Graph Analytics - EDBT Summer School

Charalampos E. Tsourakakis
University of Cyprus
Nicosia

RelationalAI and Boston U.

July 8th, 2025

Thank You!

Special thanks to the organizers:

Prof. Demetris Zeinalipour

Prof. Panos K. Chrysanthis

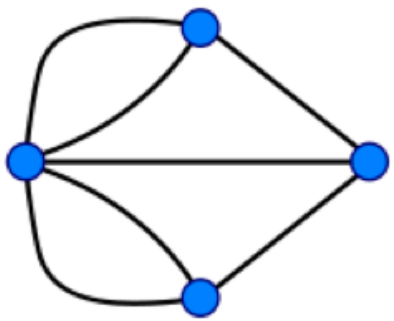
Graphs Analytics

- Algorithmic Techniques and Machine Learning in Graph Analytics
  - I will use established research results to illustrate fundamental theoretical concepts.
  - Goal: be instructive and interactive.

Note: If anything is unclear, just stop me and ask. Chances are others are wondering the same thing.

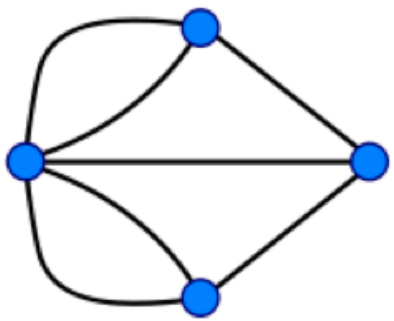
Graphs: a simple model

- entities – set of vertices
- pairwise relations among vertices – set of edges
- can add directions, weights,. . .
- graphs can be used to model many real datasets
  - people who are friends
  - computers that are interconnected
  - web pages that point to each other
  - proteins that interact



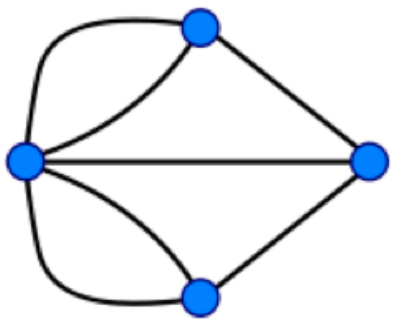
Graphs: a simple model

- entities – set of vertices
- pairwise relations among vertices – set of edges
- can add directions, weights,...
- graphs can be used to model many real datasets
  - people who are friends
  - computers that are interconnected
  - web pages that point to each other
  - proteins that interact



Graphs: a simple model

- entities – set of vertices
- pairwise relations among vertices
  – set of edges
- can add directions, weights,. . .
- graphs can be used to model many real datasets
  - people who are friends
  - computers that are interconnected
  - web pages that point to each other
  - proteins that interact

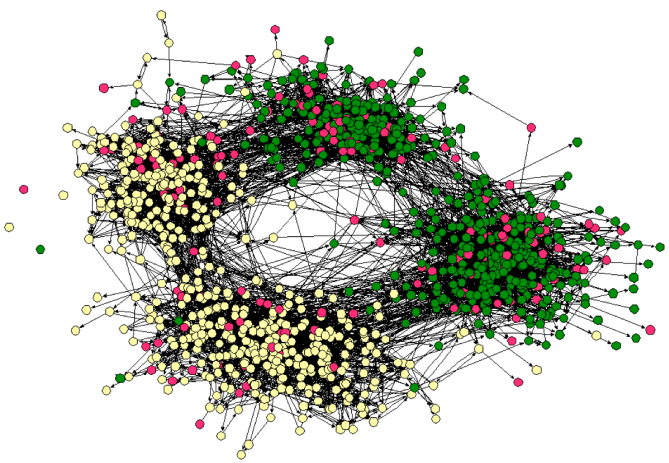


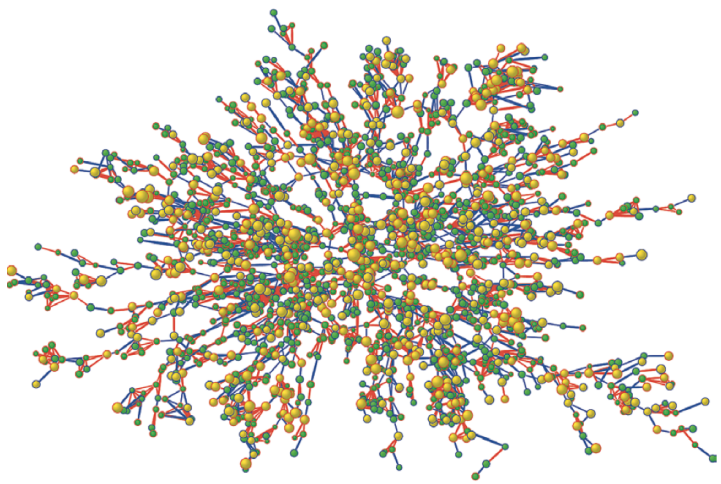
Graph theory

- graph theory started in the 18th century, with Leonhard Euler
  - the problem of Königsberg bridges: Is it possible to take a walk through the city that crosses each bridge exactly once and returns to the starting point?
  - He abstracted the land masses as nodes (vertices) and the bridges as edges, inventing what we now call a graph.
  - since then, graphs have been studied extensively

Analysis of Graph Datasets in the Past

- graphs datasets have been studied in the past
  e.g., networks of highways, social networks
  - usually these datasets were small
  - visual inspection can reveal a lot of information

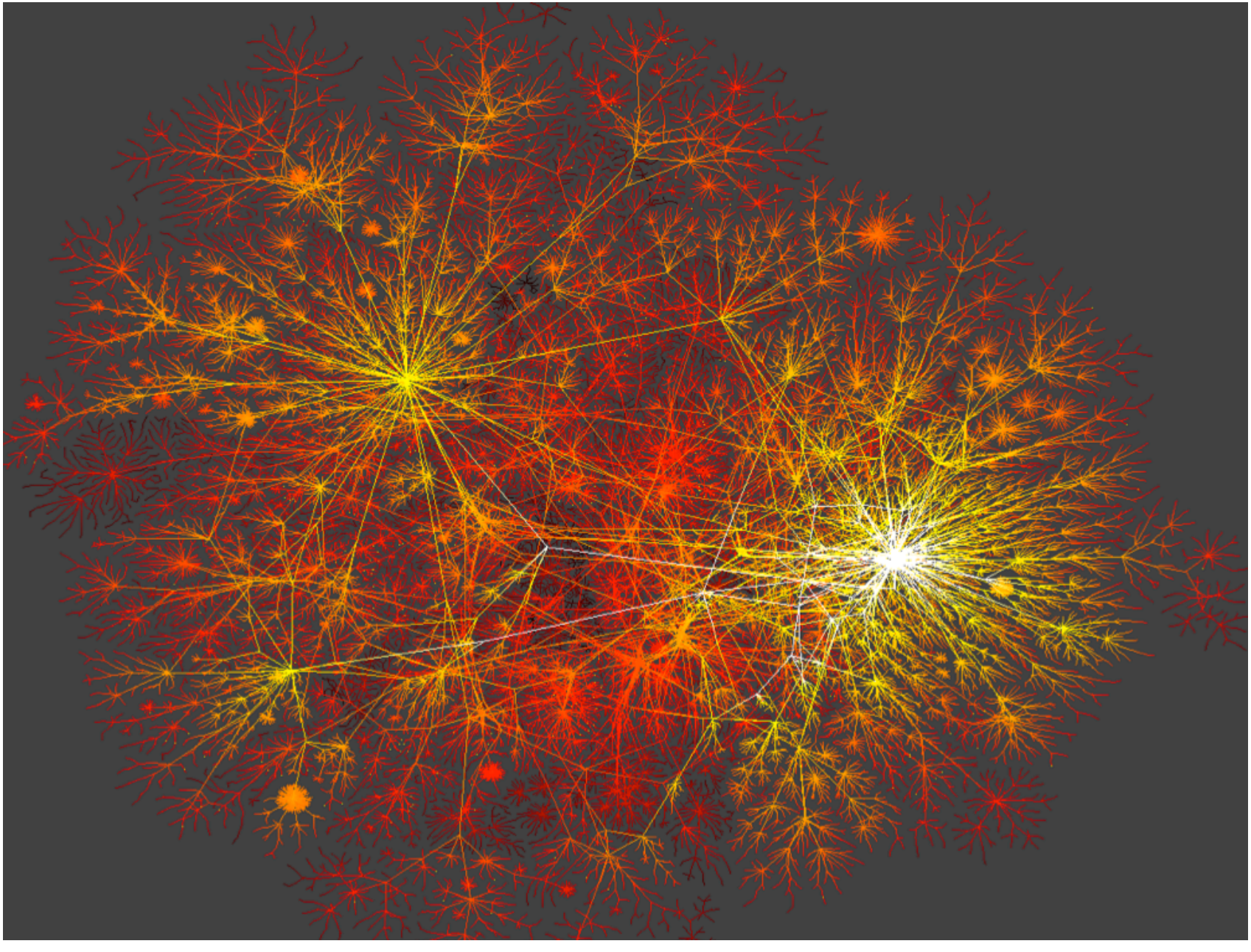




Analysis of Graph Datasets Now

- more and larger networks appear
  - products of technological advancement
    - e.g., internet, web
  - result of our ability to collect more, better-quality, and more complex data
    - e.g., gene regulatory networks
- networks of thousands, millions, or billions of nodes
  - impossible to visualize
- deep graph models power breakthroughs in drug design, recommendation, and fraud detection
  - growing interest in deep learning on graphs fuels need for scalable analytics

The Internet map

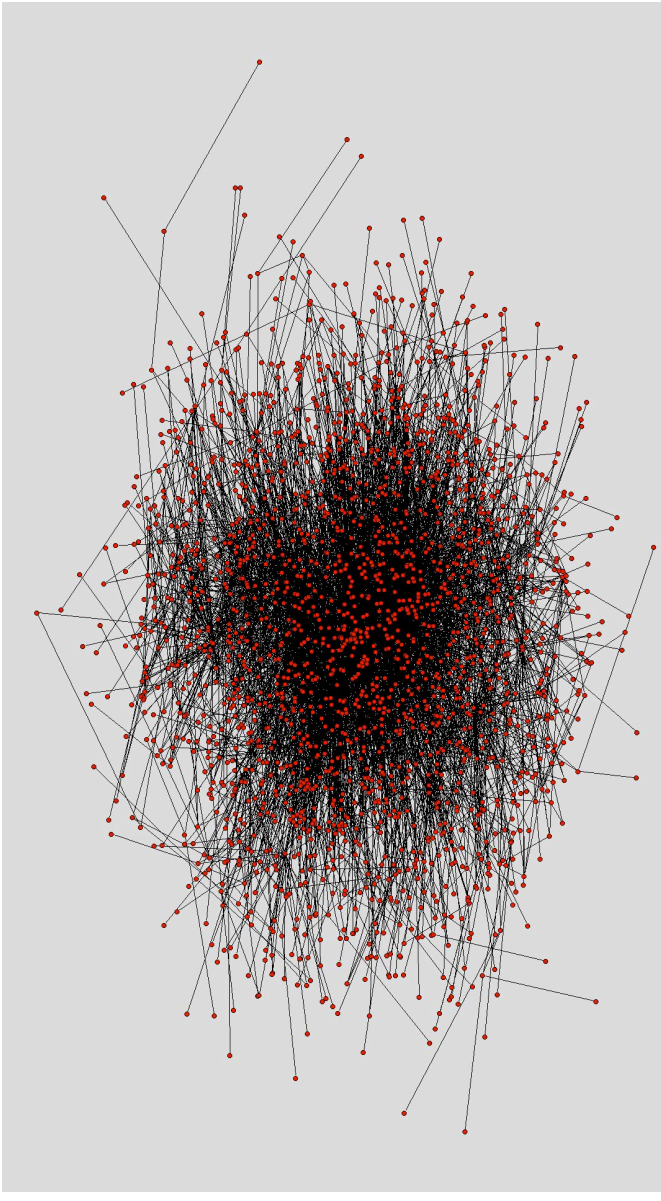


Types of Networks

- social networks
- knowledge and information networks
- technology networks
- biological networks
- neural networks
- knowledge graphs
- ...

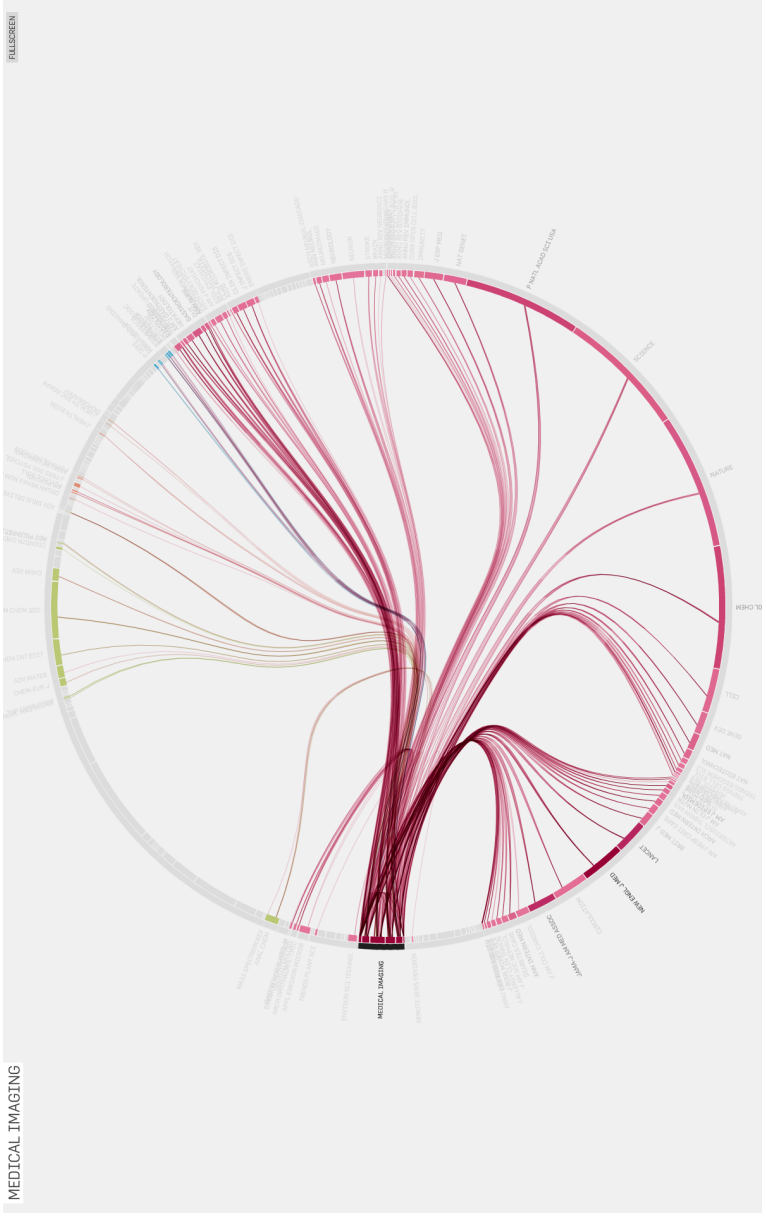
Social Networks

- links denote a social interaction
  - networks of acquaintances
  - collaboration networks
    - actor networks
    - co-authorship networks
    - director networks
  - phone-call networks
  - e-mail networks
  - IM networks
  - sexual networks



Knowledge and Information Networks

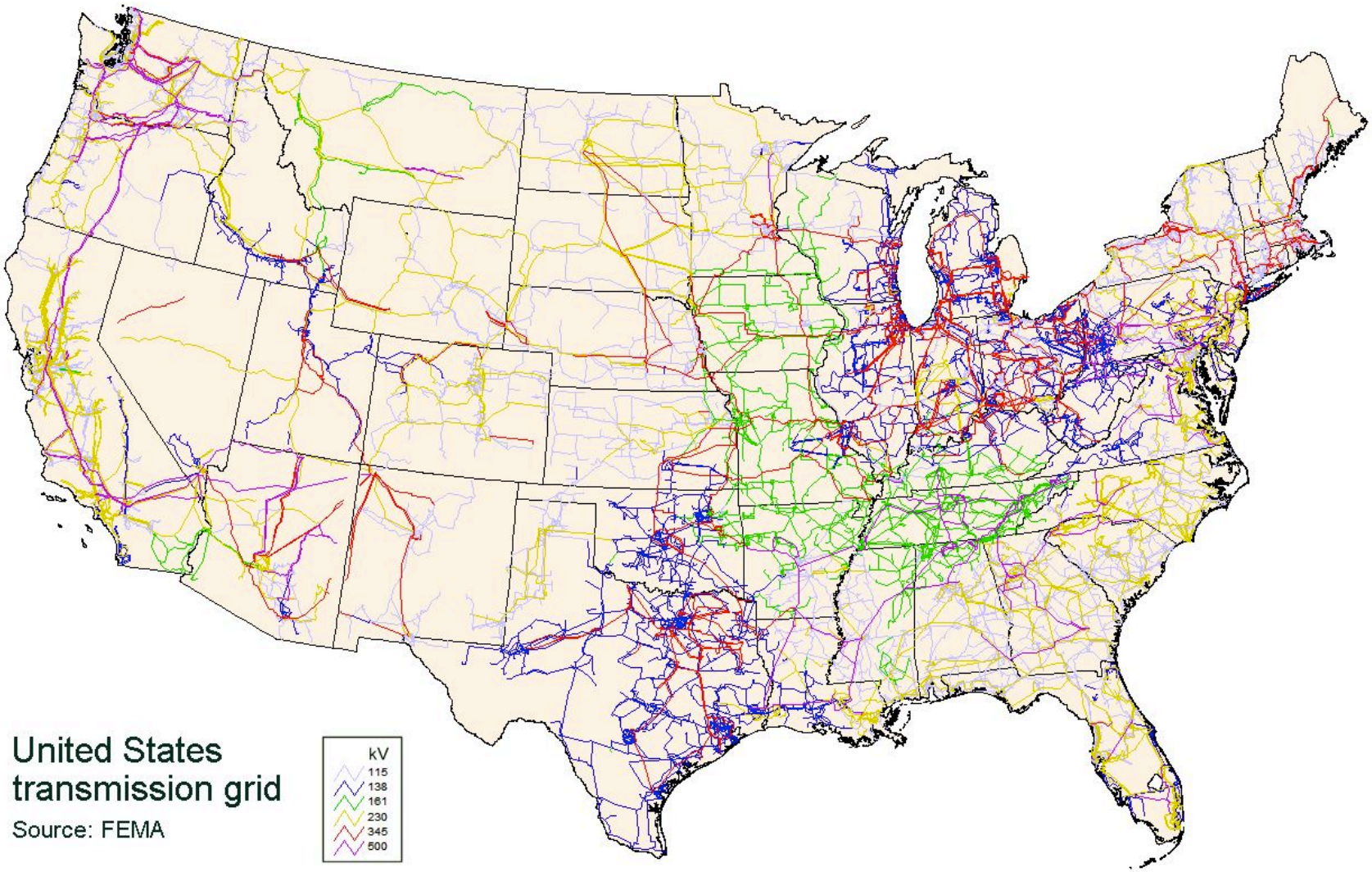
- nodes store information, links associate information
  - citation network (directed acyclic)
  - the web (directed)
  - peer-to-peer networks
  - word networks
  - networks of trust
  - software graphs
  - bluetooth networks
  - home page/blog networks



Technological Networks

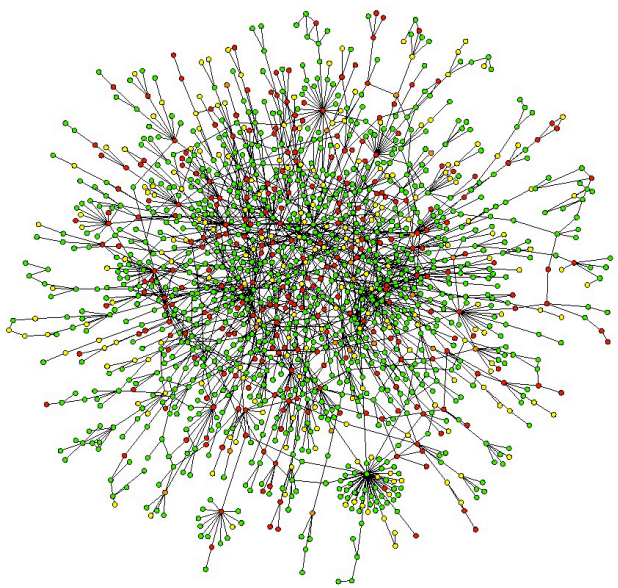
- networks built for distribution of a commodity
  - the internet, power grids, telephone networks
  - airline networks, transportation networks

US power grid



Biological Networks

- biological systems represented as networks
  - protein-protein interaction networks
  - gene regulation networks
  - gene co-expression networks
  - metabolic pathways
  - the food web
  - neural networks



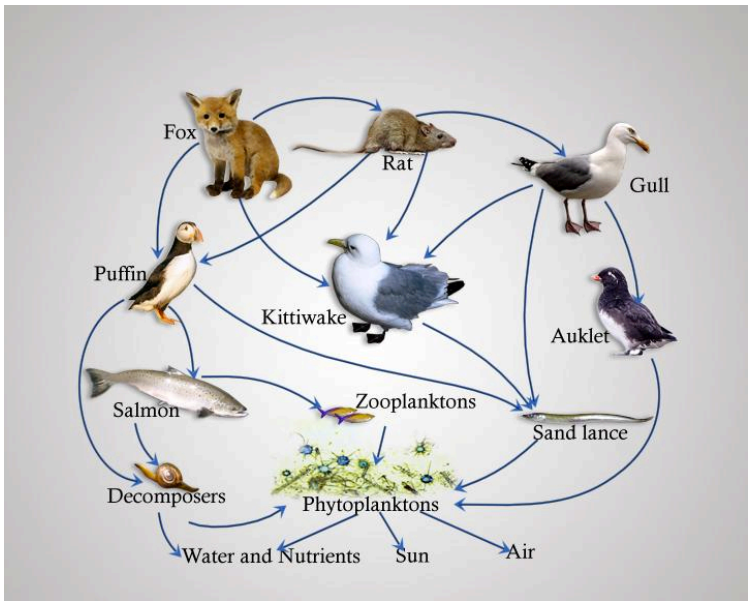


Photo-sharing Site

flickr from YAHOO!

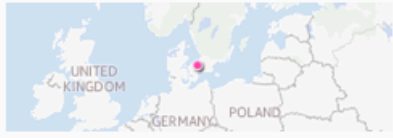
Signed in as Aris Gionis Help Sign Out

Home You Organize & Create Contacts Groups Explore Upload Search

Favorite Actions Share ← Newer Older →

By michael.dreves
Michael Dreves Beier + Add Contact

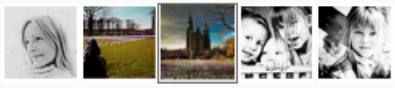
This photo was taken on April 7, 2010 in Tornebuskegade, Copenhagen, Hovedstaden, DK, using a Canon EOS 5D Mark II.



7,584 390 190 9

This photo belongs to

michael.dreves' photostream (454)



This photo also appears in

- flickr - Most interesting (set)
- Project 365 (set)
- HDR compilations (set)
- Copenhagen (set)
- ***Flickr Global (group)
- Art of Images...(P1/A3) / Not... (group)
- Danmark (group)
- FlickrCentral (group)
- FlickrToday (only 1 pic per day) (group)

...and 63 more groups

People in this photo (add a person)

Adding people will share who is in this photo

Rosenborg, Copenhagen

19.365

Rosenborg Castle - where we keep the Kingdoms crown jewels.

This beautiful spot is in the heart of Copenhagen, at the Kings Garden.
The photograph was shot on a nice spring day, with wonderful flickr friends on a Copenhagen walk.

Comments and faves

What is the Underlying Graph?

- nodes: photos, tags, users, groups, albums, sets, collections, geo, query, ...
- edges: upload, belong, tag, create, join, contact, friend, family, comment, fave, search, click, ...
- also many interesting induced graphs
  - tag graph: based on photos
  - tag graph: based on users
  - user graph: based on favorites
  - user graph: based on groups

- which graph to pick — not an easy choice

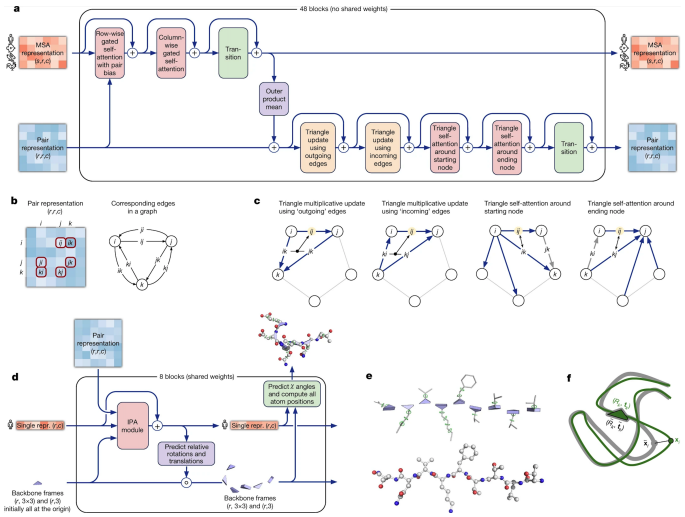
Recurring Theme

- social media, user-generated content
- user interaction is composed by many atomic actions
  - post, comment, like, mark, join, comment, fave, thumps-up, ...
  - generates all kind of interesting graphs to mine

Learning Node Embeddings in Practice

- **Challenge:** Learn node embeddings — map each node to a vector in $\mathbb{R}^d$ that captures its position & context in the graph.
- Many graph-based tasks (classification, recommendation, link prediction) need fixed-length feature vectors for nodes.
- **Idea:** DeepWalk [Perozzi et al., 2014] and node2vec [Grover and Leskovec, 2016] learn node embeddings by treating random walks as "sentences" and applying *word2vec* [Mikolov et al., 2013].
- **Use Cases:**
  - **Pinterest** uses node2vec to recommend pins based on user-item interaction graphs.
  - **Alibaba** applies DeepWalk-style embeddings to model behavior graphs in fraud detection.

AlphaFold: A Graph-Inspired Model



Source *Highly accurate protein structure prediction with AlphaFold* [Jumper et al., 2021]

- AlphaFold breakthrough : predicts the 3D structure of proteins from their amino acid sequences.
- Relationship between amino acids = graph

Demis Hassabis and John Jumper
Nobel Prize 2024

Graph Neural Networks (GNNs)

GNNs outperform classical methods when structure matters, relationships are key, and feature engineering is hard or domain-specific.

- Drug Discovery: Predict molecular properties and protein–ligand binding using graph-based models of molecules.
- Knowledge Graph Completion: Power question answering and semantic search by inferring missing edges in massive knowledge graphs.
- Traffic and Infrastructure Modeling: Learn from dynamic road networks, power grids, or sensor networks, accounting for both spatial and relational dependencies.

Graph Neural Networks (GNNs)

Graphs power modern infrastructure—from your code editor to your firewall.

- </> Code Intelligence: Source code forms abstract syntax trees and dependency graphs. GNNs predict bugs, suggest fixes, and find vulnerabilities — powering tools like GitHub Copilot and static analyzers.
- 🔒 Cybersecurity: IT systems form access graphs (users, files, servers). GNNs detect anomalous behavior, lateral movement, and insider threats. Used in threat hunting, zero-day attack detection, and role inference.
- 💵 FinTech, 👥 Social Networks ...

RelationalAI

RelationalAI brings:

- graph reasoning ...
- rule-based workflows ...
- predictive/prescriptive logic directly into the observability stack—all running securely inside Snowflake ...
- founded on state-of-the-art data management ideas (Worst Case Optimal Joins) and graph algorithms

The 7 Deadly Sins of Data Systems - Bob Muglia

1. **Data breach** — Unauthorized access to sensitive data
2. **Data loss or corruption** — Missing, overwritten, or damaged records
3. **Wrong results** — Incorrect answers due to logic, staleness, or bugs
4. **Service disruption** — Downtime, crashes, or unavailability
5. **Billing errors** — Inaccurate cost computation or user charges
6. **Unannounced behavior changes** — Silent API or semantics shifts
7. **Query failures** — Unreliable execution, timeouts, or exceptions

RelationalAI: Building reliable, trustworthy data systems means defending against all seven.

Background

- Quick Database Reminder: Join
- Graph Theoretic Concepts
- Linear Algebra Basics
- Matrix Representations of Graphs
- Sparsifiers
- Random graphs

Database Reminder

Join:

- Given two relations R and S with a common attribute
- The join $R \bowtie_{R.A=S.B} S$ combines tuples where the values on the common attribute match

Example:

Employee	DeptID
Alice	10
Bob	20
Carol	10

DeptID	DeptName
10	HR
20	IT

Result (Join on DeptID):

Employee	DeptID	DeptName
Alice	10	HR
Bob	20	IT
Carol	10	HR

Graph Theoretic Concepts

- graph $G = (V, E)$ with vertices V and edges $E \subseteq V \times V$
- degree of a node $u \in V$ with respect to $X \subseteq V$ is

$$\deg_X(u) = |\{v \in X \text{ such that } (u, v) \in E\}|$$

- degree of a node $u \in V$ is $\deg(u) = \deg_V(u)$
- edges between $S \subseteq V$ and $T \subseteq V$ are

$$E(S, T) = \{(u, v) \text{ such that } u \in S \text{ and } v \in T\}$$

  use shorthand $E(S)$ for $E(S, S)$, and $e(S) = |E(S)|$.
- graph cut is defined by a subset of vertices $S \subseteq V$
- edges of a graph cut $S \subseteq V$ are $E(S, \bar{S})$
- induced subgraph by $S \subseteq V$ is $G(S) = (S, E(S))$
- induced triangles:
  $t(S) = \{(u, v, w) \mid (u, v), (u, w), (v, w) \in E(S)\}$

Graph Theoretic Concepts

- average degree/degree density :

$$d(S) = \frac{2\,|E(S,S)|}{|S|} = \frac{2\,|E(S)|}{|S|}, \quad \rho(S) = \frac{|E(S)|}{|S|}$$

(sometimes just drop 2)

- edge ratio:

$$f_e(S) = \frac{|E(S,S)|}{\binom{|S|}{2}} = \frac{|E(S)|}{\binom{|S|}{2}} = \frac{2\,|E(S)|}{|S|(|S|-1)}$$

- triangle density:

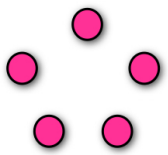
$$t(S) = \frac{|t(S)|}{|S|}$$

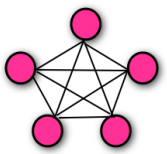
- triangle ratio:

$$\tau(S) = \frac{|T(S)|}{\binom{|S|}{3}}$$

Graph Theoretic Concepts

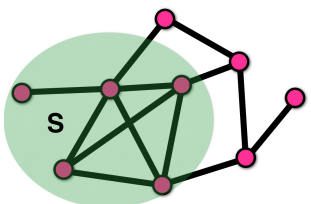
Edge ratio: $0 \leq \boxed{f_e(S) = \frac{e(S)}{\binom{|S|}{2}}} \leq 1$





Independent set

Clique

Degree density: $\boxed{\rho(S) = \frac{e(S)}{|S|}}$. E.g., 

$\rho(S) = \frac{7}{5}$

Graph Theoretic Concepts

- **Definition:** The **arboricity** $\alpha(G)$ of an undirected graph $G = (V, E)$ is the minimum number of forests needed to cover all edges of G.
- **Nash-Williams Theorem:**

$$\alpha(G) = \max_{S \subseteq V,\ |S| \geq 2} \left\lceil \frac{|E(S)|}{|S| - 1} \right\rceil$$

where $E(S)$ denotes the edges induced by the node set $S \subseteq V$.

- **Connection to Densest Subgraph:**

$$\rho^* = \max_{S \subseteq V} \frac{|E(S)|}{|S|} \quad \Rightarrow \quad \alpha(G) = \Theta(\rho^*)$$

(exercise: find constants c_1, b_1, c_2, b_2 such that $c_1\rho^\star + b1 \leq \alpha \leq c2\rho^\star + b_2$)

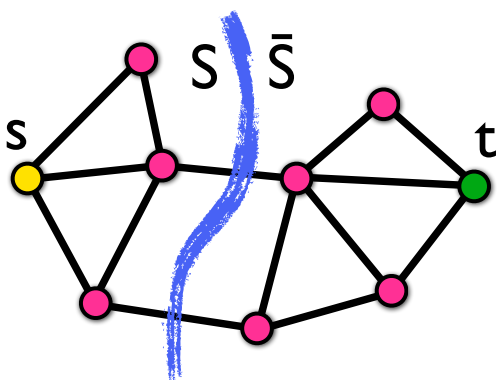
Useful in graph algorithms: controlling recursion, subgraph

Graph Theoretic Concepts

- Given a partition of the node set $V = S \cup \bar{S}$, we define the cut as

$$\text{cut}(S, \bar{S}) = \sum_{\substack{u \in S \\ v \in \bar{S}}} \mathbf{1}_{(u,v) \in E}$$

(st-)min-cut problem



- source $s \in V$, destination $t \in V$
- find $S \subseteq V$, s.t.,
- $s \in S$ and $t \in \bar{S}$, and
- minimize $e(S, \bar{S})$
- polynomially-time solvable
- equivalent to max-flow problem

Graph Theoretic Concepts

- **Conductance** of a node set $S \subseteq V$ is defined as

$$\phi(S) = \frac{e(S, \bar{S})}{\min(\text{vol}(S), \text{vol}(\bar{S}))}$$

  where $\text{vol}(S) = \sum_{i \in S} deg(i)$.

- **Graph conductance**, is defined as $\phi(G) = \min_S \phi(S)$.
- **Expanders:** Intuitively, a graph that contains no set S with low conductance.
- **Intuition:** If $\phi(G)$ is low, there exists a community, a set of nodes well connected inside but loosely connected to the rest of the network.

Matrix-Vector Multiplication: Background and Complexity

Matrix-vector multiplication is a core primitive in graph algorithms and numerical computing.

- **Problem:** Given a matrix $A \in \mathbb{R}^{n \times n}$ and a vector $x \in \mathbb{R}^n$, compute

$$y = Ax, \qquad y_i = \sum_{j=1}^{n} A_{ij} x_j$$

- **Naïve Algorithm:** For a dense matrix, this requires $\mathcal{O}(n^2)$ operations.
- **Sparse Case:** If A has m non-zero entries, then

  Matrix-vector multiplication takes $\mathcal{O}(n + m)$ time.

- **Applications:**
  - Power iteration (eigenvector computation), solving linear systems, iterative methods, PageRank and diffusion

Matrix-Matrix Multiplication: Background and Complexity

Matrices' multiplication is also fundamental across numerical and graph algorithms.

- **Problem:** Given two $n \times n$ matrices A and B, compute their product:

$$C = AB, \quad C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

- **Naive yet practical Algorithm:** n matrix-vector mults $\rightarrow$ 3 nested loops $\rightarrow$ $O(n^3)$ operations.
- **Breakthrough:** Strassen showed how to multiply two 2×2 matrices using only 7 multiplications. This naturally gives a recursive $O(n^{\log_2 7}) \approx O(n^{2.81})$ algorithm
- **Current State of the Art:** $O(n^{\omega}), \quad \omega < 2.37286$ [Alman and Williams, 2024]

Eigenvalues and the Spectral Theorem

Definition: A scalar λ is an **eigenvalue** of a matrix M if there exists a nonzero vector x such that

$$Mx = \lambda x$$

The vector x is called the corresponding **eigenvector**.

Spectral Theorem: If M is a symmetric matrix ($M = M^\top$), then:

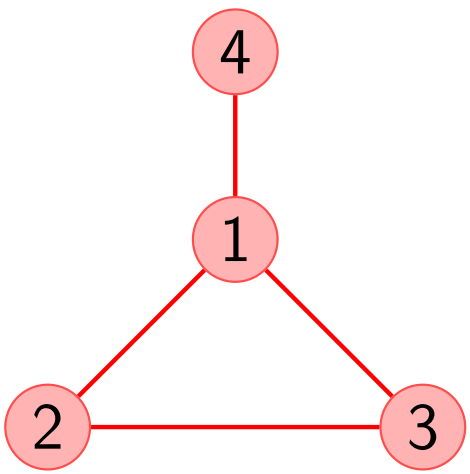
- M has n real eigenvalues $\lambda_1, \ldots, \lambda_n$
- M is diagonalizable: $M = X\Lambda X^\top$
- The eigenvectors (columns of X) can be chosen to be orthonormal

$$\boxed{M = \sum_{i=1}^{n} \lambda_i x_i x_i^\top}$$

Each x_i is an eigenvector and λ_i its corresponding eigenvalue.

Adjacency Matrix

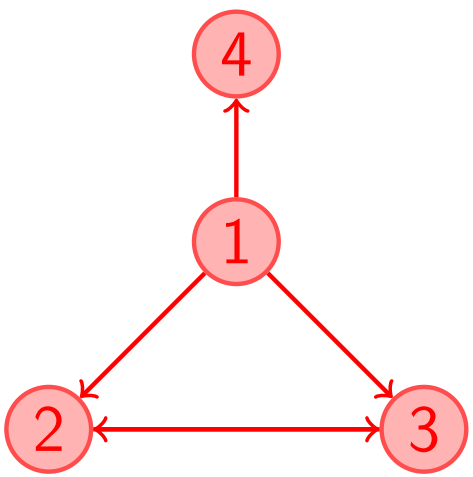
$$A_{uv} = \begin{cases} 1 & \text{if } \{u, v\} \in E(G) \\ 0 & \text{otherwise} \end{cases}$$



$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Adjacency Matrix

$$A_{uv} = \begin{cases} 1 & \text{if } (u, v) \in E(G) \\ 0 & \text{otherwise} \end{cases}$$

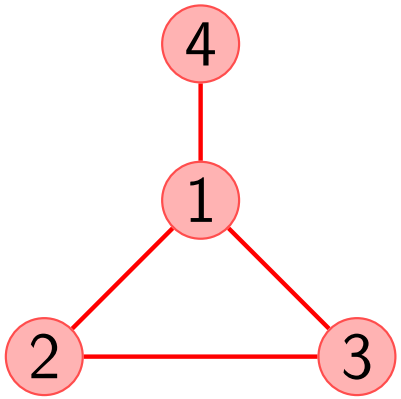


$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Laplacian

$$L_{uv} = \begin{cases} deg_u & \text{if } u = v \\ -1 & \text{if } (u, v) \in E(G) \\ 0 & \text{otherwise} \end{cases}$$

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



Symmetric, positive semidefinite matrix

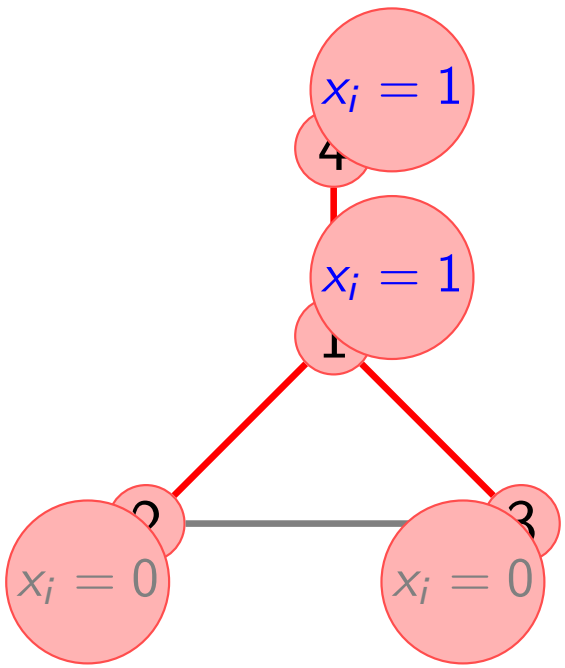
Laplacian and Cut Size

- Let $\mathbf{x} \in \{0,1\}^n$ be the indicator vector of a node set $S \subseteq V$
- Then:

$$\mathbf{x}^\top \mathbf{L}\mathbf{x} = \sum_{(u,v)\in E} (x_u - x_v)^2$$

- Each edge crossing from S to $\bar{S}$ contributes 1
- So:

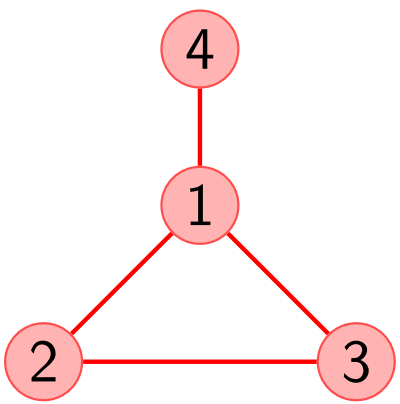
$$\mathbf{x}^\top \mathbf{L}\mathbf{x} = \text{cut}(S, \bar{S})$$



Normalized Laplacian $\mathcal{L} = I - D^{-1/2}AD^{-1/2}$

$$\mathcal{L}_{uv} = \begin{cases} 1 & \text{if } u = v \\ -\frac{1}{\sqrt{d_u d_v}} & \text{if } (u,v) \in E(G) \\ 0 & \text{otherwise} \end{cases}$$

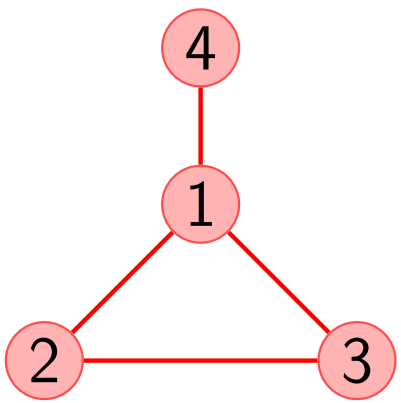
$$\mathcal{L} = \begin{pmatrix} 1 & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{6}} & 1 & -\frac{1}{2} & 0 \\ -\frac{1}{\sqrt{6}} & -\frac{1}{2} & 1 & 0 \\ -\frac{1}{\sqrt{3}} & 0 & 0 & 1 \end{pmatrix}$$



Symmetric, positive semidefinite

Random Walk Matrix $P = D^{-1}A$

$$P_{uv} = \begin{cases} \frac{1}{d_u} & \text{if } (u,v) \in E(G) \\ 0 & \text{otherwise} \end{cases}$$



$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{A} = \begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Row-stochastic

Why Does the Random Walk Matrix Have Real Eigenvalues?

- The random walk matrix $P = D^{-1}A$ is generally **not symmetric**.
- Yet, all its eigenvalues are **real**.
- This holds because P is **similar** to a symmetric matrix:

$$P \sim D^{-1/2}AD^{1/2} = I - \mathcal{L}_{\text{sym}}$$

  where $\mathcal{L}_{\text{sym}} = I - D^{-1/2}AD^{-1/2}$ is the normalized Laplacian.

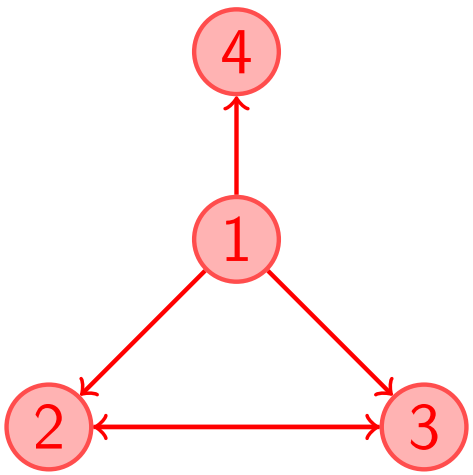
- **Reminder (Similarity Transformation):**
  - Matrices A and B are similar if $A = SBS^{-1}$ for some invertible matrix S.
  - Similar matrices have the same eigenvalues.
- Since $\mathcal{L}_{\text{sym}}$ is symmetric, its eigenvalues are real, therefore, so are those of P.

Walks

- A walk of length r in a directed graph:

$$u_0 \to u_1 \to \cdots \to u_r$$

  where a node can be used more than once.
- Closed walk when: $u_0 = u_r$, e.g., $2 \to 3 \to 2 \to 3 \to 2$



Graph Theoretic Concepts

Spectral clustering (SC) – Cheeger's Inequality: Let $0 = \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_n \leq 2$ be the eigenvalues of ...

- the normalized Laplacian matrix

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2} \quad \text{[Alon and Milman, 1985]}$$

- combinatorial Laplacian matrix

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2deg_{\max}\lambda_2} \quad \text{[Alon and Milman, 1985]}$$

Cheeger's inequality forms a foundational bridge between the combinatorial and spectral views of graph structure.

Spectral Sweep: Partitioning via the Fiedler Vector

1. **Input:** Undirected graph $G = (V, E)$ with normalized Laplacian $\mathcal{L}$.
2. **Compute the Fiedler vector** v_2: the eigenvector corresponding to the smallest non-zero eigenvalue of $\mathcal{L}$.
3. **Sort** the nodes according to the entries of v_2:

$$v_2(i_1) \leq v_2(i_2) \leq \cdots \leq v_2(i_n)$$

4. **Sweep cut:** For each prefix $S_k = \{i_1, \ldots, i_k\}$, compute the conductance:

$$\phi(S_k) = \frac{e(S_k, \bar{S}_k)}{\min(\text{vol}(S_k), \text{vol}(\bar{S}_k))}$$

5. **Output:** The set S_k with the smallest conductance.

Guarantees: Cheeger's inequality relates $\phi(G)$ to λ_2 — the quality of this cut is provably good.

Spectral Clustering (Multiple Eigenvectors)

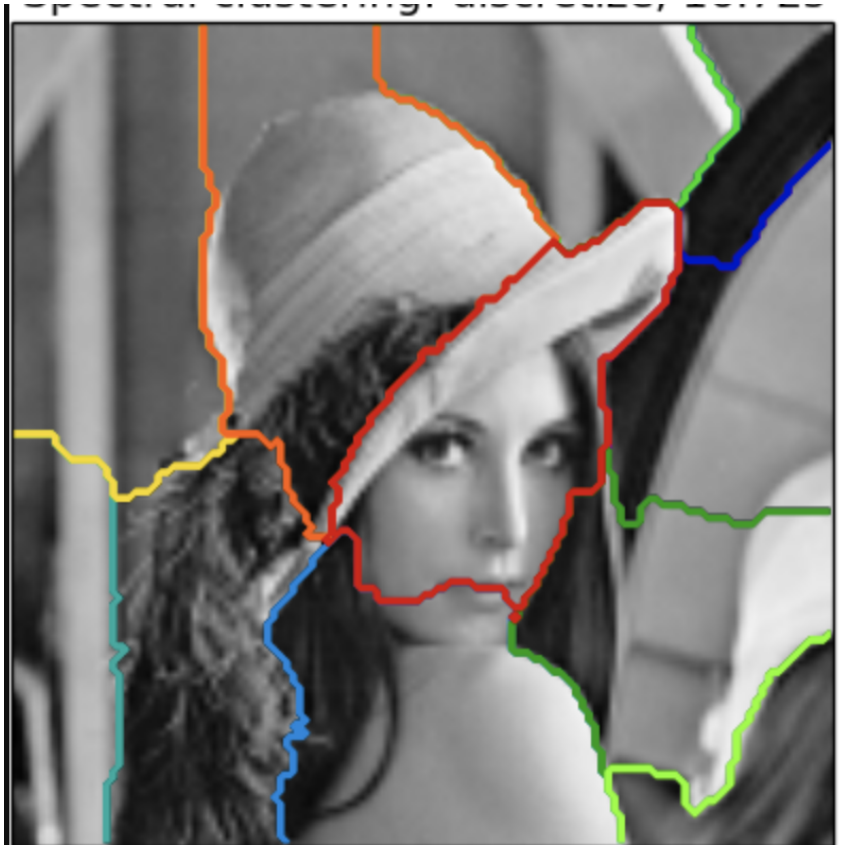
Key Idea: Use the bottom k eigenvectors of the (normalized) Laplacian as a low-dimensional embedding of nodes.

1. Construct the graph Laplacian $\mathcal{L} = I - D^{-1/2}AD^{-1/2}$
2. Compute the bottom k eigenvectors $v_1, \ldots, v_k \in \mathbb{R}^n$
3. Form matrix $V \in \mathbb{R}^{n \times k}$ with v_i as columns
4. Normalize each row of V to unit length (optional)
5. Apply **k-means** clustering on the rows of V
6. Assign cluster labels from k-means back to nodes

Interpretation

Each node is represented as a point in $\mathbb{R}^k$ capturing structural similarity; clustering in this space often reveals community structure.

Spectral Clustering (Multiple Eigenvectors)



- Recommended readings: Two highly influential papers [Shi and Malik, 2000, Ng et al., 2001]

What is a graph sparsifier?

- **Definition.** A sparsifier of a graph $G(V, E)$ is a sparse graph H, possibly weighted, that is similar to G in *some* useful notion.

E.g., if we want to preserve:

1. Triangle counts: Triangle sparsifier
2. Densest subgraph: Densest subgraph sparsifier
3. Spectral properties: Spectral sparsifier

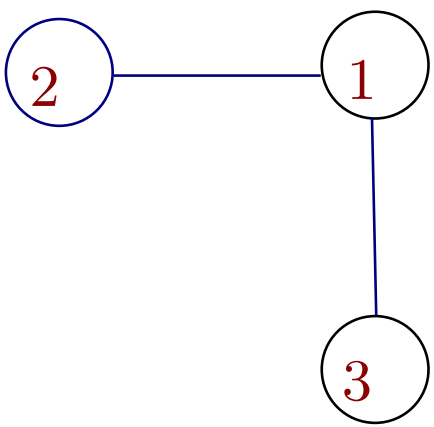
Why do you believe this concept is useful in practice?

- ✔**Faster Algorithms on Large Graphs:**
  - Approximate min-cuts, Laplacian solvers, PageRank.
  - Reduce runtime while preserving key properties.
- ✔**Scalable Machine Learning:**
  - Speed up GNN training by reducing edge count.
  - Preserve spectral or structural features.
- ✔**Efficient Network Monitoring:**
  - Simplify traffic networks or sensor layouts.
  - Maintain connectivity and flow approximations.
- ✔**Graph Visualization and Exploration:**
  - Remove redundant edges for cleaner views.
  - Useful for social, citation, or biological networks.
- ✔**Streaming and Distributed Algorithms:**
  - Estimate metrics with limited memory or bandwidth.
  - Use in sketching or sublinear algorithms.
- ✔**Privacy-Preserving Data Release:**
  - Publish sparsified versions of graphs.
  - Preserve utility while enhancing anonymity.

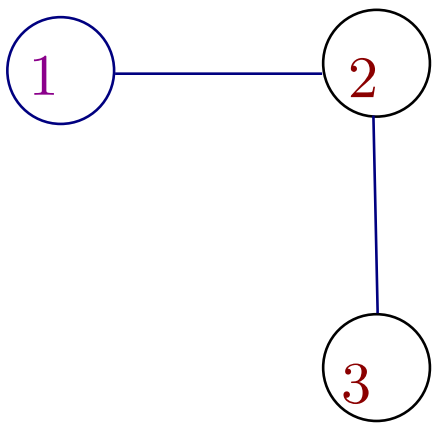
random graphs

A random graph is a set of graphs together with a probability distribution on that set.

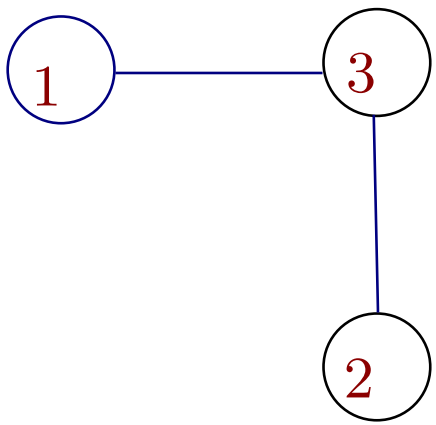
Example:



Probability $\frac{1}{3}$



Probability $\frac{1}{3}$

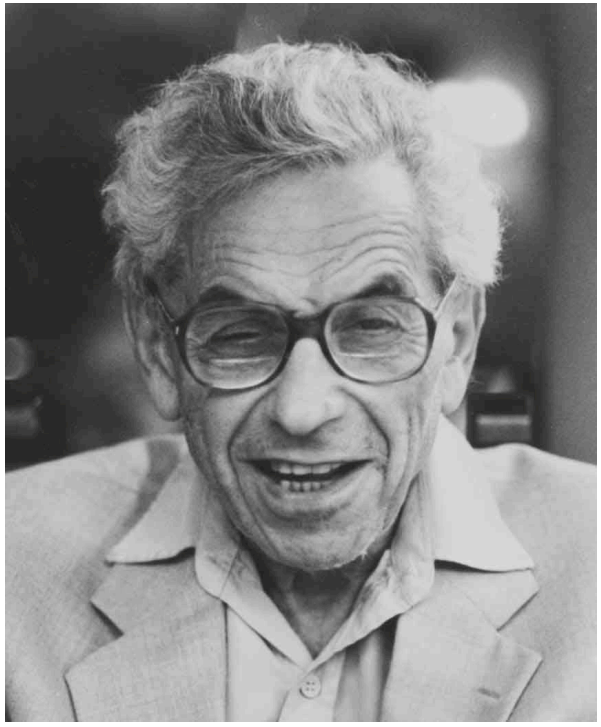


Probability $\frac{1}{3}$

A random graph on $\{1, 2, 3\}$ with 2 edges with the uniform distribution.

random graphs

- Erdős-Rényi random graph model



Paul Erdős
1913 – 1996

Alfréd Rényi
1921 – 1970

random graphs

- the $G(n, p)$ model:
- n : the number of vertices
- $0 \leq p \leq 1$: probability
- for each pair (u, v), independently generate the edge (u, v) with probability p
- $G(n, p)$ a family of graphs, in which a graph with m edges appears with probability $p^m(1-p)^{\binom{n}{2}-m}$
- the $G(n, m)$ model: related, but not identical

random graphs

- the $G(n, p)$ model:
- n : the number of vertices
- $0 \leq p \leq 1$: probability
- for each pair (u, v), independently generate the edge (u, v) with probability p
- $G(n, p)$ a family of graphs, in which a graph with m edges appears with probability $p^m(1-p)^{\binom{n}{2}-m}$
- the $G(n, m)$ model: related, but not identical

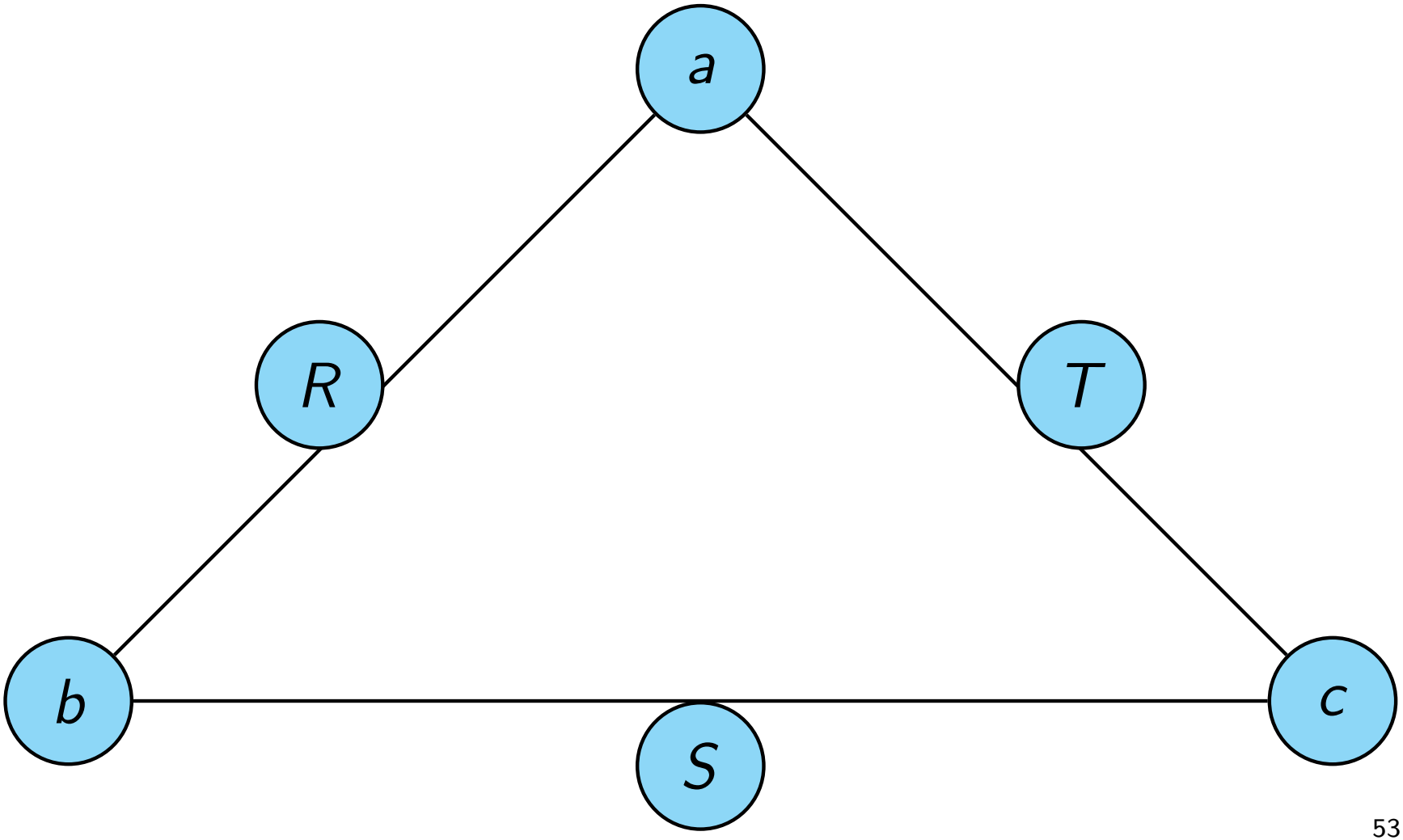
random graphs

- the $G(n, p)$ model:
- n : the number of vertices
- $0 \leq p \leq 1$: probability
- for each pair (u, v), independently generate the edge (u, v) with probability p
- $G(n, p)$ a family of graphs, in which a graph with m edges appears with probability $p^m(1-p)^{\binom{n}{2}-m}$
- the $G(n, m)$ model: related, but not identical

Counting Triangles

Triangle Query as a Database Join

$$Q_\Delta(a, b, c) = R(a, b) \wedge S(b, c) \wedge T(a, c)$$



Triangle Counting as a 3-Way Join

Setup: Consider a graph $G = (V, E)$ where the edge list is stored as a relation (i.e., database table):

$$\texttt{Edges}(u, v)$$

Each row corresponds to an undirected edge $\{u, v\}$ with $u < v$.

Goal: Count the number of triangles $\{u, v, w\}$ such that all three edges exist: $(u, v), (v, w), (w, u)$.

Relational Join View:

$$\texttt{Edges}(u, v) \bowtie \texttt{Edges}(v, w) \bowtie \texttt{Edges}(w, u)$$

Triangle counting is fundamentally a conjunctive query on three binary relations.

Triangle Counting in SQL and Datalog

SQL Query:

```
SELECT COUNT(*) AS triangle_count
FROM Edges e1
JOIN Edges e2 ON e1.v = e2.u
JOIN Edges e3 ON e2.v = e3.u
WHERE e3.v = e1.u
  AND e1.u < e1.v
  AND e1.v < e2.v;
```

Datalog-style Rule:

$$Q(u,v,w) = E(u,v),\ E(v,w),\ E(w,u),\ u<v,\ v<w$$

Geometric View of Triangle Query Output Size

Idea: Think of the triangle query output $Q_\triangle$ as a set of points (a, b, c) in 3D space.

- The projection onto the (A, B)-plane lies in R
- The projection onto the (B, C)-plane lies in S
- The projection onto the (A, C)-plane lies in T

Theorem (Loomis–Whitney, 1949): Let $A \subset \mathbb{R}^3$ be a finite set of points. Then,

$$|A| \leq \sqrt{|\pi_{xy}(A)| \cdot |\pi_{yz}(A)| \cdot |\pi_{xz}(A)|}$$

where $\pi_{xy}(A)$ is the projection of A onto the (x, y)-plane, and similarly for the other axes.

Discrete Version (for triangle query):

$$|Q_\triangle| \leq \sqrt{|R| \cdot |S| \cdot |T|}$$

Limitations of Binary Join Plans

Example: Consider domains:

$$\{a_0, \ldots, a_N\}, \quad \{b_0, \ldots, b_N\}, \quad \{c_0, \ldots, c_N\}$$

Define the input relations:

$$R^I = (\{a_0\} \times \{b_0, \ldots, b_N\}) \cup (\{a_0, \ldots, a_N\} \times \{b_0\})$$
$$S^I = (\{b_0\} \times \{c_0, \ldots, c_N\}) \cup (\{b_0, \ldots, b_N\} \times \{c_0\})$$
$$T^I = (\{c_0\} \times \{a_0, \ldots, a_N\}) \cup (\{c_0, \ldots, c_N\} \times \{a_0\})$$

Observation:

- Each relation has size $\mathbf{2N + 1}$
- The *output* of the triangle join has size $\mathbf{3N + 1}$ (**Exercise: verify!**)
- However, every **pairwise** join:

$$R \bowtie S, \quad S \bowtie T, \quad R \bowtie T$$

produces $\Omega(N^2)$ intermediate tuples.

Counting the triangles in G

Conclusion: Binary join plans (two-way joins) *cannot* achieve the optimal $O(N^{3/2})$ Loomis-Whitney bound. A multiway join strategy is needed.

- RelationalAI provides worst case optimality guarantees, in the sense that the run time is linear to the maximum possible size of the output of the query over any database with given input relation sizes.
- In the context of graphs, WCOJ guarantee triangle counting in time $O(n + m + m^{3/2})$.

Matrix-Matrix Multiplication and Walks

Theorem: Let $G(V, E)$ be a directed graph with adjacency matrix A. The number of walks from node u to node v in G of length r is given by A^r_{uv}

Quick exercises:

- Assuming that G has no self loops, what is the trace of A?
- $trace(A^2) = ?$
- $trace(A^3) = ?$

Theorem: We can count triangles in G in time $O(n^\omega)$ via matrix-matrix multiplication.

Answers

- $trace(A) = 0$
- $trace(A^2) = 2m$
- $trace(A^3) = 6t$

Exact triangle counting

- Clearly a naive cubic algorithm (i.e., $O(\binom{n}{3})$ can list (and hence) count triangles.
- ✔Matrix multiplication
- Node-iterator
- Edge-iterator
  - Actually, we will show an improved triangle listing algorithm due to Chiba and Nishizeki, 1985 that runs in $O(m\alpha(G))$ time where $\alpha(G)$ is the arboricity of the graph.

Node Iterator

Goal: Enumerate all triangles in $G = (V, E)$

Input: Graph $G = (V, E)$, total order $<$ over V

Algorithm:

- For each vertex $v \in V$:
  - For all pairs of neighbors $\{u, w\}$ of v:
    - If $\{u, w\} \in E$ and $u < v < w$, then output triangle $\{u, v, w\}$

Output: All triangles in G

Complexity:

$$\mathcal{O}(n \cdot d_{\max}^2) \quad \text{in worst case}$$

where $d_{\max}$ is the maximum degree of any vertex.

Chiba–Nishizeki Triangle Counting

Goal: Count all triangles in $G = (V, E)$ efficiently.
Algorithm:

1. Sort vertices such that $\deg(v_1) \geq \deg(v_2) \geq \cdots \geq \deg(v_n)$
2. For $u = v_1, \ldots, v_{n-2}$ do:
   1. For each $v \in N(u)$, mark v
   2. For each $v \in N(u)$ do:
      1. For each $w \in N(v)$:
      2. If w is marked: output triangle $\{u, v, w\}$
      3. Unmark v
   3. Remove u from G to avoid duplicates

Theorem (Chiba–Nishizeki '85): Run-time is $\mathcal{O}(m \cdot \alpha(G))$ where $\alpha(G)$ is the arboricity of G since

$$\sum_{\{u,v\} \in E} \min\{\deg(u), \deg(v)\} \leq 2 \cdot \alpha(G) \cdot m \in \mathcal{O}(m^{3/2}).$$

EigenTriangle: Spectral Triangle Estimation

- **Goal:** Estimate the number of triangles in a graph via spectral methods.
- **Key insight:** The trace of A^3 counts 6 times the number of triangles.

$$t = \frac{1}{6}\,\mathrm{Tr}(A^3)$$

Can you prove this formula?

- **Spectral formulation:** Since A is symmetric for undirected graphs:

$$\mathrm{Tr}(A^3) = \sum_{i=1}^{n} \lambda_i^3$$

where λ_i are the eigenvalues of A.

Theorem: EigenTriangle Global and Local

Global Triangle Count:

$$6t(G) = \sum_{i=1}^{|V|} \lambda_i^3$$

where:

- $t(G)$ is the number of triangles in $G(V, E)$
- $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{|V|}$ are eigenvalues of adjacency matrix A_G

Local Triangle Count:

$$2t(i) = \sum_{j=1}^{|V|} \lambda_j^3 u_{ij}^2$$

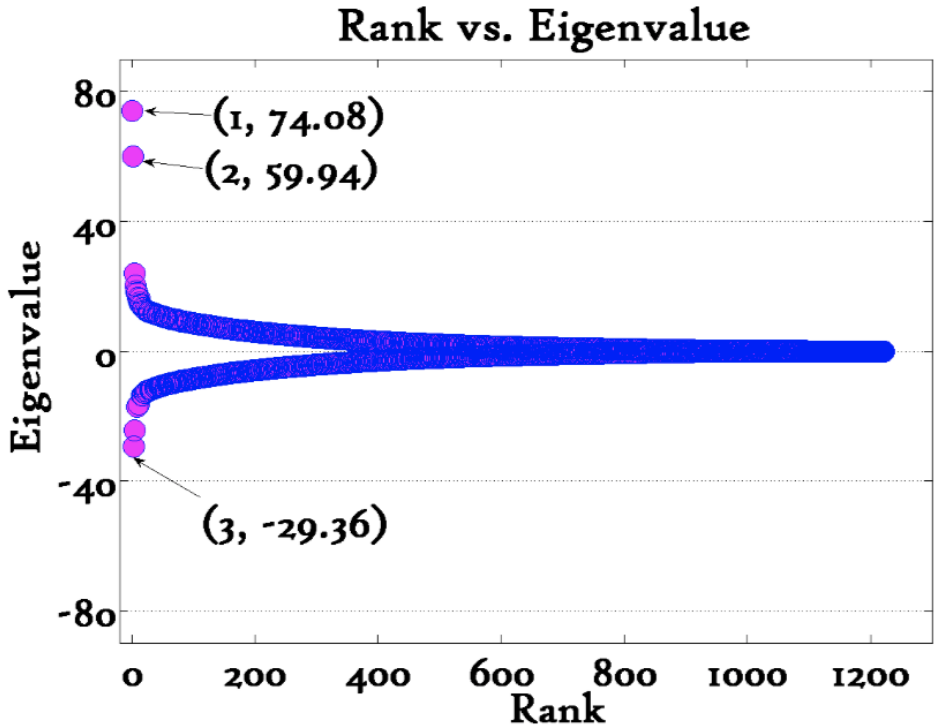
where:

- $t(i)$ is the number of triangles vertex i participates in
- u_j is the j-th eigenvector
- u_{ij} is the i-th coordinate of u_j

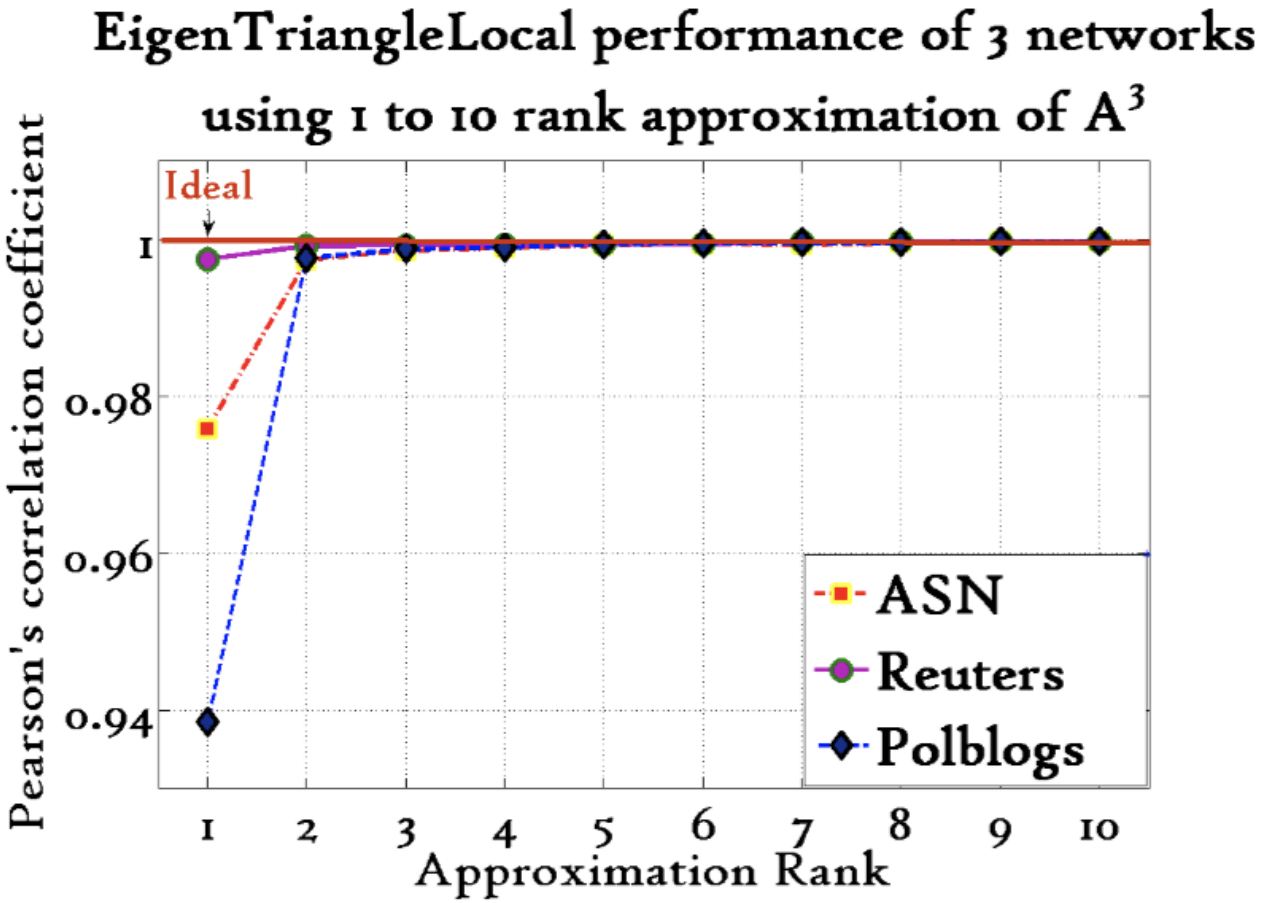
Fast and interpretable triangle count from spectral signature.

EigenTriangle's Key idea

- EigenTriangle: Only few eigenvalues suffice to get a good approximation in real-world networks.
  - Matrix-matrix multiplication $O(n^\omega) \rightarrow$ matrix vector multiplication $O(n+m)$



EigenTriangle on 3 datasets



Github demo notebook

Triangle Sparsifiers

Counting Triangles In Massive Graphs with a Coin [Tsourakakis et al. 2009, 2010, 2012]

Algorithm (T. et al.)

Input: Graph $G([n], E)$

1. Toss a coin independently for each edge, and keep it with probability p. Let $G'([n], E')$ be the resulting sparsified graph
2. $t' \leftarrow$ Count-Triangles(G')
3. Return $T \leftarrow \frac{t'}{p^3}$

Clearly,

$$\boxed{\mathbb{E}[t'] = p^3 t \rightarrow \mathbb{E}[T] = t}$$

How should we choose p in order $t \approx T$ whp?

Do we need to toss a coin for each edge?

Claim. The edge sampling can run in $O(pm)$ expected time.

- We do not wish to "toss a p-coin" m times in order to construct E'.
- Sublinear sampling: if p is small, then we toss significantly less than m coins!
- **How can we achieve this?**
- Observation. The number of unsuccessful events, i.e., edges which are not selected in our sample, until a successful one follows a **geometric distribution**.
- Key idea: Generate these numbers assuming access to uniform RV in $[0, 1]$ (**Exercise**).

Triangle Sparsifiers

- Let $X_e = 1$ or 0 depending on whether the edge e of graph G survives in G'.
- Then

$$T = \sum_{\Delta(e,f,g)} X_e X_f X_g$$

  where $\Delta(e, f, g) = 1(\text{edges } e, f, g \text{ form a triangle})$.
- What is the expectation $\mathbb{E}[T]$?

$$\mathbb{E}[T] = p^3 t$$

- **How small can p be, and still guarantee concetration?**

Math Challenge Probabilistic Concentration of multivariate polynomials.

Triangle Sparsifiers

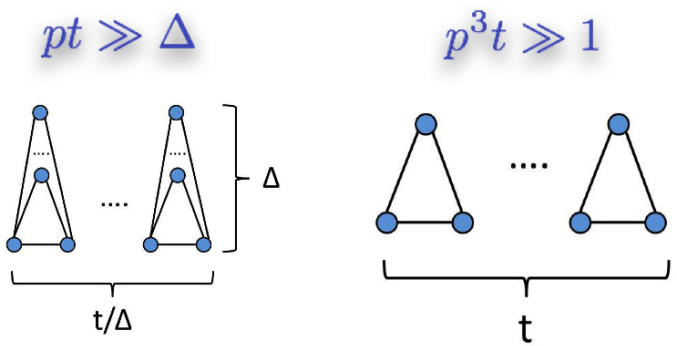
Theorem: **[T., Kolountzakis, Miller]** Let Δ be the maximum # triangles an edge is contained (notice $\Delta \leq n-2$). **If**

$$p \geq \max\left\{\frac{\tilde{O}(\Delta)}{t}, \frac{\tilde{O}(1)}{t^{1/3}}\right\}$$

then

$$\mathbf{Pr}\left[|T - \mathbb{E}[T]| \geq \epsilon \mathbb{E}[T]\right] \leq n^{-K}$$

for any constants $K, \epsilon > 0$.



Triangle Sparsifiers

Expected speedup: $O(\frac{1}{p^2})$ (node iterator)

Corollary 1: If $t \geq n^{3/2+\epsilon}$ and $\Delta \sim n$, we get $p = n^{-1/2} \rightarrow$ we can justify speedups $O(n)$!

\# nodes n, edges m, and triangles t in **Millions**.

Name (Abbr.)	n	m	t
⊙ AS-Skitter (AS)	1.7	11	28.8
⋆Flickr (FL)	1.9	15.6	548.7
⋆Livejournal (LJ)	5.3	48.7	310.9
⋆Orkut-links (OR)	3.1	116.6	622
◇Web-EDU (WE)	9.9	46.2	254.7
◇Wikipedia 2007/2 (Wiki)	3.6	42.4	102.4

Triangle Sparsifiers – Some Experiments

Results of experiments averaged over 5 trials using $p = 0.1$.

	Results		
	Exact	Triangle Sparsifier	
	Avg. time	Avg. err.% (std)	Avg. time (std)
AS	4.452	2.60 (0.022)	0.79 (0.023)
FL	41.981	0.11 (0.003)	0.96 (0.014)
LJ	50.828	0.34 (0.001)	2.85 (0.054)
OR	202.012	0.60 (0.004)	5.60 (0.159)
WE	8.502	0.70 (0.005)	2.79 (0.090)
Wiki	122.395	0.80 (0.015)	2.48 (0.012)

Triangle sparsifiers in practice

- Let's see how one may apply our theorem in practice.
- The next slide shows an example for the **Wikipedia 2005/09**.
  - $n = 1\,634\,989,\ m = 18\,540\,589,\ t = 44\,667\,095.$
- Doubling trick – concentration for $p = 0.02$.
- Speedups reported are over node iterator and averaged over the ten experiments

Triangle sparsifiers in practice – Doubling trick

p	Ratios $\frac{T}{t}$	Sparsification (secs)	Average Speedup (x*faster*)
0.01	0.9442, 1.4499 1.14, 1.37	8	7090
0.02	0.9112, 1.0183 0.8975, 0.9579 0.9716, 0.9771 0.9524, 0.9551 0.9606, 0.9716	8.64	2880
0.03	1.0043, 1.0336, 1.0035 0.9791, 1.0222 0.9865, 0.9816	8.65	1500
0.04	0.9895, 1.018	8.58	825
0.05	0.9979, 0.9716	9.84	402

Triple Sampling for Triangle Counting

Idea: Each triangle corresponds to a triple (u, v, w) of vertices. Let U be the set of all such vertex triples.

Algorithm:

- Uniformly sample s triples $(u, v, w) \in U$.
- For each sampled triple, check if it forms a triangle (i.e., all 3 edges exist).
- Let $X_i = 1$ if the i^{th} triple is a triangle, 0 otherwise.

Chernoff Bound:

$$\Pr\left[\left|\frac{1}{s}\sum_{i=1}^{s} X_i - \frac{t}{|U|}\right| > \epsilon\frac{t}{|U|}\right] \leq 2e^{-\epsilon^2 ts/(2|U|)}$$

Guarantee: If $s = \Omega\left(\frac{n^3 \log n}{t\epsilon^2}\right)$, then with high probability, the estimator approximates t within a factor of ϵ.

Hybrid Sampling for Triangle Counting

Key Insight: Optimize sampling by treating low- and high-degree vertices differently:

- Low-degree: Sample a vertex u with $\deg(u) \leq \sqrt{m}$ and pick **pairs of neighbors** (v, w).
- High-degree: Sample an **edge** (v, w) and then pick u from common neighbors.

- **Exercise**: Show that $|U| \leq O(m^{3/2})$

Runtime:

$$O\left(\frac{m^{3/2} \log n}{t\epsilon^2}\right) \quad \text{(via Chernoff bounds)}$$

Conclusion: Hybrid strategy achieves improved bounds by adapting sampling method to vertex degree.

Combining Sparsification and Sampling

Idea: Combine Triangle Sparsifiers with hybrid sampling for faster approximate triangle counting.

Steps:

- Sparsify the graph by retaining each edge independently with probability p
- Count triangles approximately via triple sampling on the sparsified graph

Runtime:

$$O\left(mp + \frac{\log n \cdot (mp)^{3/2}}{\epsilon^2 t p^3}\right) = O\left(mp + \frac{\log(n) \cdot m^{3/2}}{\epsilon^2 t p^{3/2}}\right)$$

Optimal p (balancing both terms): $p = \left(\frac{m^{1/2} \log n}{t\epsilon^2}\right)^{2/5}$

Datasets

Name	Nodes	Edges	Triangle Count	Description
AS-Skitter	1,696,415	11,095,298	28,769,868	Autonomous Systems
Flickr	1,861,232	15,555,040	548,658,705	Person to Person
Livejournal-links	5,284,457	48,709,772	310,876,909	Person to Person
Orkut-links[39]	3,072,626	116,586,585	621,963,073	Person to Person
Soc-LiveJournal	4,847,571	42,851,237	285,730,264	Person to Person
Web-EDU	9,845,725	46,236,104	254,718,147	Web Graph (page to page)
Web-Google	875,713	3,852,985	11,385,529	Web Graph
Wikipedia 2005/11	1,634,989	18,540,589	44,667,095	Web Graph (page to page)
Wikipedia 2006/9	2,983,494	35,048,115	84,018,183	Web Graph (page to page)
Wikipedia 2006/11	3,148,440	37,043,456	88,823,817	Web Graph (page to page)
Wikipedia 2007/2	3,566,907	42,375,911	102,434,918	Web Graph (page to page)
Youtube[39]	1,157,822	2,990,442	4,945,382	Person to Person

Performance without Sparsification

Graph	**Exact**		**Simple**		**Hybrid**	
	err(%)	time	err(%)	time	err(%)	time
AS-Skitter	0.000	4.452	1.308	0.746	0.128	1.204
Flickr	0.000	41.981	0.166	1.049	0.128	2.016
Livejournal-links	0.000	50.828	0.309	2.998	0.116	9.375
Orkut-links	0.000	202.012	0.564	6.208	0.286	21.328
Soc-LiveJournal	0.000	38.271	0.285	2.619	0.108	7.451
Web-EDU	0.000	8.502	0.157	2.631	0.047	3.300
Web-Google	0.000	1.599	0.286	0.379	0.045	0.740
Wiki-2005	0.000	32.472	0.976	1.197	0.318	3.613
Wiki-2006/9	0.000	86.623	0.886	2.250	0.361	7.483
Wiki-2006/11	0.000	96.114	1.915	2.362	0.530	7.972
Wiki-2007	0.000	122.395	0.943	2.728	0.178	9.268
Youtube	0.000	1.347	1.114	0.333	0.127	0.500

Performance with Sparsification ($p = 0.1$)

Graph	**Exact**		**Simple**		**Hybrid**	
	err(%)	time	err(%)	time	err(%)	time
AS-Skitter	2.188	0.641	3.208	0.651	1.388	0.877
Flickr	0.530	1.389	0.746	0.860	0.818	1.033
Livejournal-links	0.242	3.900	0.628	2.518	1.011	3.475
Orkut-links	0.172	9.881	1.980	5.322	0.761	7.227
Soc-LiveJournal	0.681	3.493	0.830	2.222	0.462	2.962
Web-EDU	0.571	2.864	0.771	2.354	0.383	2.732
Web-Google	1.112	0.251	1.262	0.371	0.264	0.265
Wiki-2005	1.249	1.529	7.498	1.025	0.695	1.313
Wiki-2006/9	0.402	3.431	6.209	1.843	2.091	2.598
Wiki-2006/11	0.634	3.578	4.050	1.947	0.950	2.778
Wiki-2007	0.819	4.407	3.099	2.224	1.448	3.196
Youtube	1.358	0.210	5.511	0.302	1.836	0.268

Colorful triangle counting – Pagh-Tsourakakis

$f : V \to [C]$

Require: Unweighted graph $G([n], E)$
Require: Number of colors $C = 1/p$
  Let $f : V \to [C]$ have uniformly random values
  $E' \leftarrow \{\{u, v\} \in E \mid f(u) = f(v)\}$
  $T \leftarrow$ number of triangles in the graph (V, E')
  return T/p^2

We can guarantee whp high-quality approximations when

$$\frac{1}{C} = p \geq \max\left\{\frac{\tilde{O}(\Delta)}{t}, \frac{\tilde{O}(1)}{t^{1/2}}\right\}$$

The Beauty of the Triangle Query

- Simple to state: `SELECT * FROM G WHERE (u,v), (v,w), (w,u)` $\in$ `E`
- Rich in algorithmic depth: from algebraic to combinatorial, from exact to approximate
- Foundational for graph mining: clustering, transitivity, motifs, community structure
- drives theory and systems (database systems, streaming, MapReduce etc.) and sparks innovations across domains: social networks, biology, web graphs, infrastructure

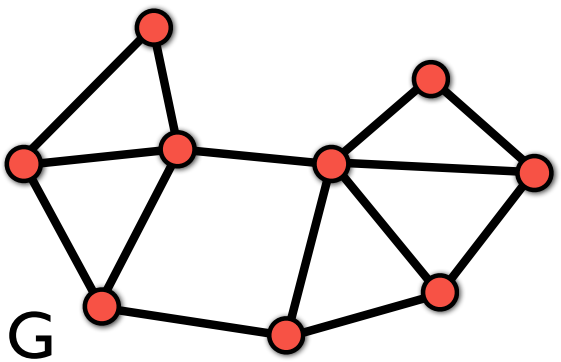
Key refs I discussed:
[Tsourakakis, 2008, Tsourakakis et al., 2009, Tsourakakis et al., 2011, Kolountzakis et al., 2012, Pagh and Tsourakakis, 2012, Alon et al., 1997]

Cluster detection

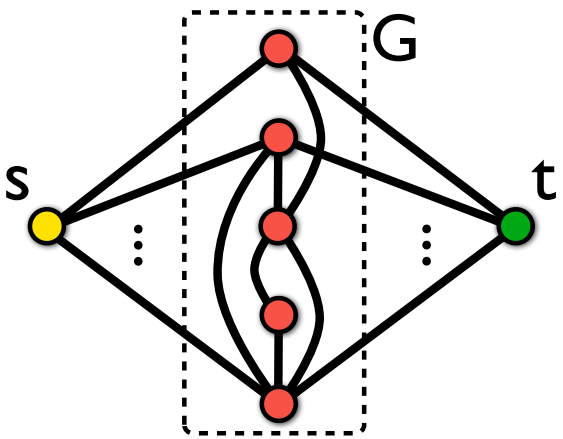
A) Dense clusters
B) Community detection

Goldberg's algorithm for densest subgraph

- consider first degree density d



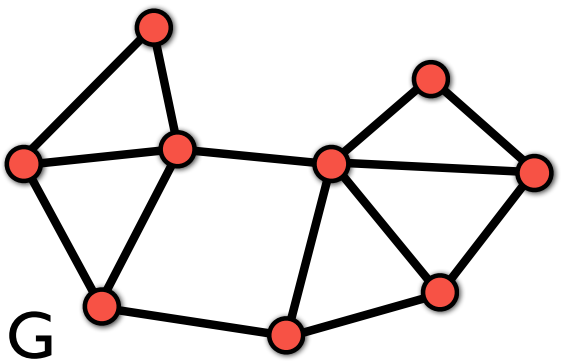
- is there a subgraph S with $d(S) \geq c$?
- transform to a min-cut instance



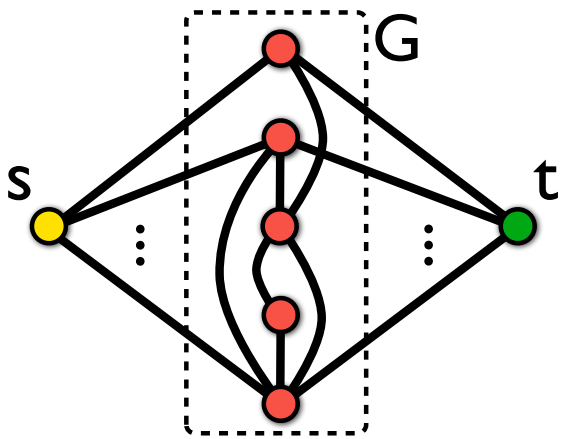
- on the transformed instance:
- is there a cut smaller than a certain value?

Goldberg's algorithm for densest subgraph

- consider first degree density d



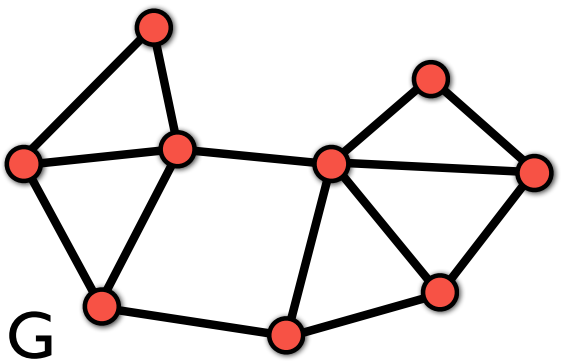
- is there a subgraph S with $d(S) \geq c$?
- transform to a min-cut instance



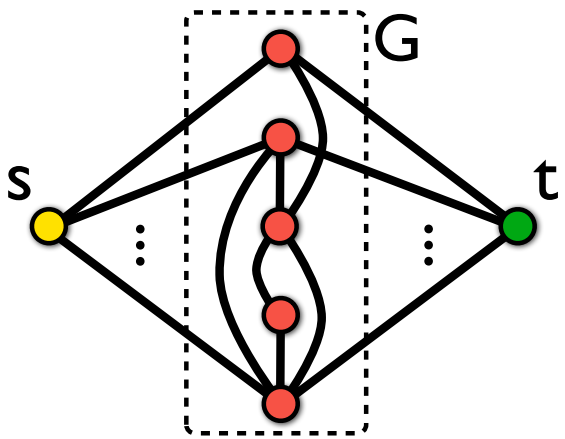
- on the transformed instance:
- is there a cut smaller than a certain value?

Goldberg's algorithm for densest subgraph

- consider first degree density d



- is there a subgraph S with $d(S) \geq c$?
- transform to a min-cut instance



- on the transformed instance:
- is there a cut smaller than a certain value?

Goldberg's algorithm for densest subgraph

is there S with $d(S) \geq c$?

$$\frac{2\,|E(S,S)|}{|S|} \geq c$$

$$2\,|E(S,S)| \geq c|S|$$

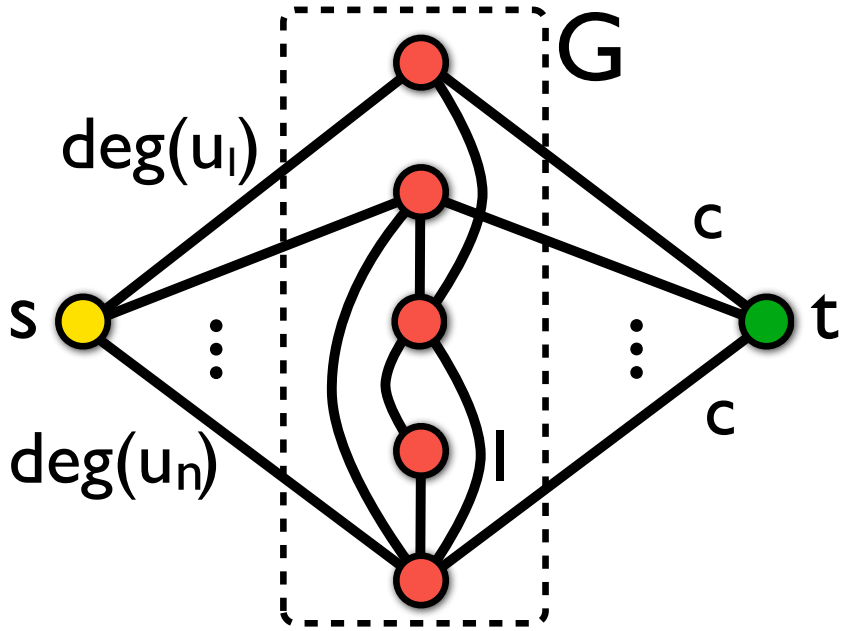
$$\sum_{u \in S} \deg(u) - |E(S,\bar{S})| \geq c|S|$$

$$\sum_{u \in S} \deg(u) + \sum_{u \in \bar{S}} \deg(u) - \sum_{u \in \bar{S}} \deg(u) - |E(S,\bar{S})| \geq c|S|$$

$$\sum_{u \in \bar{S}} \deg(u) + |E(S,\bar{S})| + c|S| \leq 2\,|E|$$

Goldberg's algorithm for densest subgraph

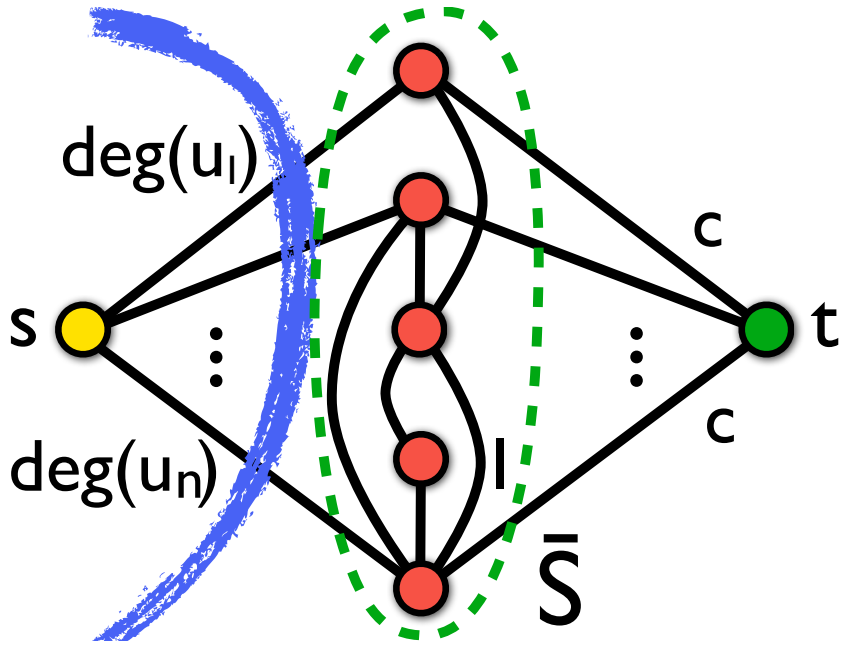
- transformation to min-cut instance



- is there S s.t. $\sum_{u \in \bar{S}} \deg(u) + |e(S, \bar{S})| + c|S| \leq 2|E|$?

Goldberg's algorithm for densest subgraph

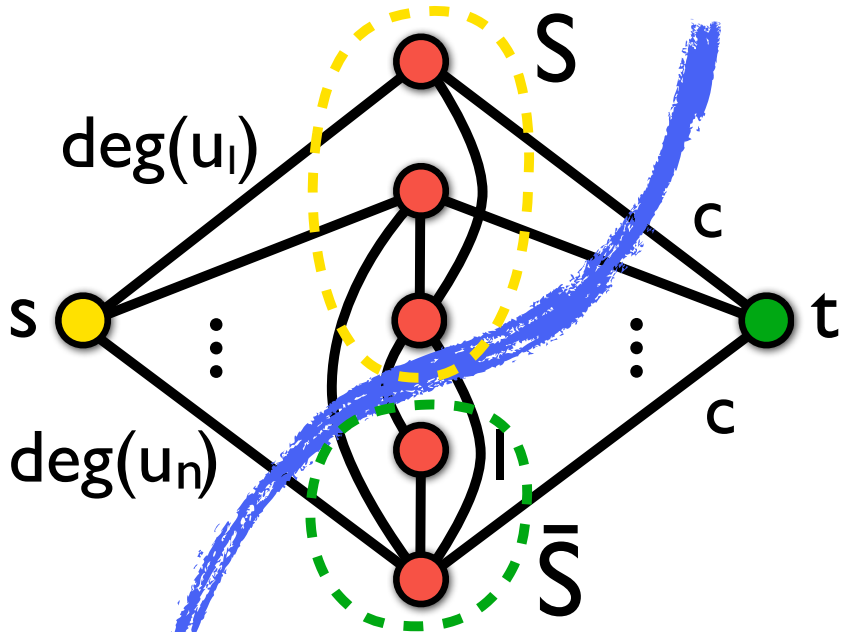
- transform to a min-cut instance



- is there S s.t. $\sum_{u \in \bar{S}} \deg(u) + |e(S, \bar{S})| + c|S| \leq 2|E|$?
- a cut of value $2|E|$ always exists, for $S = \emptyset$

Goldberg's algorithm for densest subgraph

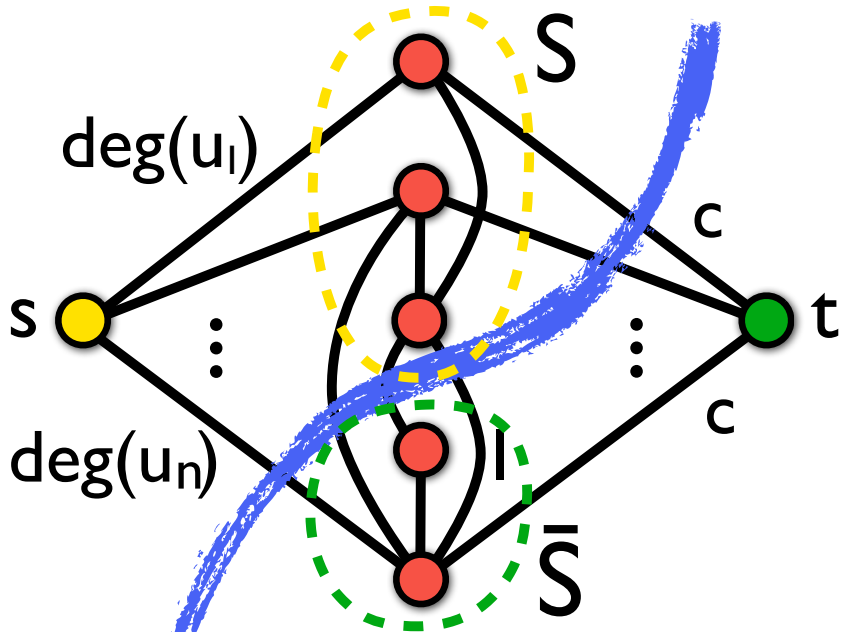
- transform to a min-cut instance



- is there S s.t. $\sum_{u \in \bar{S}} \deg(u) + |e(S, \bar{S})| + c|S| \leq 2|E|$?
- $S \neq \emptyset$ gives cut of value $\sum_{u \in \bar{S}} \deg(u) + |e(S, \bar{S})| + c|S|$

Goldberg's algorithm for densest subgraph

- transform to a min-cut instance



- is there S s.t. $\sum_{u \in \bar{S}} \deg(u) + |e(S, \bar{S})| + c|S| \leq 2|E|$?
- YES, if min cut achieved for $S \neq \emptyset$

Goldberg's algorithm for densest subgraph

[Goldberg, 1984]

input: undirected graph $G = (V, E)$, number c
output: S, if $d(S) \geq c$

1 transform G into min-cut instance $G' = (V \cup \{s\} \cup \{t\}, E', w')$
2 find min cut $\{s\} \cup S$ on G'
3 if $S \neq \emptyset$ return S
4 else return NO

- to find the densest subgraph perform binary search on c
- logarithmic number of min-cut calls
- problem can also be solved with one min-cut call using the parametric max-flow algorithm

densest subgraph problem – discussion

- Goldberg's algorithm polynomial algorithm, but
- $\mathcal{O}(nm)$ time for one min-cut computation
- not scalable for large graphs (millions of vertices / edges)

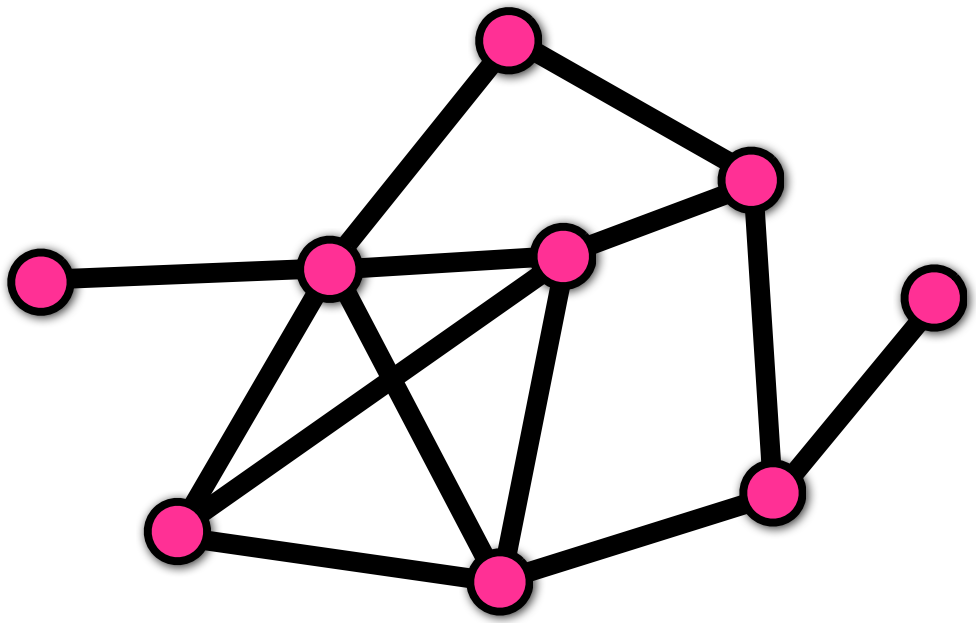
- faster algorithm due to [Charikar, 2000]
- greedy and simple to implement
- approximation algorithm

densest subgraph problem – discussion

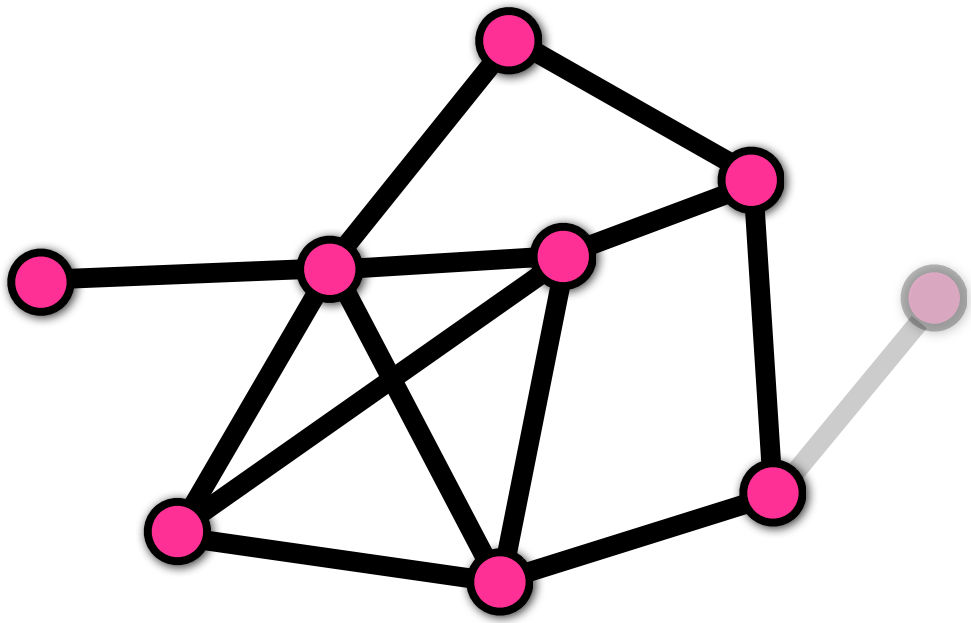
- Goldberg's algorithm polynomial algorithm, but
- $\mathcal{O}(nm)$ time for one min-cut computation
- not scalable for large graphs (millions of vertices / edges)

- faster algorithm due to [Charikar, 2000]
- greedy and simple to implement
- approximation algorithm

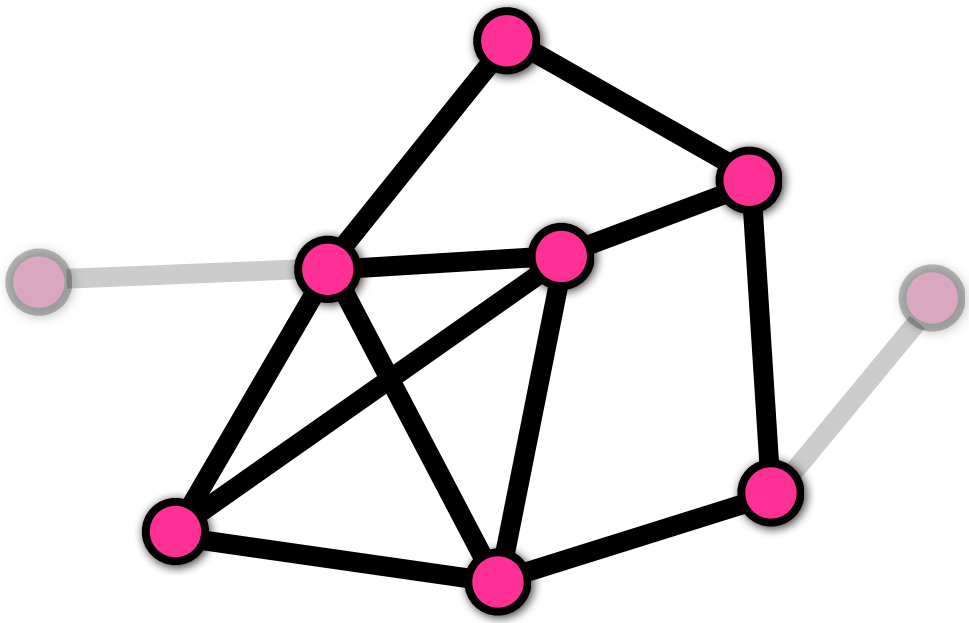
greedy algorithm for densest subgraph — example



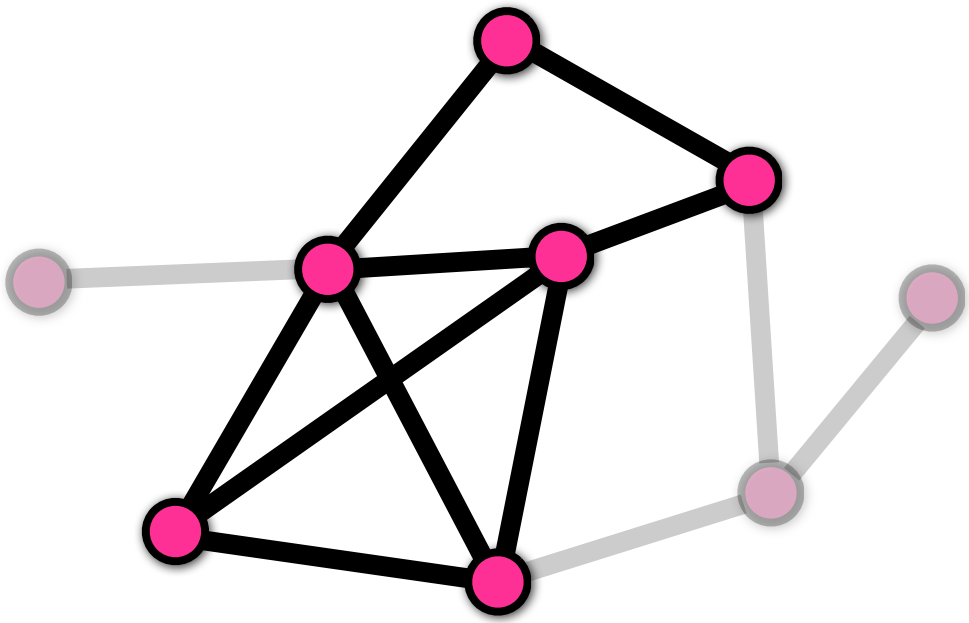
greedy algorithm for densest subgraph — example



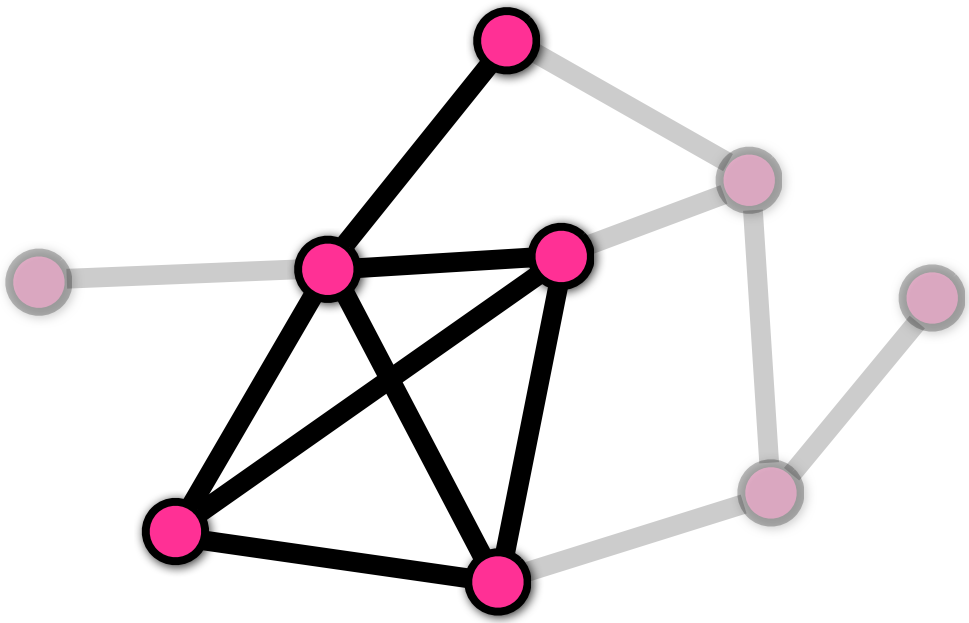
greedy algorithm for densest subgraph — example



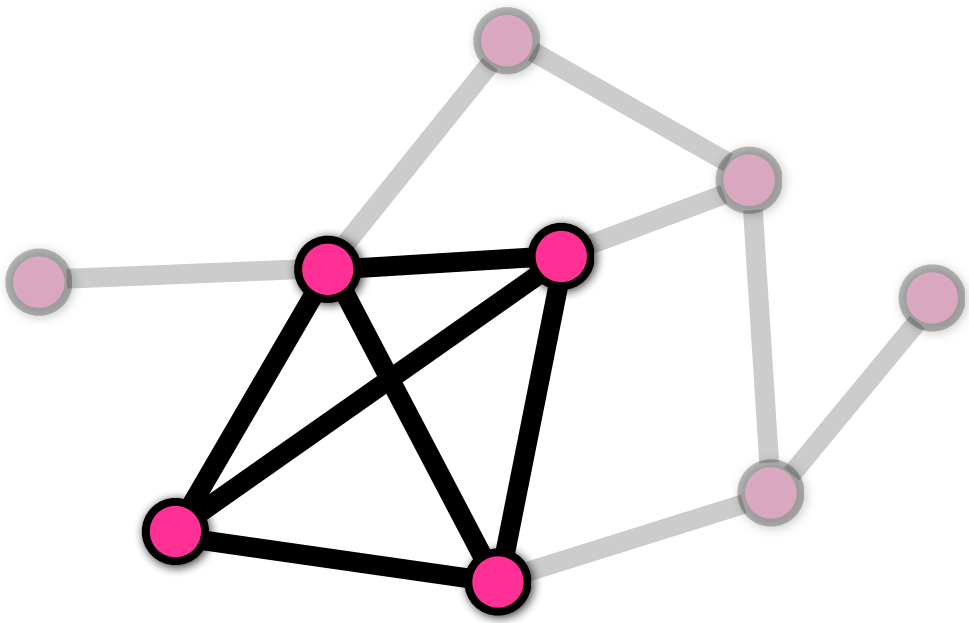
greedy algorithm for densest subgraph — example



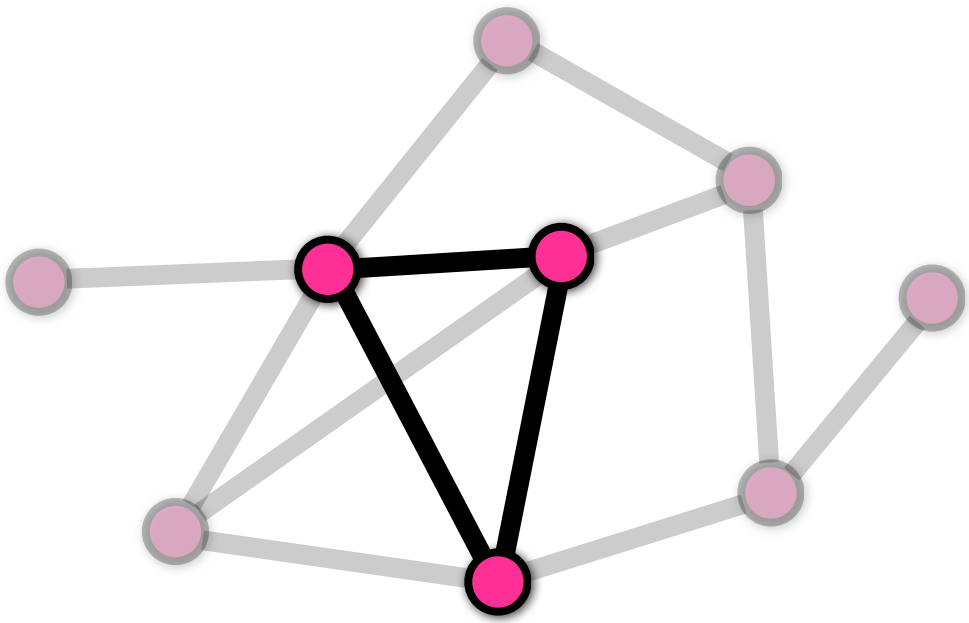
greedy algorithm for densest subgraph — example



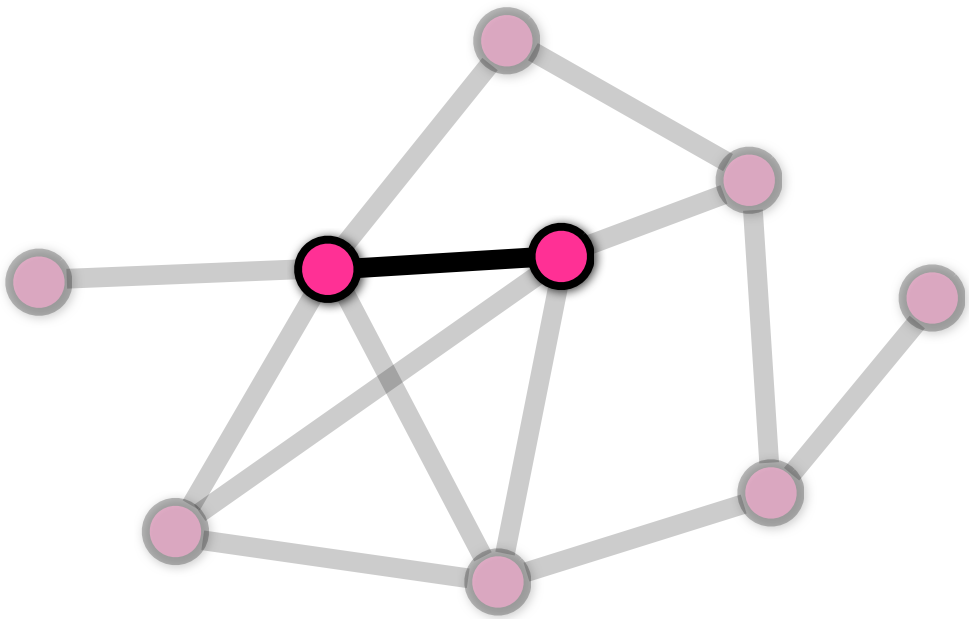
greedy algorithm for densest subgraph — example



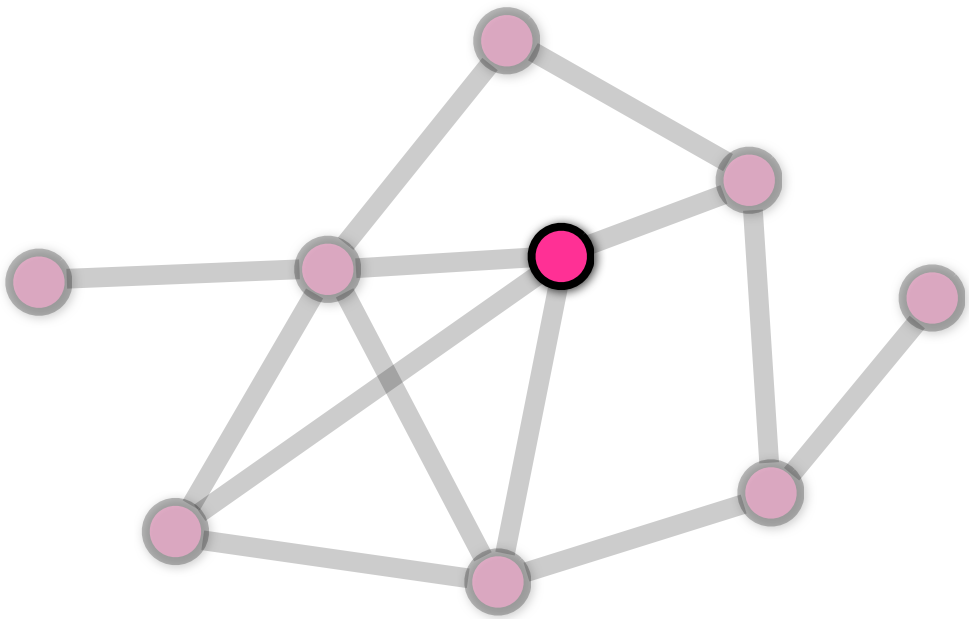
greedy algorithm for densest subgraph — example



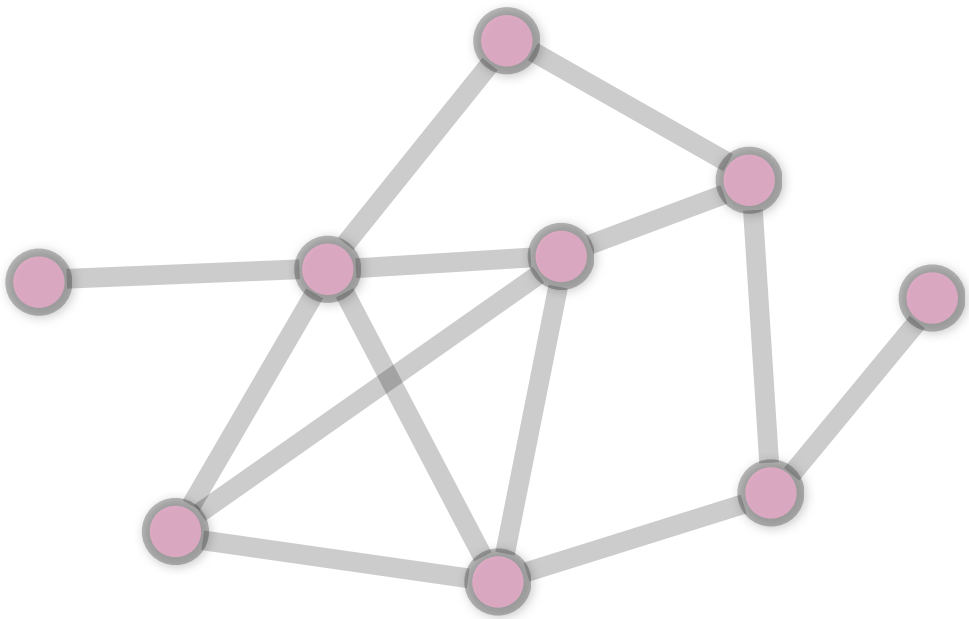
greedy algorithm for densest subgraph — example



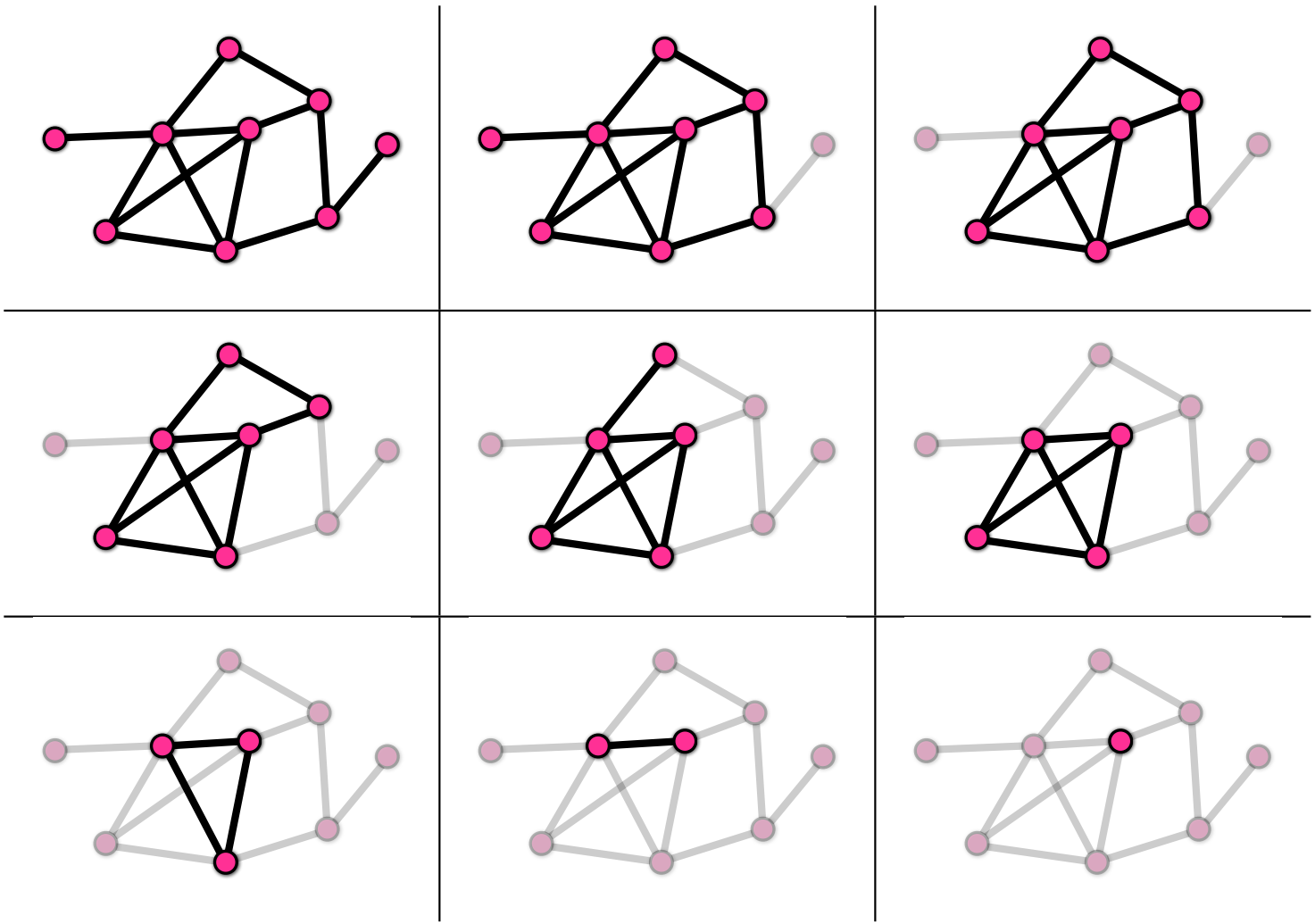
greedy algorithm for densest subgraph — example



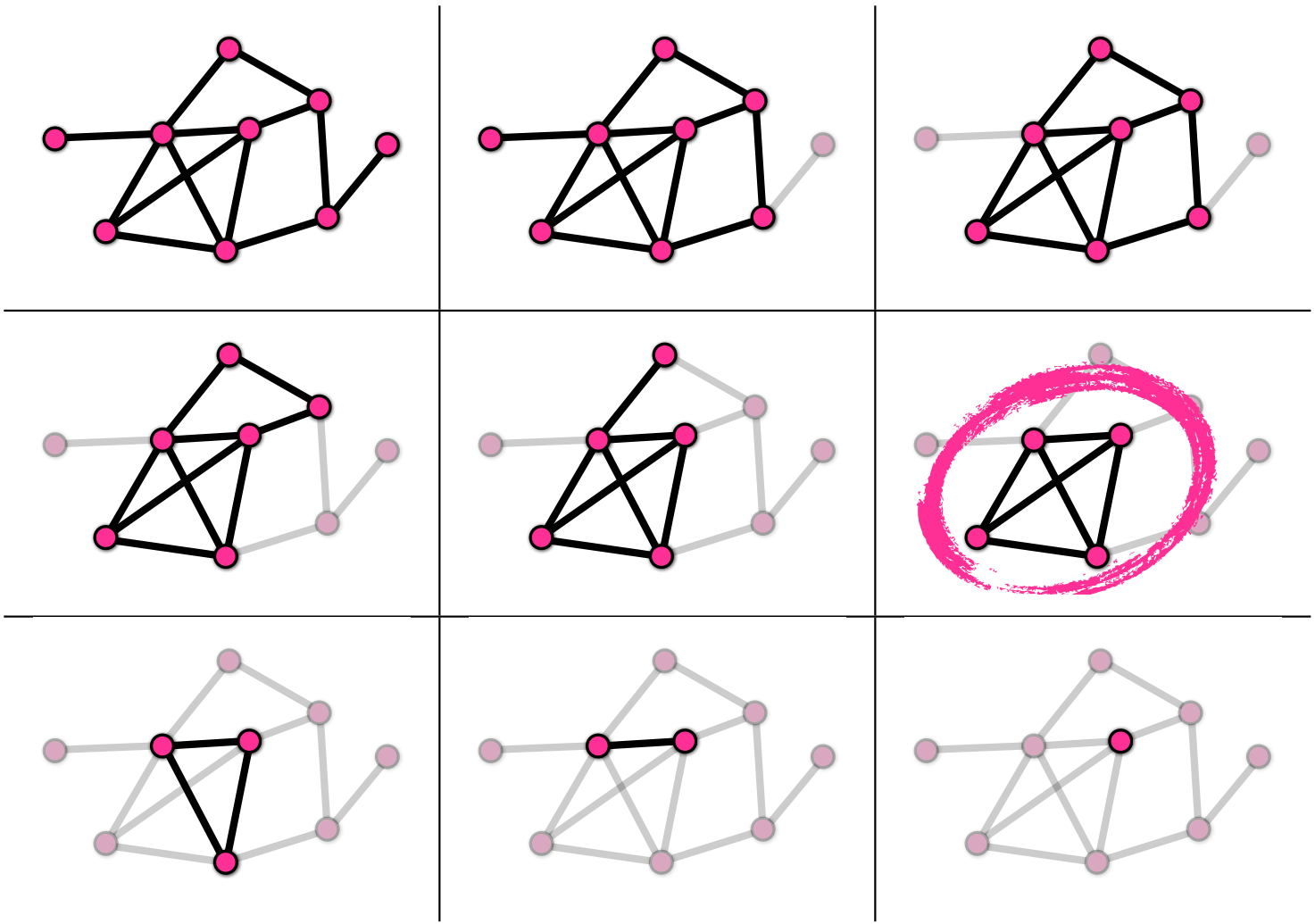
greedy algorithm for densest subgraph — example



greedy algorithm for densest subgraph — example



greedy algorithm for densest subgraph — example



greedy algorithm for densest subgraph

[Charikar, 2000]

input: undirected graph $G = (V, E)$
output: S, a dense subgraph of G

1 set $G_n \leftarrow G$
2 for $k \leftarrow n$ downto 1
2.1 let v be the smallest degree vertex in G_k
2.2 $G_{k-1} \leftarrow G_k \setminus \{v\}$
3 output the densest subgraph among $G_n, G_{n-1}, \ldots, G_1$

Theorem

Charikar's algorithm is a $\frac{1}{2}$-approximation algorithm for DSP.

LP formulation

$$\textsc{Primal}(G)$$

$$\begin{aligned}
\text{maximize} \quad & \sum_{e \in E} y_e \\
\text{subject to} \quad & y_e \leq x_u, x_v, && \forall e = uv \in E \\
& \sum_{v \in V} x_v \leq 1, \\
& y_e \geq 0, x_v \geq 0, && \forall e \in E, \forall v \in V
\end{aligned}$$

Theorem [Charikar '00]: OPT of this LP $= \rho^*_G$

LP formulation

$$\textsc{Primal}(G)$$

$$\begin{aligned}
\text{maximize} \quad & \sum_{e \in E} y_e \\
\text{subject to} \quad & y_e \leq x_u, x_v, && \forall e = uv \in E \\
& \sum_{v \in V} x_v \leq 1, \\
& y_e \geq 0, x_v \geq 0, && \forall e \in E, \forall v \in V
\end{aligned}$$

Theorem [Charikar '00]: OPT of this LP $= \rho^*_G$

Dual of the LP

$\textsc{Primal}(G)$

$$\begin{aligned} \max \quad & \sum_{e \in E} y_e \\ \text{s. to} \quad & y_e \leq x_u, x_v, \\ & \sum_{v \in V} x_v \leq 1, \\ & y_e \geq 0, x_v \geq 0, \end{aligned}$$

$\textsc{Dual}(G)$

$$\begin{aligned} \min \quad & D \\ \text{s. to} \quad & f_e(u) + f_e(v) \geq 1, \\ & \sum_{e \ni v} f_e(v) \leq D, \\ & f_e(v) \geq 0, \end{aligned}$$

Dual of the LP

$\textsc{Primal}(G)$

$$\begin{aligned} \max \quad & \sum_{e \in E} y_e \\ \text{s. to} \quad & y_e \leq x_u, x_v, \\ & \sum_{v \in V} x_v \leq 1, \\ & y_e \geq 0, x_v \geq 0, \end{aligned}$$

$\textsc{Dual}(G)$

$$\begin{aligned} \min \quad & D \\ \text{s. to} \quad & f_e(u) + f_e(v) \geq 1, \\ & \sum_{e \ni v} f_e(v) \leq D, \\ & f_e(v) \geq 0, \end{aligned}$$

Dual LP

$\textsc{Dual}(G)$

$$\begin{array}{lll} \text{minimize} & D & \\ \text{subject to} & f_e(u)+f_e(v) \geq 1, & \forall e = uv \in E \\ & \sum_{e \ni v} f_e(v) \leq D, & \forall v \in V \\ & f_e(u) \geq 0, f_e(v) \geq 0, & \forall e = uv \in E \end{array}$$

Find the smallest D such that:

$$f_e(u)+f_e(v) = 1$$
$$\sum_{e \ni v} f_e(v) \leq D$$

Dual LP

$$\textsc{Dual}(G)$$

$$\begin{array}{lll} \text{minimize} & D & \\ \text{subject to} & f_e(u)+f_e(v) \geq 1, & \forall e = uv \in E \\ & \sum_{e \ni v} f_e(v) \leq D, & \forall v \in V \\ & f_e(u) \geq 0, f_e(v) \geq 0, & \forall e = uv \in E \end{array}$$

Find the smallest D such that:

$$f_e(u)+f_e(v) = 1$$
$$\sum_{e \ni v} f_e(v) \leq D$$

Alternate visualization of dual

- Each edge has load $= 1$
- Assign this load fractionally to both end points
- **Minimize maximum load**

Optimal assignment:

- assign be to e; df to f
- all other edges equally to end points
- max load $= 1.5$

Alternate visualization of dual

- Each edge has load $= 1$
- Assign this load fractionally to both end points
- **Minimize maximum load**

Optimal assignment:

- assign be to e; df to f
- all other edges equally to end points
- max load $= 1.5$

Alternate visualization of dual

- Each edge has load $= 1$
- Assign this load fractionally to both end points
- **Minimize maximum load**

Optimal assignment:

- assign be to e; df to f
- all other edges equally to end points
- max load $= 1.5$

Alternate visualization of dual

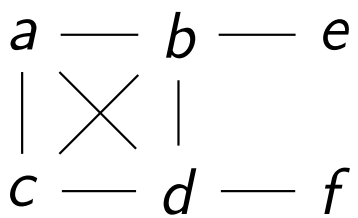
- Each edge has load $= 1$
- Assign this load fractionally to both end points
- **Minimize maximum load**

Optimal assignment:

- assign be to e; df to f
- all other edges equally to end points
- max load $= 1.5$

Alternate visualization of dual

- Each edge has load $= 1$
- Assign this load fractionally to both end points
- **Minimize maximum load**

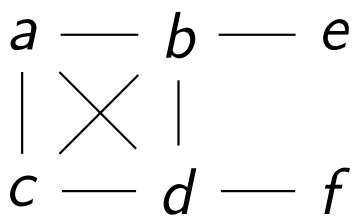


Optimal assignment:

- assign be to e; df to f
- all other edges equally to end points
- max load $= 1.5$

Alternate visualization of dual

- Each edge has load $= 1$
- Assign this load fractionally to both end points
- **Minimize maximum load**

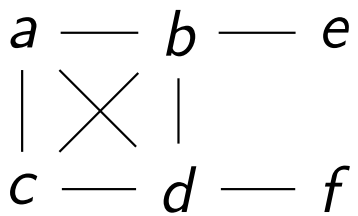


Optimal assignment:

- assign be to e; df to f
- all other edges equally to end points
- max load $= 1.5$

Alternate visualization of dual

- Each edge has load $= 1$
- Assign this load fractionally to both end points
- **Minimize maximum load**

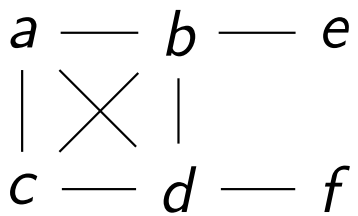


Optimal assignment:

- assign be to e; df to f
- all other edges equally to end points
- max load $= 1.5$

Alternate visualization of dual

- Each edge has load $= 1$
- Assign this load fractionally to both end points
- **Minimize maximum load**



Optimal assignment:

- assign be to e; df to f
- all other edges equally to end points
- max load $= 1.5$

Greedy++ - our algorithm

1. Run several iterations of greedy
2. First iteration $\rightarrow$ regular greedy algorithm
3. Update vertex degrees $\leftarrow$ degree **+ load assigned previously**
4. Run greedy with new "degrees" and repeat

Gives a far more balanced dual solution (recall: our aim is to minimize max load).

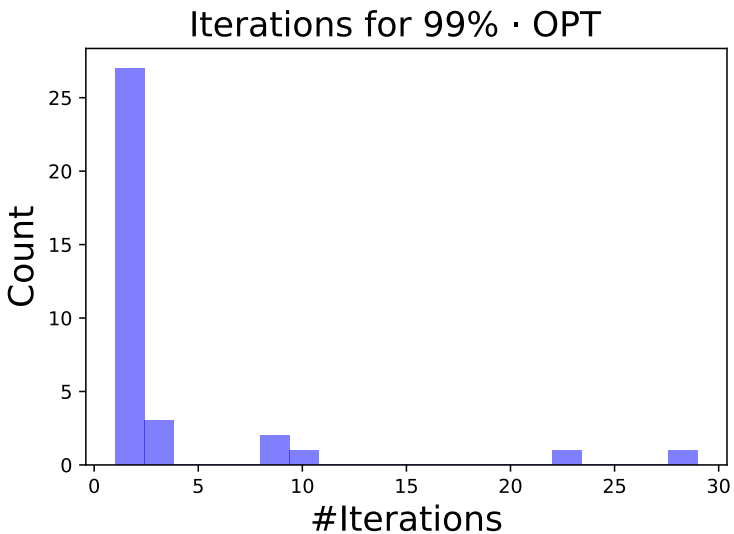
Greedy++ pseudocode – Input $G(V, E), T$

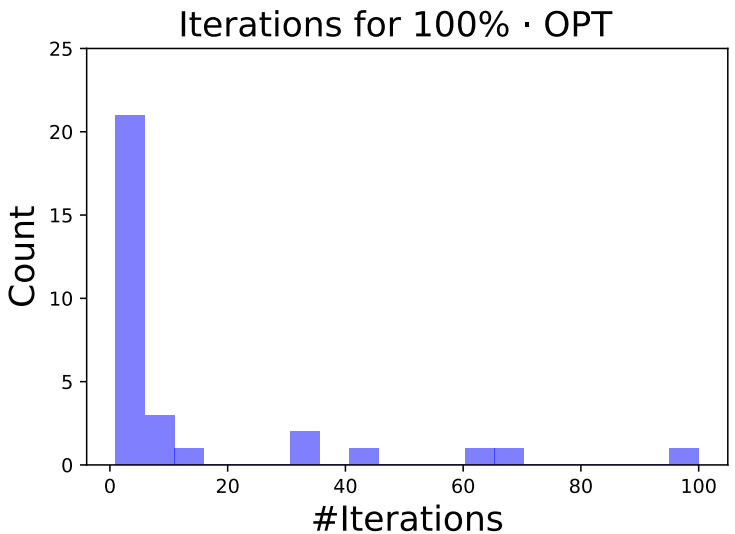
$G_{\text{densest}} \leftarrow G$
Initialize the vertex load vector $\ell^{(0)} \leftarrow 0 \in \mathbb{Z}^n$;
for $i : 1 \to T$ **do**
  $H \leftarrow G$;
  while $H \neq \emptyset$ **do**
    Find the vertex $u \in H$ with minimum $\ell_u^{(i-1)} + \deg_H(u)$;
    $\ell_u^{(i)} \leftarrow \ell_u^{(i-1)} + \deg_H(u)$;
    Remove u and all its adjacent edges uv from H;
    if $\rho(H) > \rho(G_{\text{densest}})$ **then**
      $G_{\text{densest}} \leftarrow H$
    end if
  end while
end for
Return G_{densest}

Large collection of datasets

Name	n	m
web-trackers [23]	40 421 974	140 613 762
orkut [23]	3 072 441	117 184 899
livejournal-affiliations [23]	10 690 276	112 307 385
wiki-topcats	1 791 489	25 447 873
cit-Patents	3 774 768	16 518 948
actor-collaborations [23]	382 219	15 038 083
ego-gplus	107 614	12 238 285
dblp-author	5 425 963	8 649 016
web-BerkStan	685 230	6 649 470
flickr [37]	80 513	5 899 882
wiki-Talk	2 394 385	4 659 565
web-Google	875 713	4 322 051
com-youtube	1 134 890	2 987 624
roadNet-CA	1 965 206	2 766 607
web-Stanford	281 903	1 992 636
roadNet-TX	1 379 917	1 921 660
roadNet-PA	1 088 092	1 54 898
Ego-twitter	81 306	1 342 296
com-dblp	317 080	1 049 866
com-Amazon	334 863	925 872
soc-slashdot0902	82 168	504 230
soc-slashdot0811	77 360	469 180
soc-Epinions	75 879	405 740
blogcatalog [37]	10,312	333 983
email-Enron	36 692	183 831
ego-facebook	4 039	88 234
ppi [31]	3 890	37 845
twitter-retweet [30]	316 662	1 122 070
twitter-favorite [30]	226 516	1 210 041
twitter-mention [30]	571 157	1 895 094
twitter-reply [30]	196 697	296 194
soc-sign-slashdot081106	77 350	468 554
soc-sign-slashdot090216	81 867	497 672
soc-sign-slashdot090221	82 140	500 481
soc-sign-epinions	131 828	711 210

Number of iterations T required to obtain f% accuracy





- Histograms of number of iterations to reach 99% of the optimum degree density (left), and the optimum degree density (right)
- **Remark**: typically, within less than 5 iterations we reach a near-optimal solution.

Experimental results

Convergence of Greedy++ on some large networks:

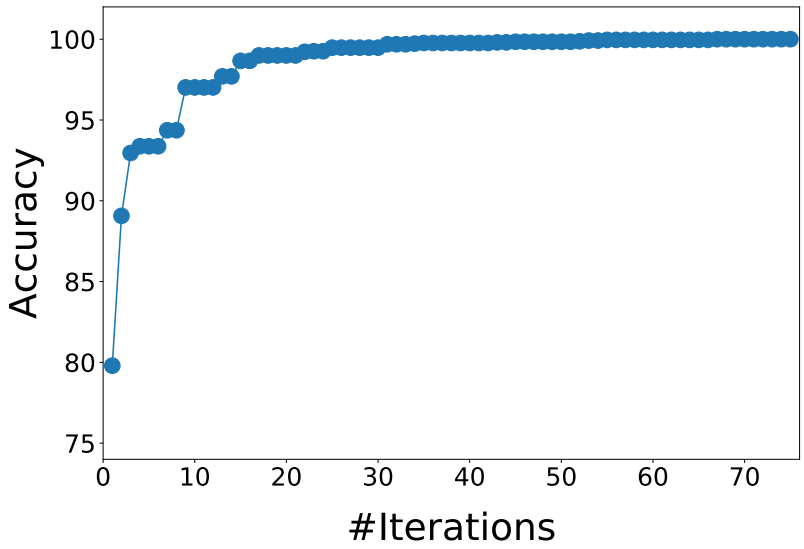


Figure: Amazon co-purchasing network

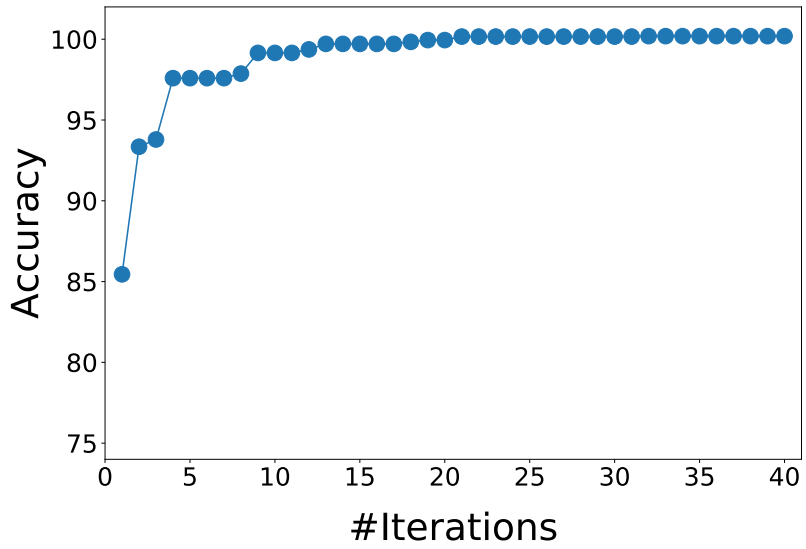


Figure: California road connection network

Experimental results

Scalability and speed of Greedy++:

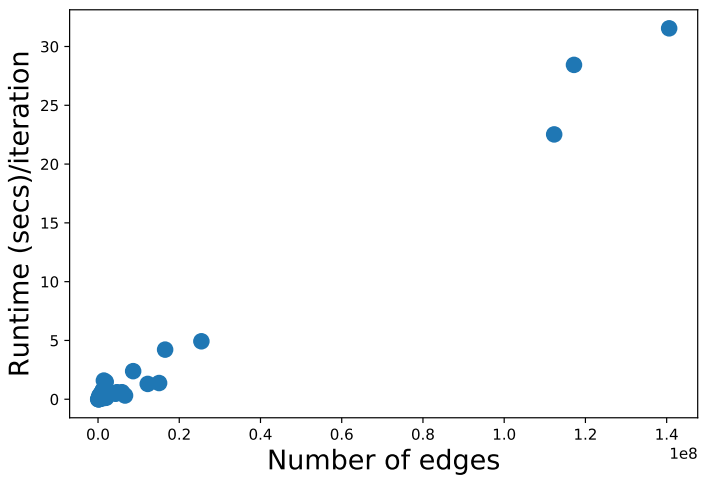


Figure: Runtime in seconds per iteration

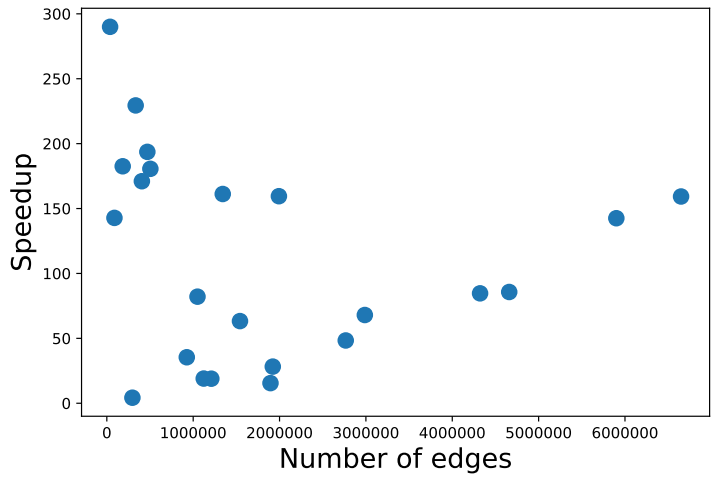


Figure: Speedup over exact algorithm (max-flow based)

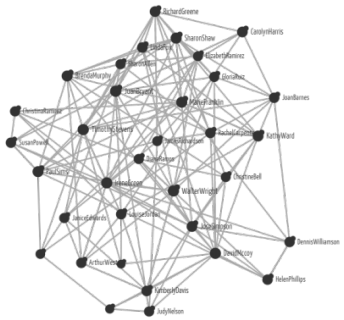
Some additional remarks

1. When we are able to run the exact algorithm (for graphs with more than 8M edges, the maximum flow code crashes) on our machine, the average speedup that our algorithm provides to reach *the optimum* is 144.6× on average, with a standard deviation equal to 57.4. The smallest speedup observed was 67.9×, and the largest speedup 290×.
2. The maximum number of iterations needed to reach 90% of the optimum is at most 3, i.e., by running two more passes compared to Charikar's algorithm, we are able to boost the accuracy by 10%.
3. Our algorithm GREEDY++ when given enough number of iterations *always* finds the optimal value, and the densest subgraph.
4. Later, analyzed by [Chekuri et al., 2022]

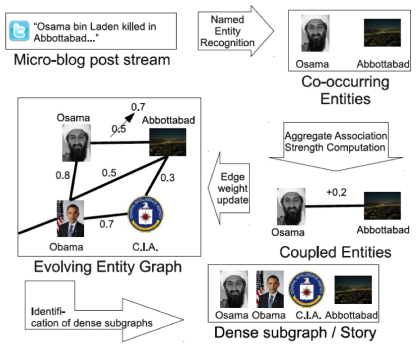
Motivating research question

- despite rich landscape of algorithmic tools, until recently, no polynomial algorithm for finding large near-cliques
- Many important applications!

Who-calls-whom



Twitter



Youtube

	Security	Social Media	Topic Clusters
V	Humans	Entities	Videos
E	Phone call	Co-occurence	Co-watched
Large Near clique	Anomaly	Thematic coherence	Thematic coherence

Some possible formulations

Why not maximize $0 \leq \boxed{f_e(S) = \frac{e(S)}{\binom{|S|}{2}}} \leq 1$

- $\max_{S \subseteq V} f_e(S)$ is **ill-posed** since $f_e($ $) = 1$
- $\max_{S \subseteq V} f_e(S)$ subject to $|S| =, \geq k$ is **NP-hard**

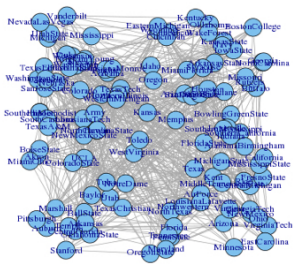
Some possible formulations

Why not maximize $0 \leq \boxed{f_e(S) = \frac{e(S)}{\binom{|S|}{2}}} \leq 1$

- $\max_{S \subseteq V} f_e(S)$ is **ill-posed** since $f_e($ $) = 1$
- $\max_{S \subseteq V} f_e(S)$ subject to $|S| =, \geq k$ is **NP-hard**

densest subgraph problem fails here

Solving the **DSP** typically **does not** result in "clique"-like sets.



- FOOTBALL NETWORK with

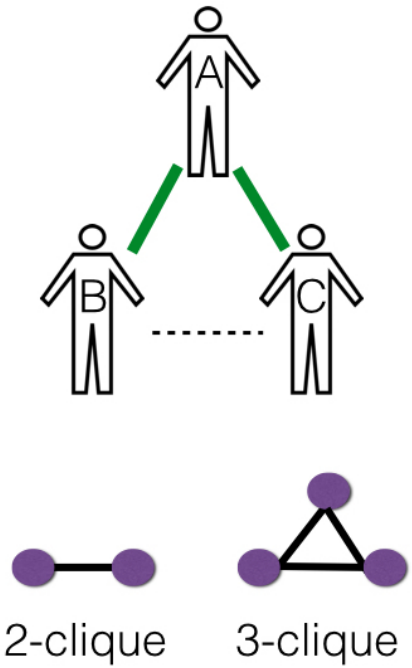
  $n = 115$ vertices $\leftrightarrow$ teams

  $m = 613$ edges $\leftrightarrow$ games

- The optimal solution S^* to the **DSP** is the whole network with resulting degree density $\rho(S^*) = 5.3$ and edge density $f_e(S) = \frac{e(S)}{\binom{|S|}{2}} = 0.094$.

- There exists S' such that $|S'| = 18, f_e(S') = 0.48$. However $\rho(S') = 4.1$.

k-clique densest subgraph problem



E.g. c_2( )=3

For any $S \subseteq V$ let

$$c_k(S) = \#\text{ k-cliques induced by } S.$$

Define ***k*-clique density**

$$\rho_k(S) = \frac{c_k(S)}{s}, k \geq 2, s = |S|$$

Solve the ***k*-clique DSP**

$$\rho_k(S^*) = \rho_k^* = \max_{S \subseteq V} \rho_k(S)$$

Triangle densest subgraph problem

We shall refer to the 3-clique DSP as the *triangle densest subgraph problem*.

$$\max_{S \subseteq V} \tau(S) = \frac{t(S)}{s}$$

.

How different can the densest subgraph be from the triangle densest subgraph? Radically different,e.g., $G = K_{n,n} \cup K_3$.

What happens on real-data? Can we solve the triangle DSP in polynomial time? The k-clique DSP?

Triangle densest subgraph problem

Theorem

There exists an algorithm which solves the TDSP and runs in $O\left(m^{3/2} + nt + \min(n,t)^3\right)$ *time.*

Furthermore,

Theorem

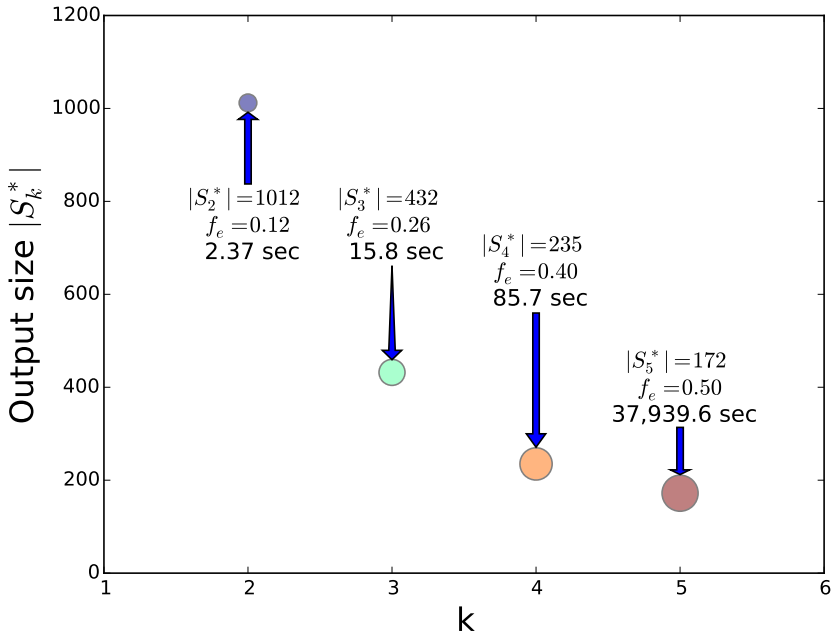
We can solve the k-clique DSP in polynomial time for any $k = \Theta(1)$.

Computation involves

- Enumerate the set C_k of k-cliques in G
- Maximum flow on an appropriate network $\mathcal{N}(\{s,t\} \cup V \cup C_k, A)$

Epinions social network

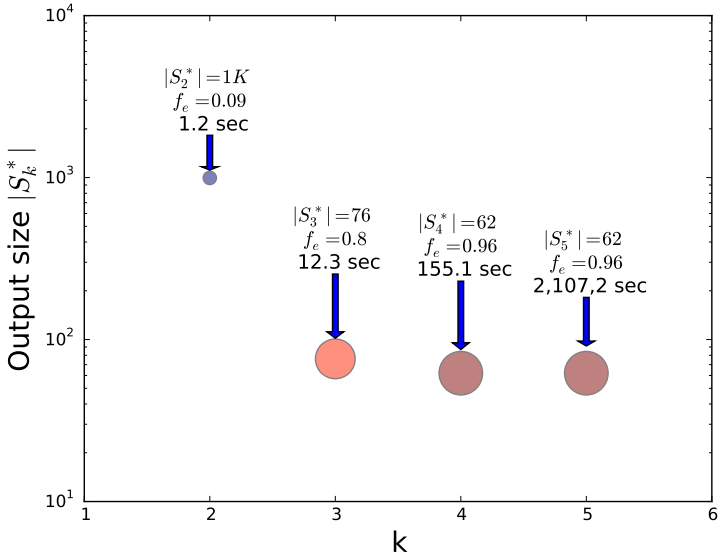
# nodes	75 877
# edges	405 739



k	c_k	T_k (sec)
3-clique	1.6M	1.6
4-clique	5.8M	4.8
5-clique	17.4M	13.4

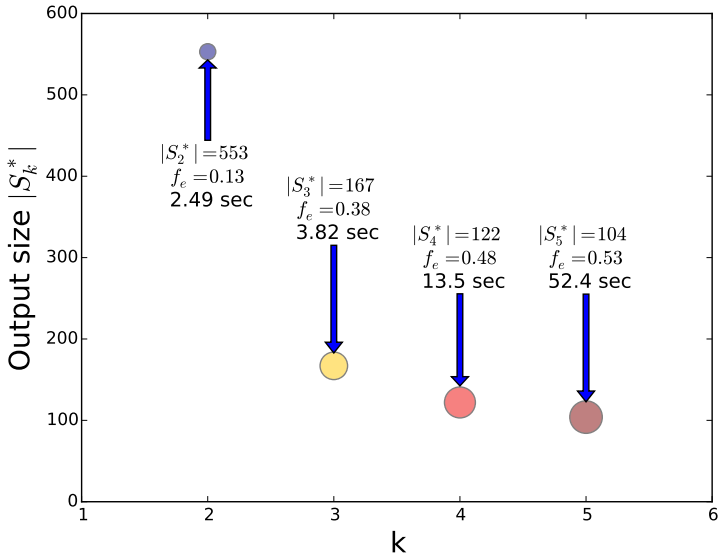
CA-Astro and Email networks

# nodes	18 772
# edges	198 050



k	c_k	T_k (sec)
3-clique	1.4M	0.6
4-clique	9.5M	3.9
5-clique	65M	27.2

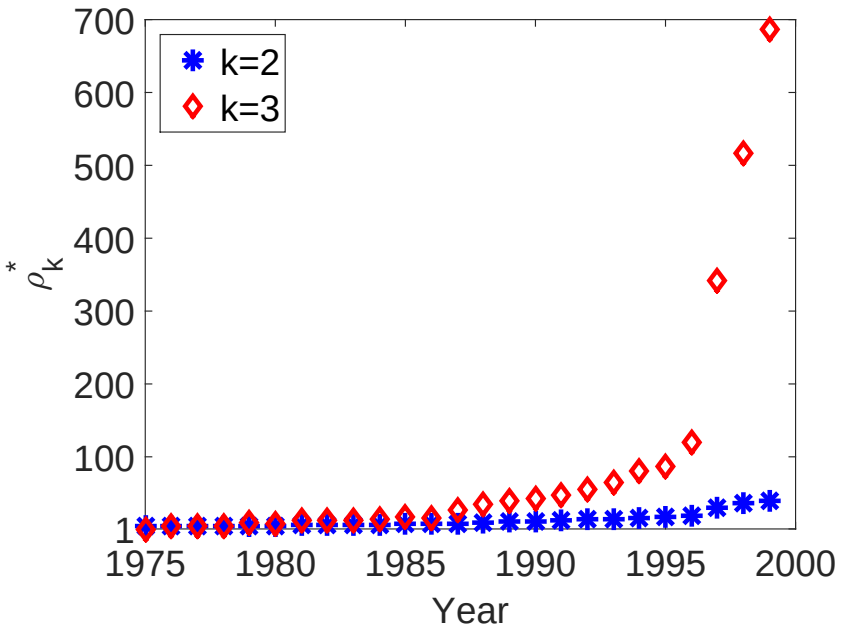
# nodes	234 352
# edges	383 111

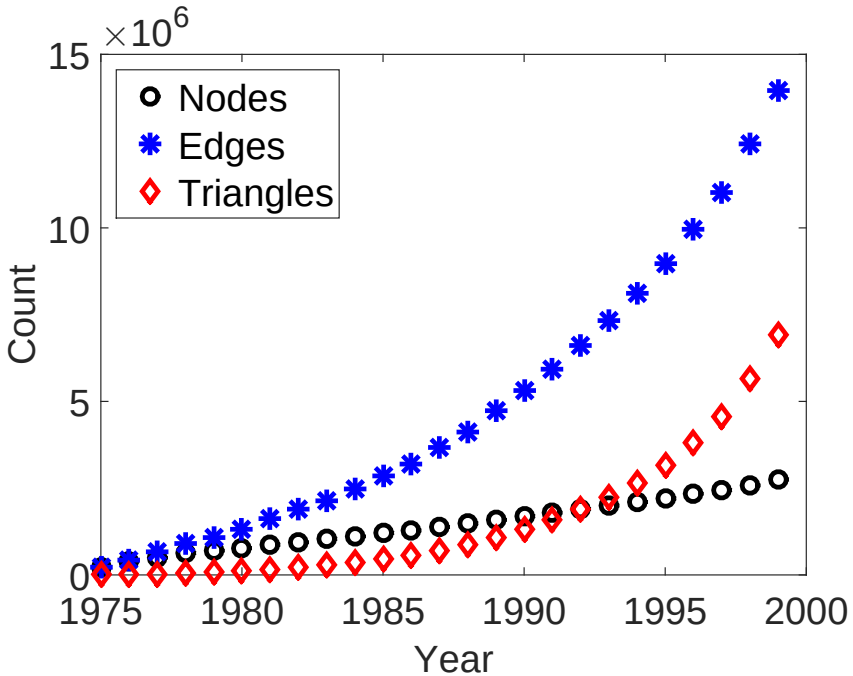


k	c_k	T_k (sec)
3-clique	0.4M	0.4
4-clique	1M	0.9
5-clique	2.6M	1.9

Time evolving networks

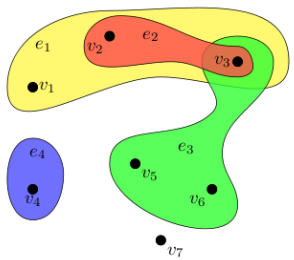
Patents citation network that spans 37 years, specifically from January 1, 1963 to December 30, 1999.





Densest subgraph sparsifiers

Definition: "A hypergraph is a generalization of a graph in which an edge can connect any number of vertices."



Densest subgraph sparsifier theorem

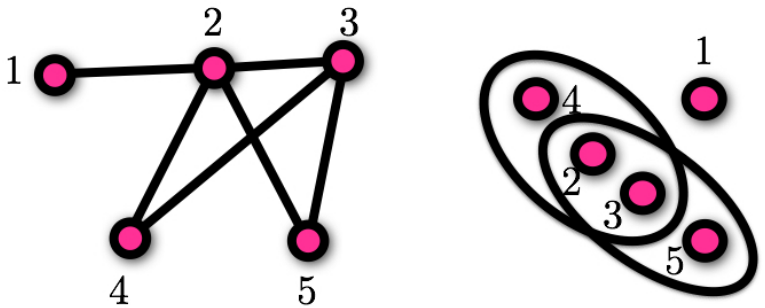
Let $\mathcal{H}(V, E_{\mathcal{H}})$ be a hypergraph, $\epsilon > 0$.

Let $E' \subseteq E_{\mathcal{H}}$ be a sample of $\frac{6n \log n}{\epsilon^2}$ edges chosen uniformly at random.

Solving the DSP on E' results in a $(1 + \epsilon)$ approximation to ρ^* **whp**.

Densest subgraph sparsifiers

Some hypergraphs of interest



Technical difficulty.

Notice that taking Chernoff bounds and a union bound does not work since by Chernoff the failure probability is $1/poly(n)$ whereas there exists an exponential number of potential bad events.

Densest subgraph sparsifiers

Corollary: Single-pass, $(1+\epsilon)$ semi-streaming algorithm! Just keep $O(n \log n/\epsilon^2)$ edges.

Let $p = \frac{6n \log n}{|E_{\mathcal{H}}|\epsilon^2}$.

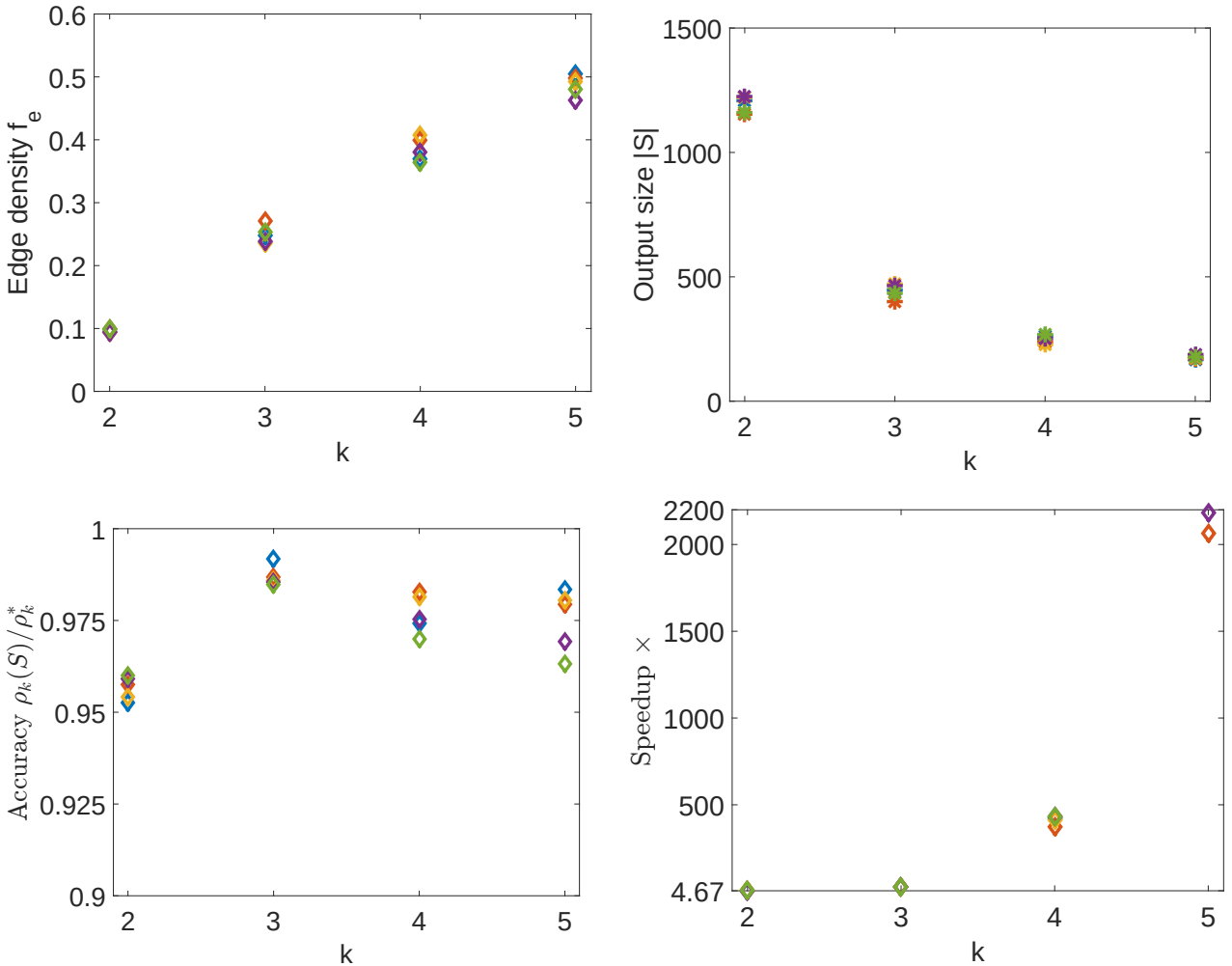
Expected space reduction is $O(\frac{1}{p})$.

Expected speedup for maximum flow computation $O(\frac{1}{p^2})$

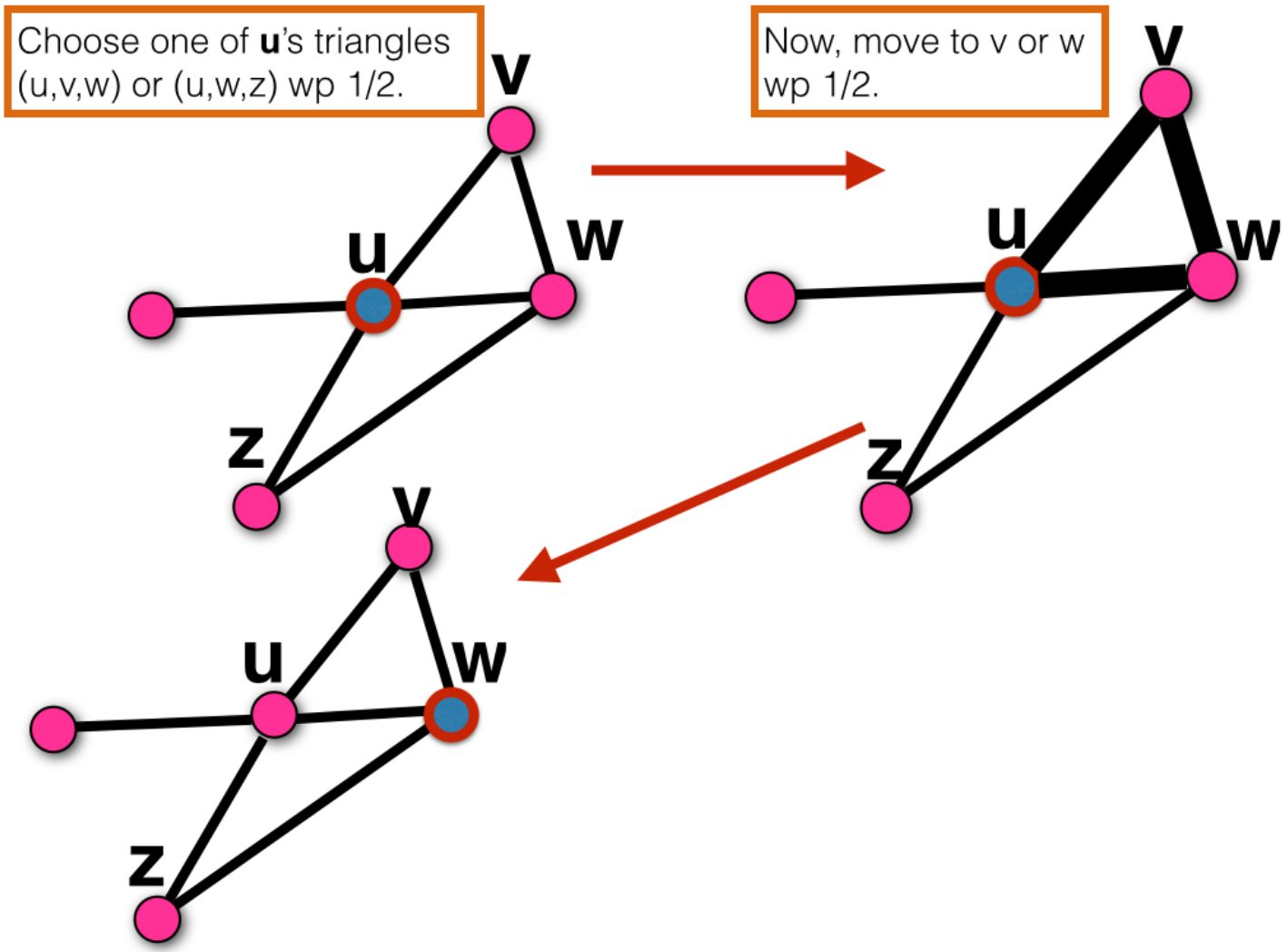
k	Avg. Speedup	Accuracy
2	$3\times$	$\geq 95\%$
3	$23.8\times$	$\geq 98\%$
4	$302\times$	$\geq 99\%$
5	$24\,000\times$	$\approx 100\%$

Zooming-in on Epinions network

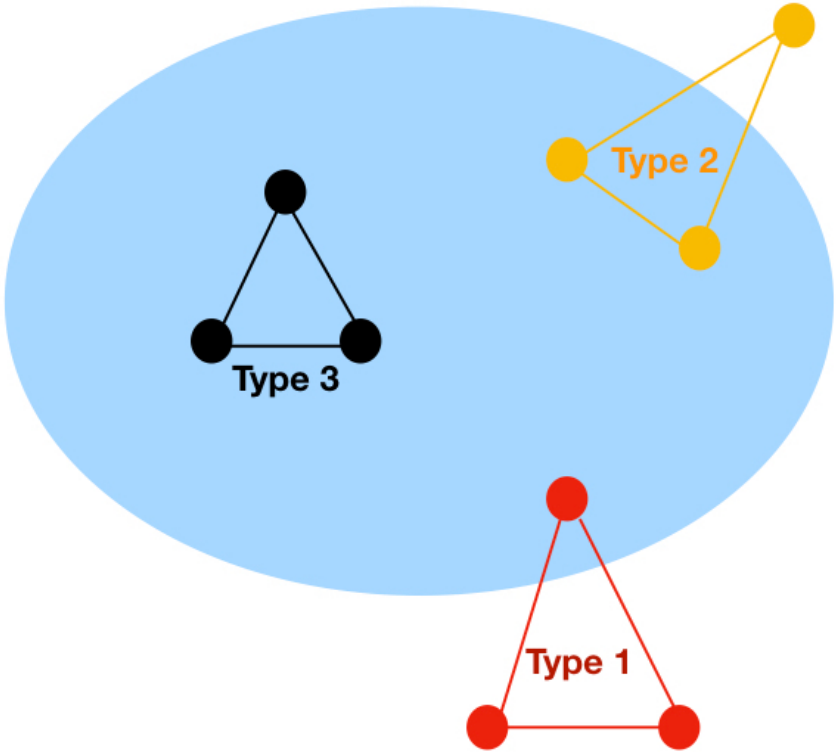
EPINIONS graph, $n = 75\,877, m = 405\,739$



Community Detection using Motif-Based Random Walk



Motif-Based Random Walk – Notation



- Define $t_i(u)$ to be the number of triangles of type $i = 1, 2, 3$.

Motif-Based Random Walk

- Let $S \subseteq V$ be any set of vertices
- $\phi_3(S)$ the probability of leaving S in one step of the walk conditioned on being at a vertex u chosen from S proportionally to the number of triangles $t(u)$ it participates in
- Simple calculation reveals that $\phi_3(S) = \frac{t_2(S)+t_1(S)}{\text{vol}_3(S)}$
- Graph Triangle Conductance: $\phi_3(G) = \min_{S \subseteq V} \phi_3(S)$

Triangle Expanders

- **Definition:** A graph $G(V, E)$ is a triangle expander if all subsets $S \subseteq V$ with $|S| = s \leq 0.5n$ have constant triangle expansion, i.e., $\phi_3(S) = \Theta(1)$.

- **Theorem:** Let $G \sim G(n, \frac{\log(n)}{n^{1/3}})$. With high probability, G is a triangle expander.
  - Conjecture: same claim holds with $p = \frac{\log(n)}{n^{2/3}}$.

- **Theorem:** There exist edge expanders that are not triangle expanders. Similarly there exist triangle expanders that are not edge expanders.

Triangle Spectral Clustering

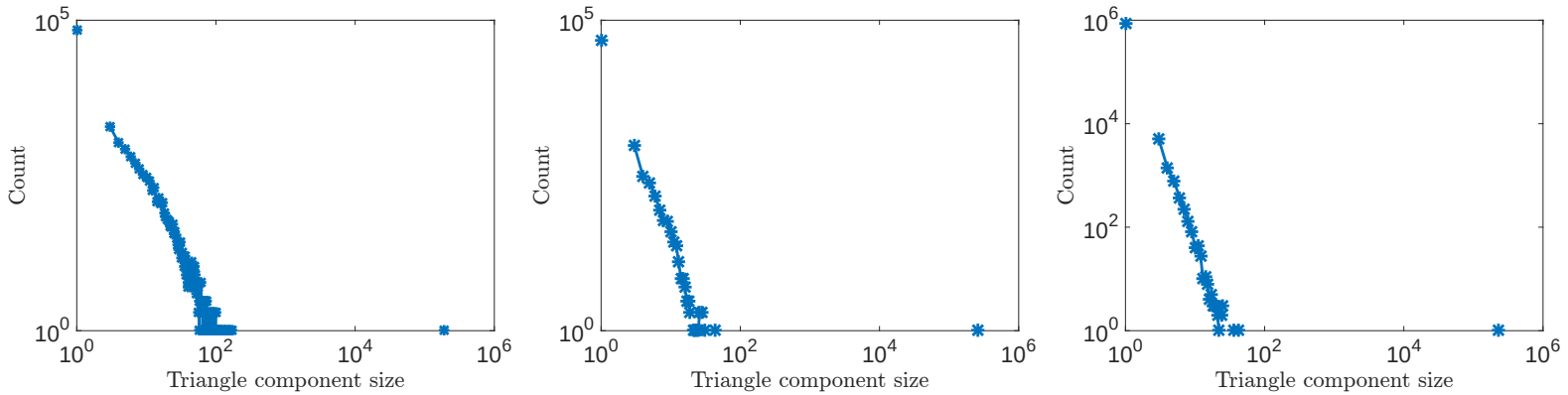
- **Algorithm:** (i) Reweight each edge in G by the number of triangles it participates in. (ii) **Next,** apply your favorite clustering algorithm on $H(V, E, w)$, the weighted version of G

- **Theorem:** Cheeger's clustering algorithm on $H(V, E, w)$ outputs a cut $(S : \bar{S})$ such that

$$\frac{\lambda_2(H)}{2} \leq \phi_3(G) \leq \sqrt{2\lambda_2(H)}.$$

Observation

Simple reweighting already gives a lot of information and outperforms existing methods!



Number of connected components versus size after reweighing each edge with triangle counts for (a) Amazon, (b) DBLP, and (c) Youtube graphs.

TECTONIC Heuristic

Input: Graph $G(V, E)$, threshold $\theta > 0$

1. Count $t(u, v)$ for each $(u, v) \in E$
2. Reweight each edge $(u, v) \in E$ by $w(u, v) \leftarrow \frac{t(u,v)}{deg(u)+deg(v)}$
3. Remove all edges (u, v) with weight $w(u, v) < \theta$
4. Output the resulting connected components

Inspired by hierarchical clustering, since

$$\frac{t(u, v)}{deg(u) + deg(v)} < \theta \Leftrightarrow dist^2(A^{(u)}, A^{(v)}) > \theta',$$

for $\theta = \frac{1}{2}\left(1 - \frac{\theta'}{deg(u)+deg(v)}\right)$.

Experimental Results

Groundtruth communities (available at SNAP)

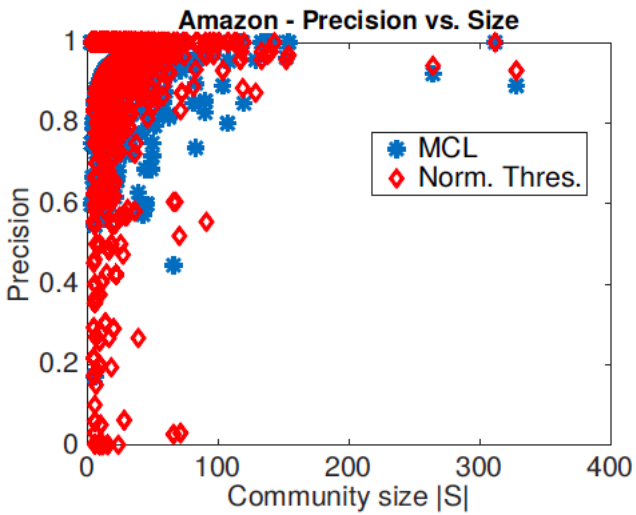
p precision, r recall, T run time in seconds

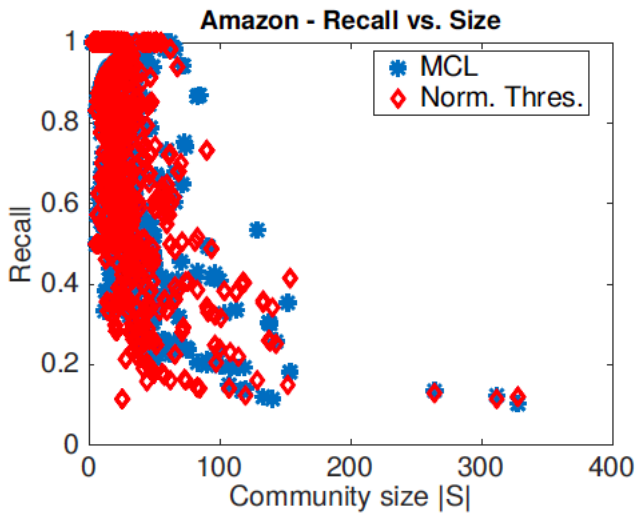
Quality: MCL, TECTONIC

Speed: TECTONIC

Method	Amazon			DBLP			YouTube		
	p	r	T	p	r	T	p	r	T
MCL	95.6	90.1	736.54	55.1	81.7	1 166	39.9	60.6	19 187.1
Louvaine	50.0	14.7	9.00	50.20	12.13	10.38	50.13	27.55	55.8
CFinder	-	-	$> 5h$	-	-	$> 5h$	-	-	$> 5h$
GN	-	-	$> 5h$	-	-	$> 5h$	-	-	$> 5h$
CNM	-	-	$> 5h$	-	-	$> 5h$	-	-	$> 5h$
Infomap	50.0	14.8	63.0	50.16	12.13	64.0	50.00	27.6	204
SC	-	-	$> 5h$	-	-	$> 5h$	-	-	$> 5h$
tSC	-	-	$> 5h$	-	-	$> 5h$	-	-	$> 5h$
Thres. 0	85.2	96.0	4.62	4.0	100.0	1.65	22.5	70.8	6.92
Thres. 1	94.1	81.1	4.61	12.0	91.4	1.65	36.1	59.7	6.92
Thres. 2	97.1	67.7	4.62	23.0	81.6	1.65	45.0	53.9	6.92
Thres. 3	98.0	52.4	4.62	35.7	71.4	1.65	49.6	50.3	6.93
TECTONIC	94.9	91.3	4.62	48.3	79.1	1.65	66.7	43.3	6.92

Experimental Results – MCL and TECTONIC





Community size vs precision (a), and recall (b) for the Amazon graph (top 5000 groundtruth communities)

As groundtruth community size increases, getting good recall results becomes increasingly hard.

thank you!

Node Embeddings (time permitted)

acknowledgements

references I

Alman, J. and Williams, V. V. (2024).

A refined laser method and faster matrix multiplication.

TheoretiCS, 3.

Alon, N. and Milman, V. D. (1985).

λ_1, isoperimetric inequalities for graphs, and superconcentrators.

Journal of Combinatorial Theory, Series B, 38(1):73–88.

Alon, N., Yuster, R., and Zwick, U. (1997).

Finding and counting given length cycles.

Algorithmica, 17(3):209–223.

Charikar, M. (2000).

Greedy approximation algorithms for finding dense components in a graph.

In *APPROX*.

references II

Chekuri, C., Quanrud, K., and Torres, M. R. (2022).

Densest subgraph: Supermodularity, iterative peeling, and flow.

In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1531–1555. SIAM.

Goldberg, A. V. (1984).

Finding a maximum density subgraph.

Technical report, University of California at Berkeley.

Grover, A. and Leskovec, J. (2016).

node2vec: Scalable feature learning for networks.

In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

references III

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021).

Highly accurate protein structure prediction with alphafold.

nature, 596(7873):583–589.

Kolountzakis, M. N., Miller, G. L., Peng, R., and Tsourakakis, C. E. (2012).

Efficient triangle counting in large graphs via degree-based vertex partitioning.

Internet Mathematics, 8(1-2):161–185.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013).

Efficient estimation of word representations in vector space.

arXiv preprint arXiv:1301.3781.

references IV

Ng, A., Jordan, M., and Weiss, Y. (2001).

On spectral clustering: Analysis and an algorithm.

Advances in neural information processing systems, 14.

Pagh, R. and Tsourakakis, C. E. (2012).

Colorful triangle counting and a mapreduce implementation.

Information Processing Letters, 112(7):277–281.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014).

Deepwalk: Online learning of social representations.

In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710.

references V

Shi, J. and Malik, J. (2000).

Normalized cuts and image segmentation.

IEEE Transactions on pattern analysis and machine intelligence, 22(8):888–905.

Tsourakakis, C. E. (2008).

Counting triangles in real-world networks: algorithms and laws.

In *Proceedings of the 2008 IEEE International Conference on Data Mining (ICDM)*, pages 608–617. IEEE.

Tsourakakis, C. E., Kang, U., Miller, G. L., and Faloutsos, C. (2009).

Doulion: Counting triangles in massive graphs with a coin.

In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 837–846. ACM.

references VI

Tsourakakis, C. E., Kolountzakis, M. N., and Miller, G. L. (2011).

Triangle sparsifiers.

In *Proceedings of the 2011 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1051–1059. ACM.