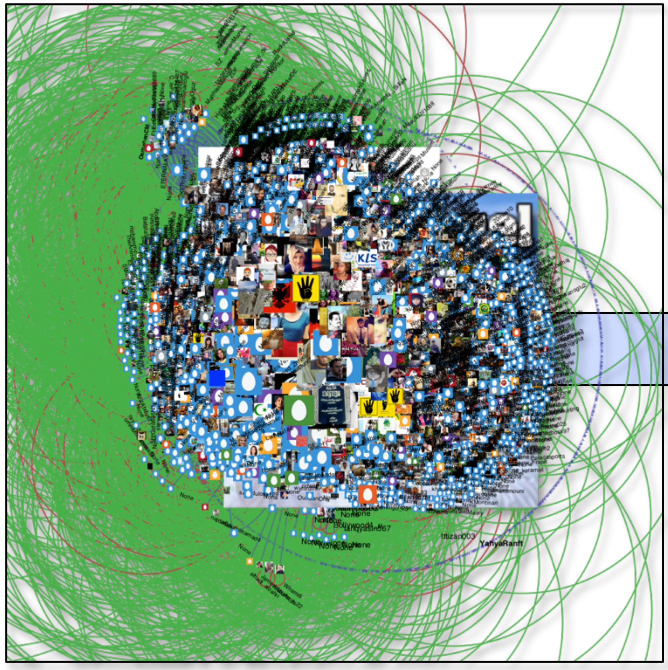


Outline

The Graph Partitioning Problem

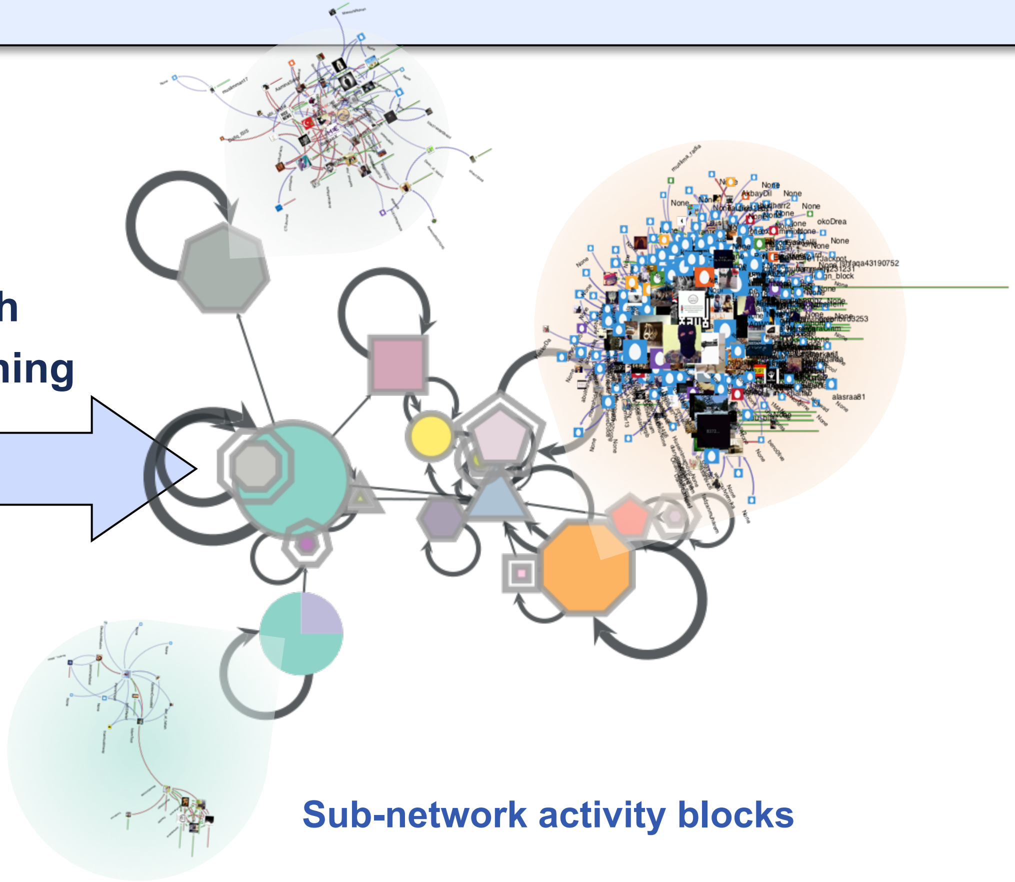
Problem: Discover community structure in graph topology and interaction data

Application: Identify significant activities within large graphs



Social Media Network

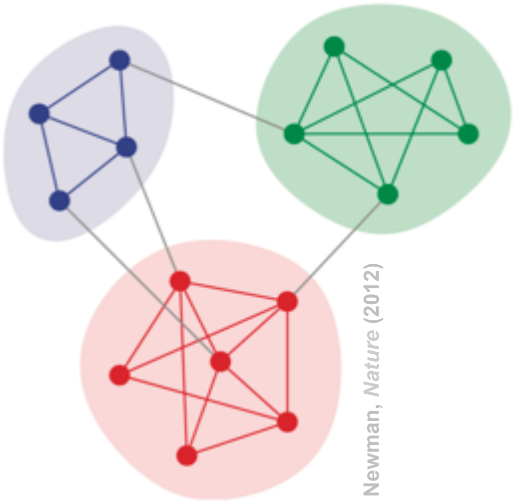
Graph Partitioning



Sub-network activity blocks

Blockmodel Graph Partitioning

- Blockmodels capture the community structure of graphs:
  - Block membership for each node
  - Block matrix describes interaction between blocks
  - Typically, nodes in the same block interact more than those between different blocks
- Rigorous statistical inference on blockmodels is a principled way to partition graphs
  - Inferred block membership on each node provides the partition
  - Model selection determines the optimal number of blocks (i.e. partition resolution)
  - Bayesian methods capture uncertainty of the partition
  - Produces better partition than spectral and modularity maximization methods

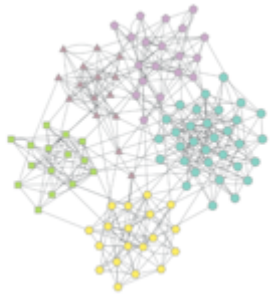


Newman, *Nature* (2012)

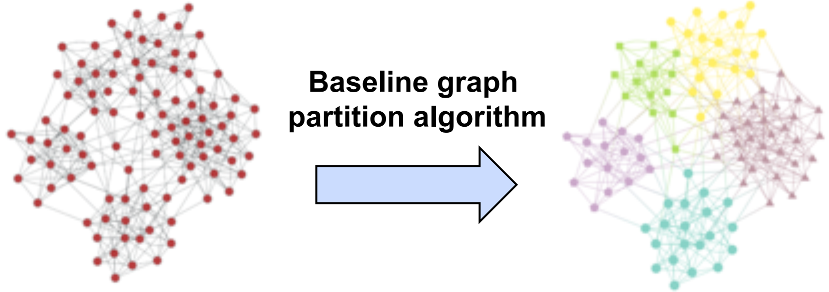
Blockmodel Partitioning Baseline Algorithm

- **Based on degree corrected stochastic blockmodel (Karrer and Newman 2010)**
  - **Realistic model obtained by varying degrees across vertices**
  - **Likelihood a function of edge counts between blocks (simple, parallel)**
- **Returns partition with shortest graph description length (Peixoto 2012, 2013)**
- **Markov chain Monte Carlo (MCMC) inference approach amenable to parallelism**
- **Challenging computational complexity for large graphs**
- **Participants may enter with a different partition algorithm**
  - **All entries should report performance using the Challenge metrics and data sets**
  - **The true number of blocks is NOT given to the algorithm**

Truth graph (reference)



Input graph **Partitioned graph**



Degree Corrected Blockmodel

Network model with Poisson Interactions:

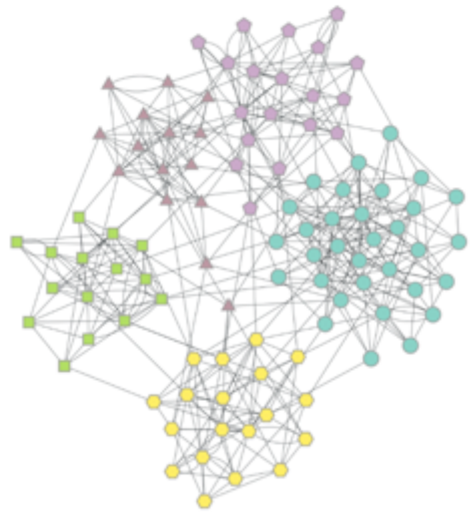
Edge interaction from *i* to *j* ~ Poisson(λ_{ij})

$$\text{Rate } \lambda_{ij} = \boxed{d_i d_j} \times \boxed{\text{block interaction rate}(b_i, b_j)}$$

Node degrees

Stochastic blockmodel

Block membership

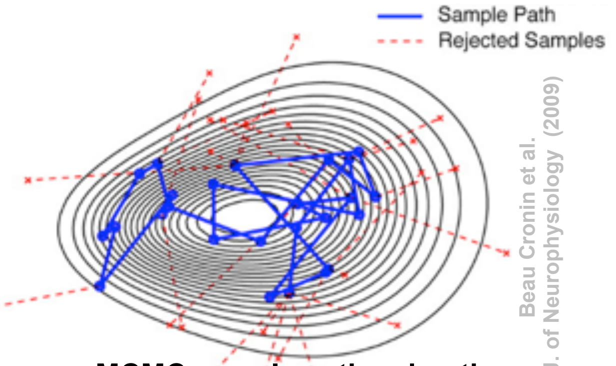


- Realistic degree distribution accounts for features like "six degrees of Kevin Bacon" (aka small world) and power-law scaling
- Realistic community structure captured by stochastic blockmodel
- A combination of two well-known random graph models

- Stochastic blockmodels and community structure in networks, Karrer and Newman (2011)
- A random graph model for power law graphs, Aiello, Chung, and Lu (2001)
- Estimation and prediction for stochastic blockmodels …, Snijders and Nowicki (1997)

Markov Chain Monte Carlo (MCMC)

- Inference on realistic models often results in complex distributions with no closed-form representation
  - This presents a challenge because such distributions represent the quantity of interest
- Markov Chain Monte Carlo (MCMC) samples the target distribution
  - New samples are proposed from the previous samples
  - Upon convergence, the samples represent exactly the target distribution
- MCMC on complex multivariate distributions is performed by updating one variable at a time (i.e. Gibbs sampling)



MCMC sample path using the Metropolis-Hasting algorithm

Parallel MCMC for Blockmodel Graph Partitioning

- **The distribution on the graph partition does not have a closed form and has high dimension (#vertices)**

$$p(b|G) \propto \sum_{ij} M_{ij} \log\left(\frac{M_{ij}}{d_i d_j}\right)$$

edges between block *i* and *j* according to partition *b* on graph *G*

edges in block *i and j*

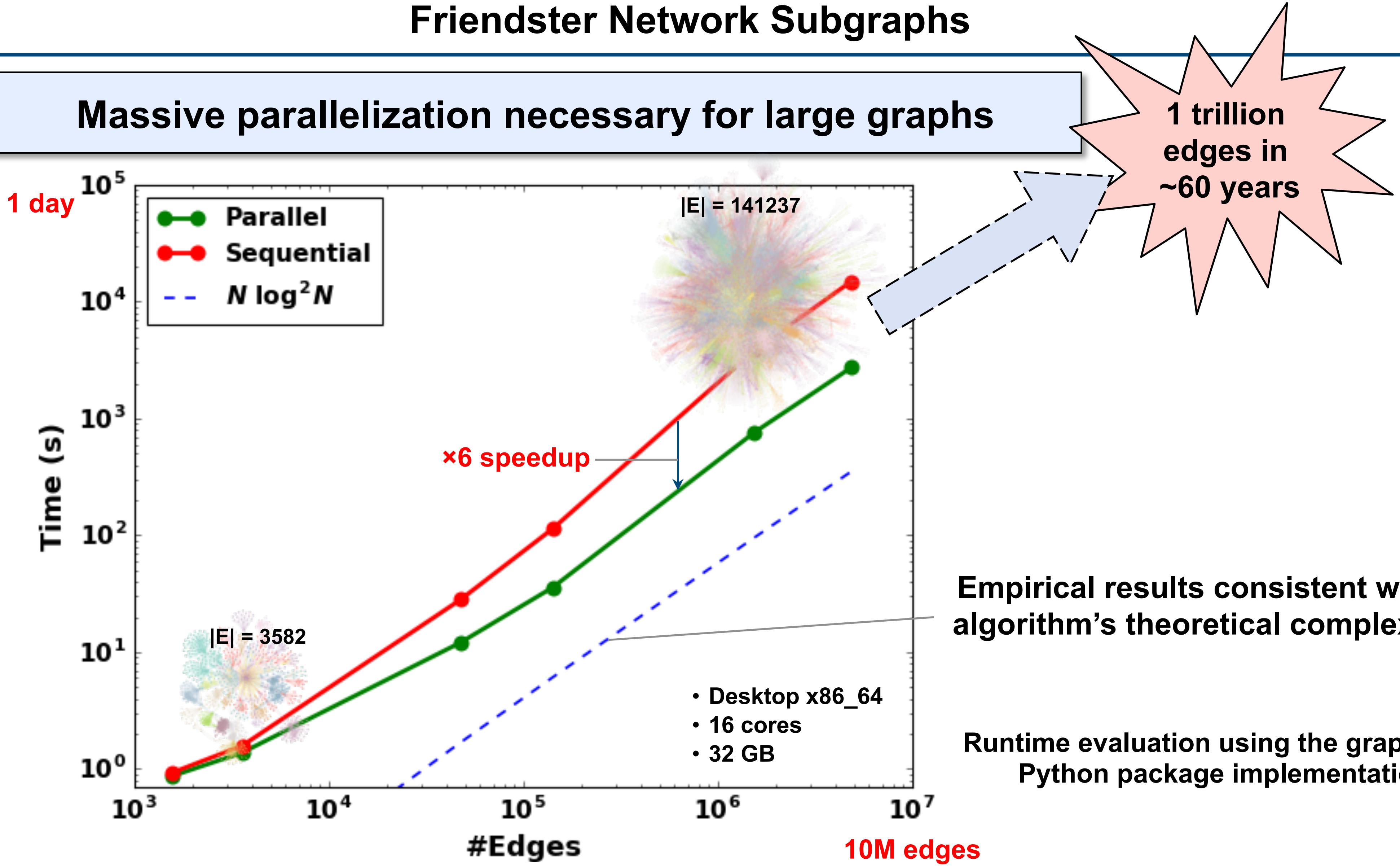
  - **MCMC sampling is ideal for computing this distribution**
  - **Efficient partition updates in parallel, one node at a time**

- **Updating each node can be massively parallelized**
  - **Sequential update gives exact graph partition distribution**
  - **Parallel update gives a reasonable approximation and produces comparable result in our evaluation over simulated and real graphs**

Processing Time vs Graph Size

Friendster Network Subgraphs

Massive parallelization necessary for large graphs



Empirical results consistent with algorithm's theoretical complexity

Runtime evaluation using the graph-tool Python package implementation

LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Parallel MCMC Partitioning Closely Approximates "Exact" Sequential MCMC

- Partitions on a range of graphs sizes show differences within the random variability of the partitioning algorithm itself
- Quantitative metrics used for comparing partitions (correctness metrics)
- Recent theoretical development in parallel MCMC suggests good performance under sparse correlation (De Sa 2016)
  - Nodal block assignments correlate sparsely since graphs are typically sparse

Sequential, O(10^3) vertices

Parallel, O(10^3) vertices

Sequential, O(10^4) vertices

Parallel, O(10^4) vertices

Baseline Algorithmic Description Documents for Performers

Vertex-Based Parallelism

Algorithm 1: Block Assignment Update At Each Node i

input : b_i^-, $\boldsymbol{b}_{\mathcal{N}_i}^-$: current block labels for node i and its neighbors $\mathcal{N}_i$
$\boldsymbol{M}^-$: current $B \times B$ inter-block edge count matrix
$\boldsymbol{A}_{i\mathcal{N}_i}, \boldsymbol{A}_{\mathcal{N}_i i}$: edges between i and all its neighbors
output: b_i^+: the new block assignment for node i

`// propose a block assignment`
obtain the current block assignment $r = b_i^-$
draw a random edge of i which connects with a neighbor j, obtain its block assignment $u = b_j^-$
draw a uniform random variable $x_1 \sim \text{Uniform}(0, 1)$
if $x_1 \leqslant \frac{B}{d_u^- + B}$ **then**
`// with some probability, propose randomly for exploration`
propose $b_i^+ = s$ by drawing s randomly from $\{1, 2, ..., B\}$
else
`// otherwise, propose by multinomial draw from neighboring blocks to` u
propose $b_i^+ = s$ from $\text{MultinomialDraw}\left(\frac{M_{u:}^- + M_{:u}^-}{d_u^-}\right)$
end
`// accept or reject the proposals`
if $s = r$ **then**
return $b_i^+ = b_i^-$ `// proposal is the same as the old assignment. done!`
else
compute $\boldsymbol{M}^+$ under proposal (update only rows and cols r and s, on entries for blocks connected to i)
compute proposal probabilities for the Hastings correction:
$$p_{r \to s} = \sum_{t \in \{\boldsymbol{b}_{\mathcal{N}_i}^-\}} \left[K_{it} \frac{M_{ts}^- + M_{st}^- + 1}{d_t^- + B}\right] \quad \text{and} \quad p_{s \to r} = \sum_{t \in \{\boldsymbol{b}_{\mathcal{N}_i}^-\}} \left[K_{it} \frac{M_{tr}^+ + M_{rt}^+ + 1}{d_t^+ + B}\right]$$
compute change in log posterior (t_1 and t_2 only need to cover rows and cols r and s):
$$\Delta S = \sum_{t_1, t_2} \left[-M_{t_1 t_2}^+ \log\left(\frac{M_{t_1 t_2}^+}{d_{t_1,\text{out}}^+ d_{t_2,\text{in}}^+}\right) + M_{t_1 t_2}^- \log\left(\frac{M_{t_1 t_2}^-}{d_{t_1,\text{out}}^- d_{t_2,\text{in}}^-}\right)\right]$$
compute probability of acceptance:
$$p_{\text{accept}} = \min\left[\exp(-\beta \Delta S) \frac{p_{s \to r}}{p_{r \to s}}, 1\right]$$
draw a uniform random variable $x_3 \sim \text{Uniform}(0, 1)$
if $x_3 \leqslant p_{\text{accept}}$ **then**
return $b_i^+ = s$ `// accept the proposal`
else
return $b_i^+ = r$ `// reject the proposal`
end
end

Array-Based (Batch) Parallelism

Algorithm 2: Batch Assignment Update for All Nodes

input : $\boldsymbol{\Gamma}^-$: current block assignment matrix for all nodes
$\boldsymbol{M}^-$: current $B \times B$ inter-block edge count matrix
$\boldsymbol{A}$: graph adjacency matrix
output: $\boldsymbol{\Gamma}^+$: new block assignments for all nodes

`// propose new block assignments`
compute node degrees: $\boldsymbol{k} = (\boldsymbol{A} + \boldsymbol{A}^T)\boldsymbol{1}$
compute block degrees: $\boldsymbol{d}_{\text{out}}^- = \boldsymbol{M}^- \boldsymbol{1}$; $\boldsymbol{d}_{\text{in}}^- = \boldsymbol{M}^{-^T} \boldsymbol{1}$; $\boldsymbol{d}^- = \boldsymbol{d}_{\text{out}}^- + \boldsymbol{d}_{\text{in}}^-$
compute probability for drawing each neighbor: $\boldsymbol{P}_{\text{Nbr}} = \text{RowDivide}(\boldsymbol{A} + \boldsymbol{A}^T, \boldsymbol{k})$
draw neighbors ($\boldsymbol{N}_{\text{br}}$ is a binary selection matrix): $\boldsymbol{N}_{\text{br}} = \text{MultinomialDraw}(\boldsymbol{P}_{rn})$
compute probability of uniform random proposal: $\boldsymbol{p}_{\text{UnifProp}} = \frac{B}{\boldsymbol{N}_{\text{br}} \boldsymbol{\Gamma}^- \boldsymbol{d}^- + B}$
compute probability of block transition: $\boldsymbol{P}_{\text{BlkTran}} = \text{RowDivide}(\boldsymbol{M}^- + \boldsymbol{M}^{-^T}, \boldsymbol{d}^-)$
compute probability of block transition proposal: $\boldsymbol{P}_{\text{BlkProp}} = \boldsymbol{N}_{br} \boldsymbol{\Gamma}^- \boldsymbol{P}_{\text{BlkTran}}$
propose new assignments uniformly: $\boldsymbol{\Gamma}_{\text{Unif}} = \text{UniformDraw}(B, N)$
propose new assignments from neighborhood: $\boldsymbol{\Gamma}_{\text{Nbr}} = \text{MultinomialDraw}(\boldsymbol{P}_{\text{BlkProp}})$
draw N Uniform$(0, 1)$ random variables $\boldsymbol{x}$
compute which proposal to use for each node: $\boldsymbol{I}_{\text{UnifProp}} = \boldsymbol{x} \leqslant \boldsymbol{p}_{\text{UnifProp}}$
select block assignment proposal for each node:
$$\boldsymbol{\Gamma}^P = \text{RowMultiply}(\boldsymbol{\Gamma}_{\text{Unif}}, \boldsymbol{I}_{\text{UnifProp}}) + \text{RowMultiply}(\boldsymbol{\Gamma}_{\text{Nbr}}, (\boldsymbol{1} - \boldsymbol{I}_{\text{UnifProp}}))$$
`// accept or reject the proposals`
compute change in edge counts by row and col: $\Delta \boldsymbol{M}_{\text{row}}^+ = \boldsymbol{A}\boldsymbol{\Gamma}^-$; $\Delta \boldsymbol{M}_{\text{col}}^+ = \boldsymbol{A}^T \boldsymbol{\Gamma}^-$
update edge count matrix for each proposal: (resulting matrix is $N \times P \times P$):
$$\mathcal{M}_{ijk}^+ = M_{jk}^- - \Gamma_{ij}^- \Delta M_{\text{row},ik}^+ + \Gamma_{ij}^P \Delta M_{\text{row},ik}^+ - \Gamma_{ik}^- \Delta M_{\text{col},ij}^+ + \Gamma_{ik}^P \Delta M_{\text{col},ij}^+$$
update block degrees for each proposal: (resulting matrix is $N \times P$):
$$D_{\text{out},ij}^+ = d_{\text{out},j}^- - \Gamma_{ij}^- \sum_k \Delta M_{\text{row},ik}^+ + \Gamma_{ij}^P \sum_k \Delta M_{\text{row},ik}^+$$
$$D_{\text{in},ij}^+ = d_{\text{in},j}^- - \Gamma_{ij}^- \sum_k \Delta M_{\text{col},ik}^+ + \Gamma_{ij}^P \sum_k \Delta M_{\text{col},ik}^+$$
compute the proposal probabilities for Hastings correction ($N \times 1$ vectors):
$$\boldsymbol{p}_{r \to s} = \left[(\boldsymbol{p}_{\text{Nbr}} \boldsymbol{\Gamma}^-) \circ (\boldsymbol{\Gamma}^P \boldsymbol{M}^- + \boldsymbol{\Gamma}^P \boldsymbol{M}^{-^T} + 1) \circ \text{RepMat}\left(\frac{1}{\boldsymbol{d}^- + B}, N\right)\right] \boldsymbol{1}$$
$$\boldsymbol{p}_{s \to r,i} = \left[(\boldsymbol{p}_{\text{Nbr}} \boldsymbol{\Gamma}^-) \circ (\boldsymbol{\Gamma}^- \boldsymbol{\mathcal{M}}_{i-}^+ + \boldsymbol{\Gamma}^- \boldsymbol{\mathcal{M}}_{i-}^{+^T} + 1) \circ \frac{1}{\boldsymbol{D}_{\text{out}}^+ + \boldsymbol{D}_{\text{in}}^+ + B}\right] \boldsymbol{1}$$
compute change in log posterior (only need to operate on the impacted rows and columns corresponding to r, s, and the neighboring blocks to i):
$$\Delta S_i = \sum_{jk} \left[-\mathcal{M}_{ijk}^+ \log\left(\frac{\mathcal{M}_{ijk}^+}{D_{\text{out},ij}^+ + D_{\text{in},ik}^+}\right) + M_{jk}^- \log\left(\frac{M_{jk}^-}{d_{\text{out},j}^- + d_{\text{in},k}^-}\right)\right]$$
compute probabilities of accepting the proposal ($N \times 1$ vector):
$$\boldsymbol{p}_{\text{Accept}} = \min\left[\exp(-\beta \Delta \boldsymbol{S}) \circ \boldsymbol{p}_{s \to r} \circ \frac{1}{\boldsymbol{p}_{r \to s}}, \boldsymbol{1}\right]$$
draw N Uniform$(0, 1)$ random variable $\boldsymbol{x}_{\text{Accept}}$
compute which proposals to accept: $\boldsymbol{I}_{\text{Accept}} = \boldsymbol{x}_{\text{Accept}} \leqslant \boldsymbol{p}_{\text{Accept}}$
return $\boldsymbol{\Gamma}^+ = \text{RowMultiply}(\boldsymbol{\Gamma}^P, \boldsymbol{I}_{\text{Accept}}) + \text{RowMultiply}(\boldsymbol{\Gamma}^-, (\boldsymbol{1} - \boldsymbol{I}_{\text{Accept}}))$

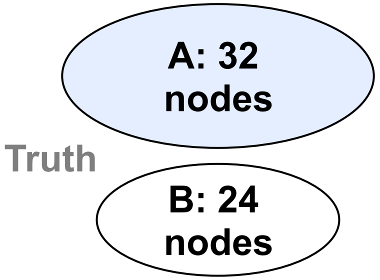
Outline

- Introduction
- Data Sets
- Graph Partitioning Challenge
- Metrics
- Summary

Graph Partition Performance Metrics

- Correctness metrics
  - Objective comparison between performer results with common datasets
  - Diagnostic performance tools for performer development
  - Baseline recommendation: Pairwise Precision-Recall
- Computational metrics
  - Total number of edges in graph partitioned
  - Execution time
  - Rate (edges/second)
  - Energy (Watts)
  - Rate per energy (edges/second/Watt)
  - Memory and processor requirement (amount and type used)
- For streaming graphs, evaluation should be done at each stage using these metrics

Graph Partition Correctness Metrics



	Output A	Output B	Output C
Truth A	30	2	0
Truth B	1	20	3

Contingency Table

- **Accuracy = Correct/Total = 50/56 = 89%**
  - **Good: Intuitive / Bad: Ignores distribution of errors**
- **Precision-Recall (per block)**
  - **Precision(A) = Correct(A)/[Correct(A)+False(A)] = 30/31 [column]**
  - **Recall(A) = Correct(A)/[Correct(A)+Missed(A)] = 30/32 [row]**
  - **Good: Intuitive, per block diagnostics / Bad: No global summary**
- **Pairwise Precision-Recall**
- **Information theoretic P-R**

- Comparing partitions, Hubert (1985)
- Comparing clusterings—an information based distance, Meilă (2005)

Graph Partition Correctness Metrics (cont.)

Truth

A: 32 nodes

B: 24 nodes

	Output A	Output B	Output C
Truth A	30	2	0
Truth B	1	20	3

Contingency Table

- Accuracy
- Precision-Recall (per block)
- **Pairwise Precision-Recall**

Baseline recommendation

  – **Four possible outcomes for every pair of nodes**

    Same/Same, Different/Different, Same/Different, Different/Same

  – **Precision = SS/(SS+DS) ; Recall = SS/(SS+SD) [for all pairs]**

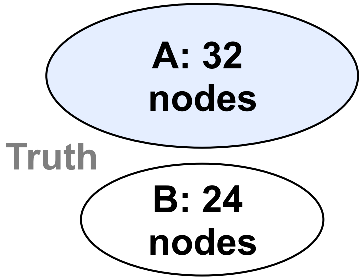
    P = 90% ; R = 81% for example

  – **Good: ROC-like summary statistic / Bad: Pairwise counting is ad hoc**

- Information theoretic P-R

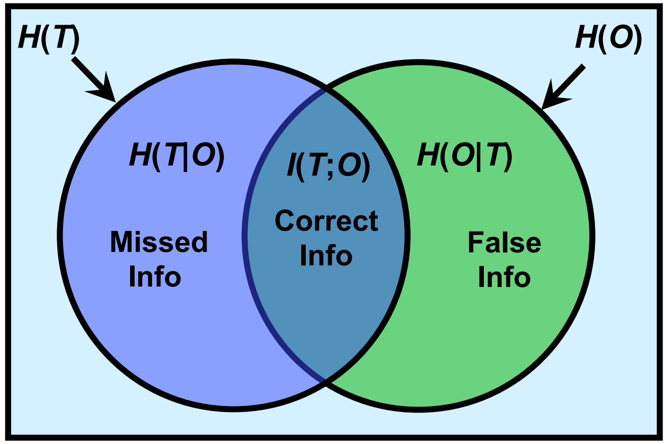
- Comparing partitions, Hubert (1985)
- Comparing clusterings—an information based distance, Meilă (2005)

Graph Partition Correctness Metrics (cont.)



	Output A	Output B	Output C
Truth A	30	2	0
Truth B	1	20	3

Contingency Table



- Accuracy
- Precision-Recall (per block)
- Pairwise Precision-Recall
- **Information theoretic P-R**
  - **Precision = I(T;O)/H(O) ; Recall = I(T;O)/H(T)**

    P = 57% ; R = 71% for example
  - **Good: ROC-like summary statistics, captures error distribution, rigorous interpretation**
  - **Bad: Entropy-space interpretation is non-intuitive**

- Comparing partitions, Hubert (1985)
- Comparing clusterings—an information based distance, Meilă (2005)

Summary

- Graph partitioning challenge on simulated graphs and embedded real-world graphs
  - Partitioning algorithm uses realistic stochastic blockmodel and statistically rigorous inference
  - Streaming versions defined
  - Correctness and computational performance metrics defined
- Performer implementations will compute the block partition on provided data sets and report the given metrics
- A submission to the IEEE HPEC Graph Challenge consists of a conference style paper describing the approach, implementation, innovations, and results
- Both hardware and software should be described in solution
  - Innovative hardware solutions are of interest (in addition to best algorithm for hardware)
  - Special interest in performance for large scale data sets (1 billion edges)