Local Graph Clustering

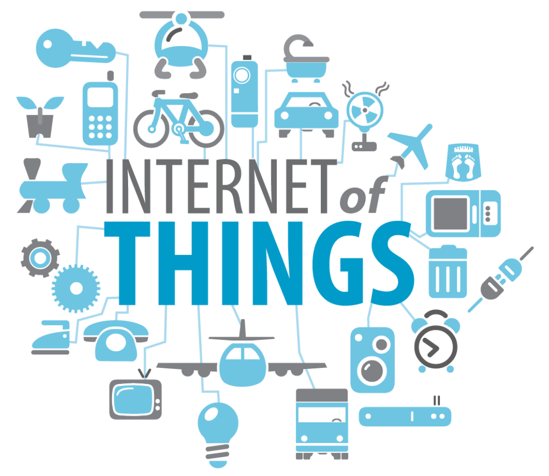
Kimon Fountoulakis @CS UWaterloo
29/04/2020
Waterloo A&C seminar

Graphs are ubiquitous

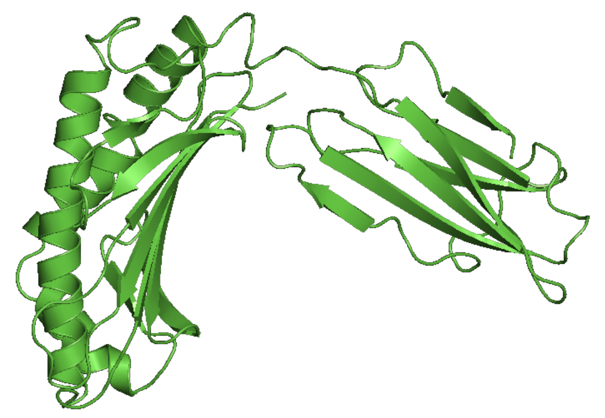
- Graphs are important for modeling relational data; they enable relationships between events to be discovered

- Social networks

- Communication networks

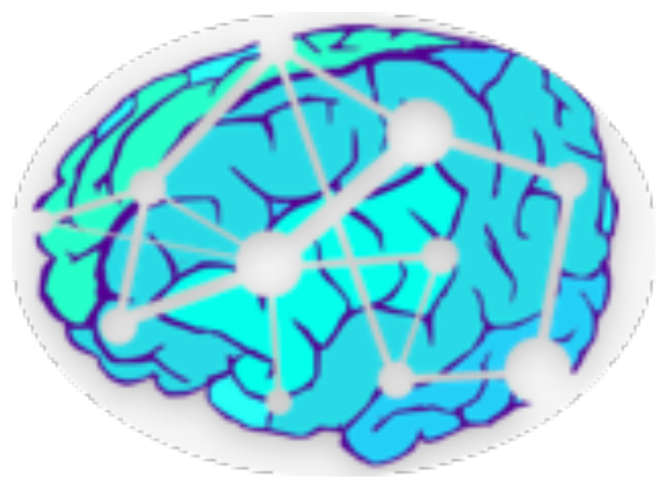


- Cellular and molecule networks



- IP/Domain networks

- Neural networks



Graphs are ubiquitous in academia as well

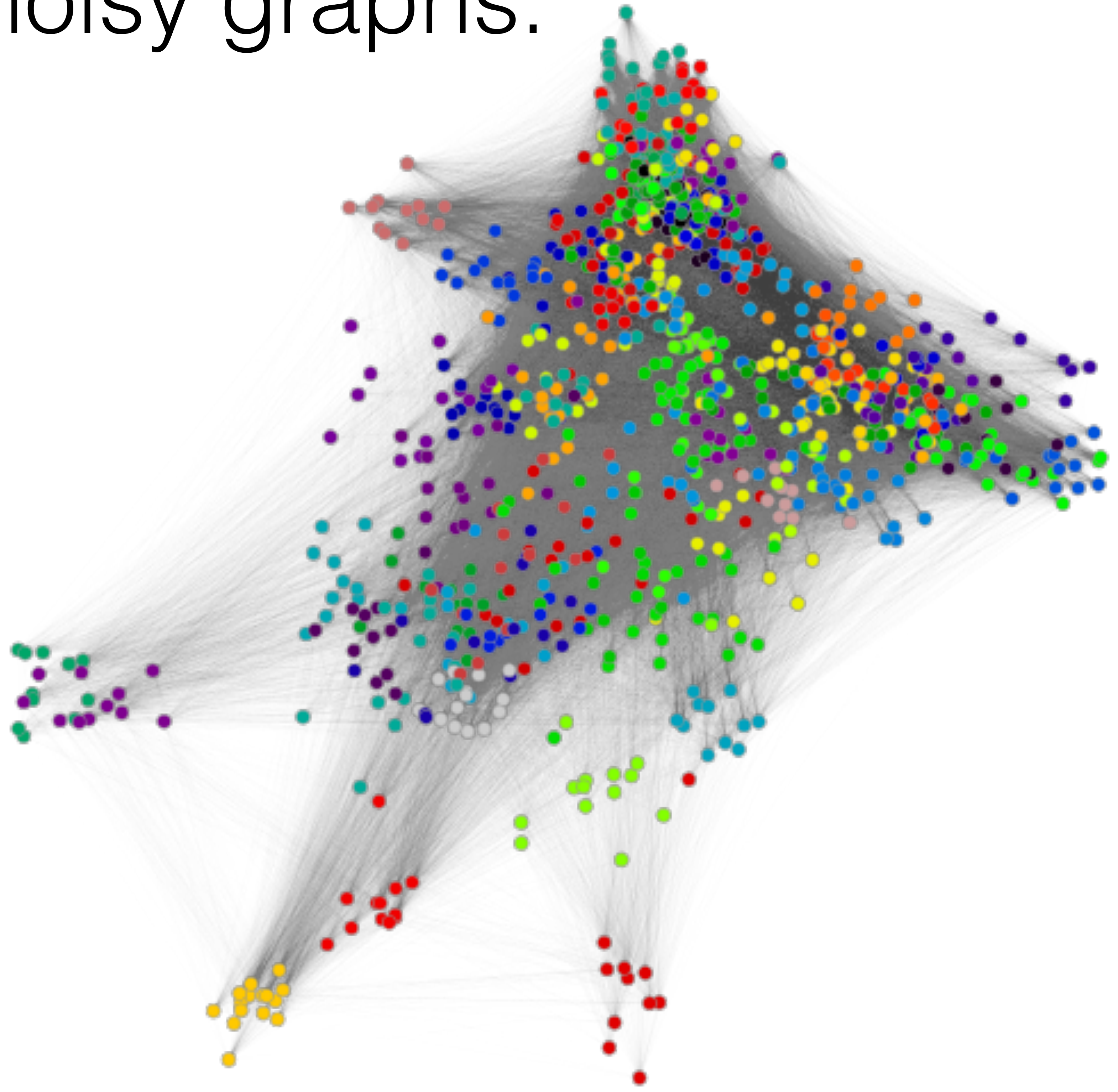
-Graphs are popular in traditional fields such as theoretical computer science and statistics. Applications include triangle counting, routing flows, planted clique etc.

-However, recently, there has been an explosion of large-scale graph data that exhibit new challenges.

Modern graphs do not exhibit a clear global structure

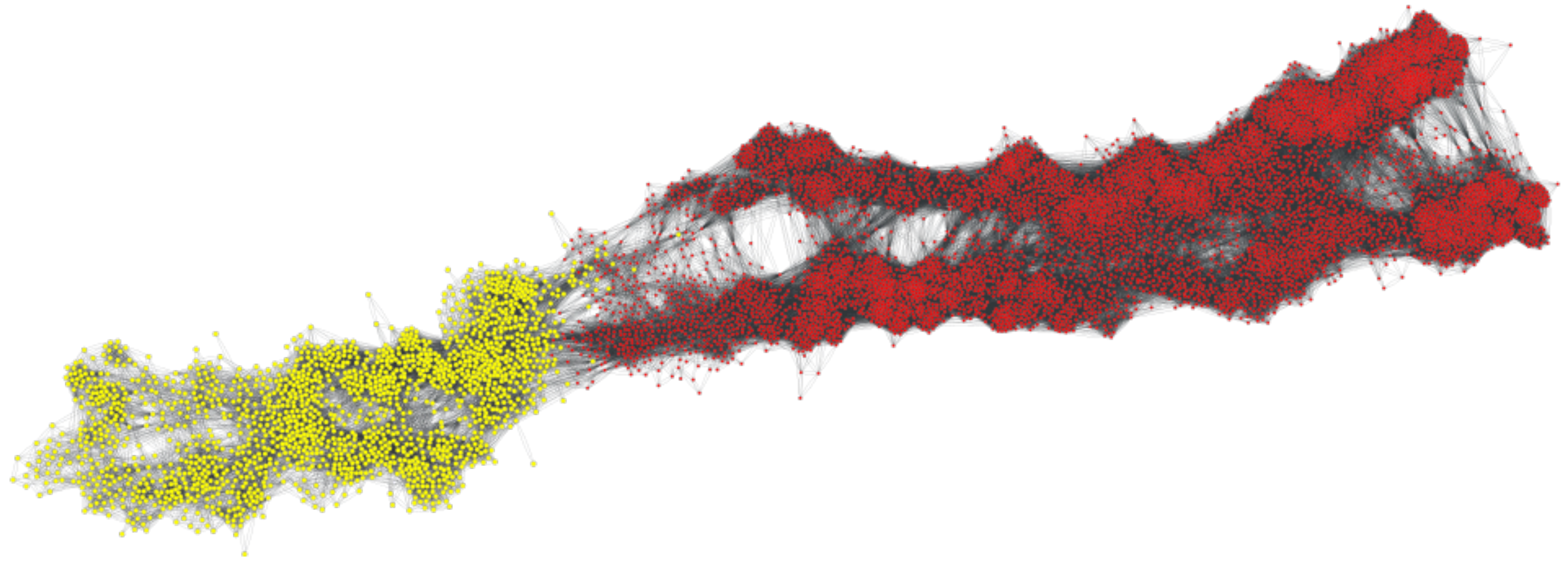
- Real large-scale graphs have rich local structure

- We often have to detect small clusters in large and noisy graphs:



protein-protein interaction graph,
color denotes similar functionality

Rather than partitioning graphs with nice structure



US-Senate graph,
nice bi-partition in year 1865 around the end of the American civil war

Our goal

- We need methods that are able to probe graphs with billions of nodes and edges.

- The running time of the new methods should depend on the size of the output instead of the size of the whole graph.

- The new methods should be supported by worst- and average-case theoretical guarantees.

Today’s topics

1. Local graph clustering, definition and examples
2. Local spectral methods
3. Local flow methods
4. Local p-norm local flow methods
5. Scalability to multi-core hardware

About this talk

-I will mostly discuss methods, I will demonstrate theoretical results and I will present experiments that promote understanding of the methods within the available time.

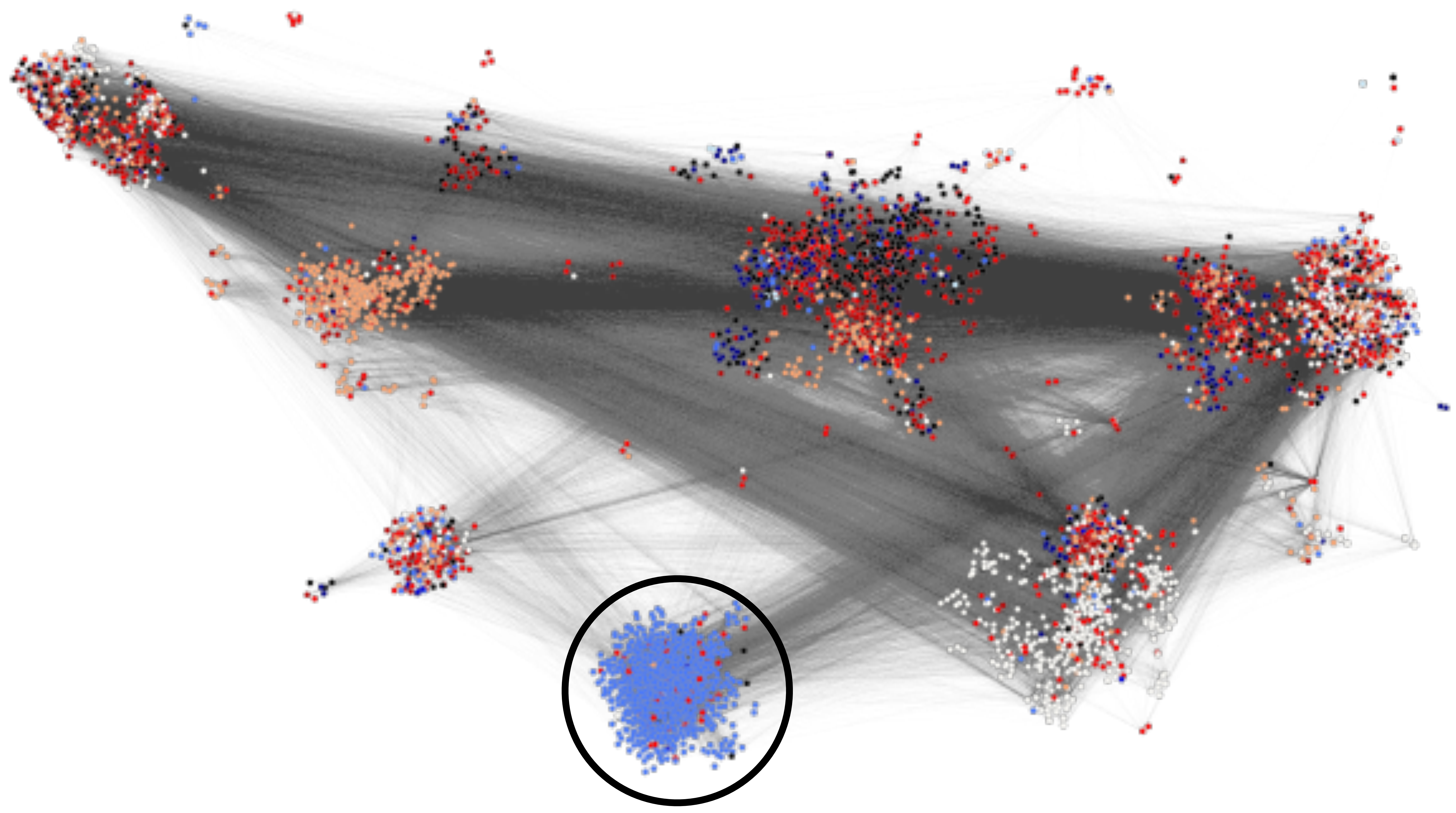
-For extensive experiments on real-data please check the cited papers. We literally have performed hundreds of experiments for measuring performance of local graph clustering methods.

Brief definition of local graph clustering

-Find set of nodes A given a seed node in target set B

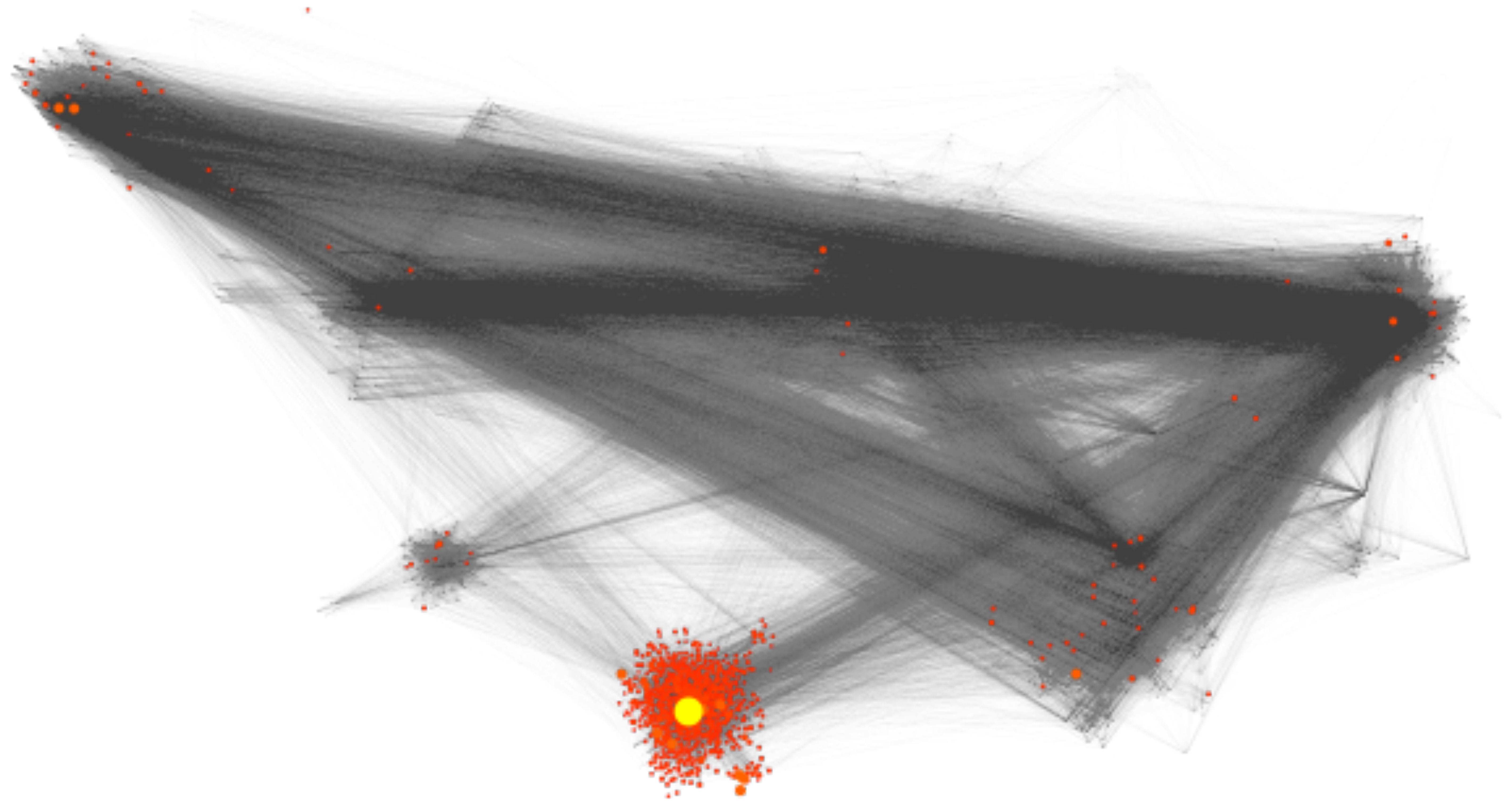
- -Set A has good precision/recall w.r.t set B
- -The running time depends on size of B instead of the whole graph

Facebook Johns Hopkins social network: color denotes class year



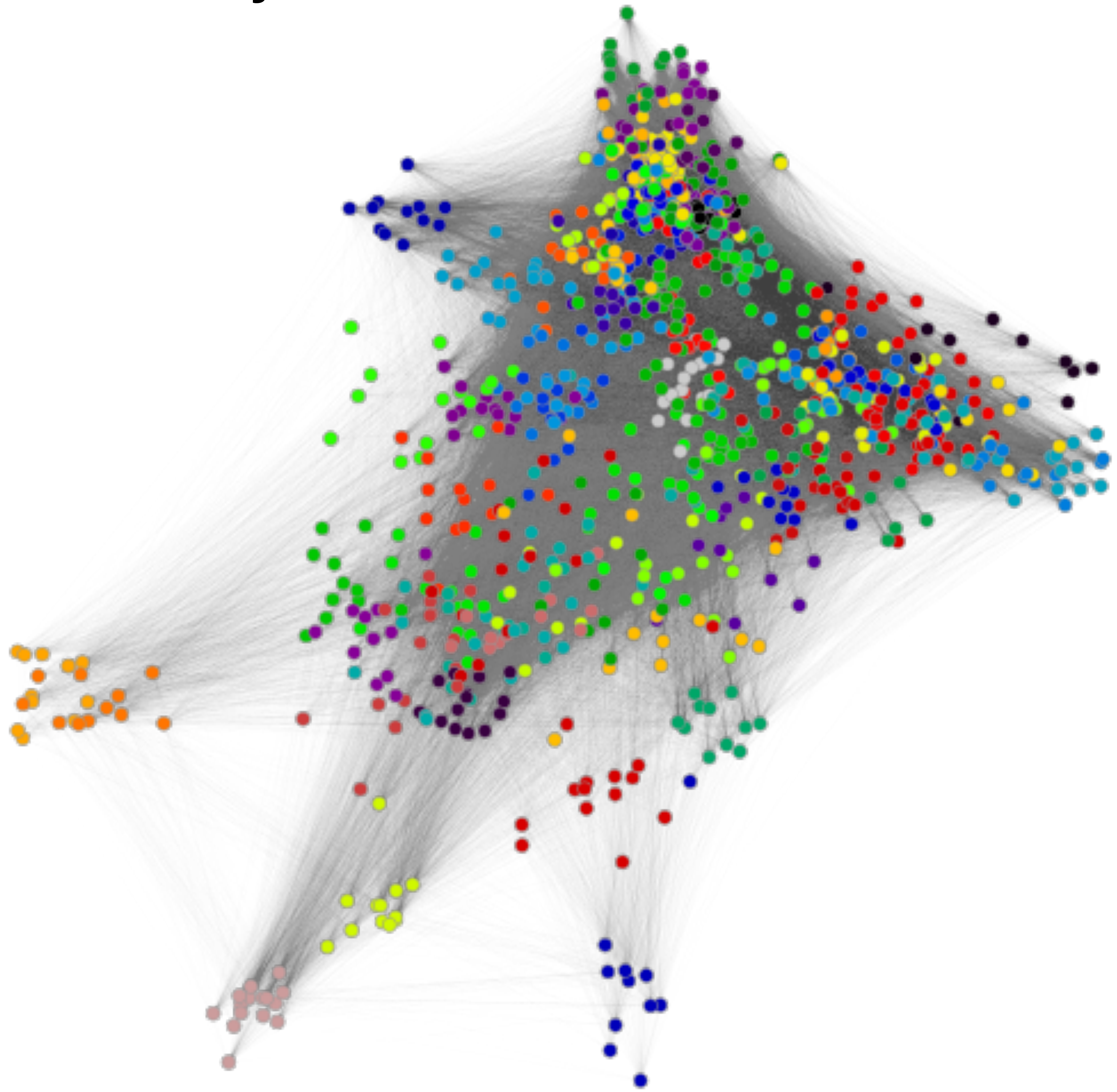
Data: Facebook Johns Hopkins, A. L. Traud, P. J. Mucha and M. A. Porter, Physica A, 391(16), 2012

Example



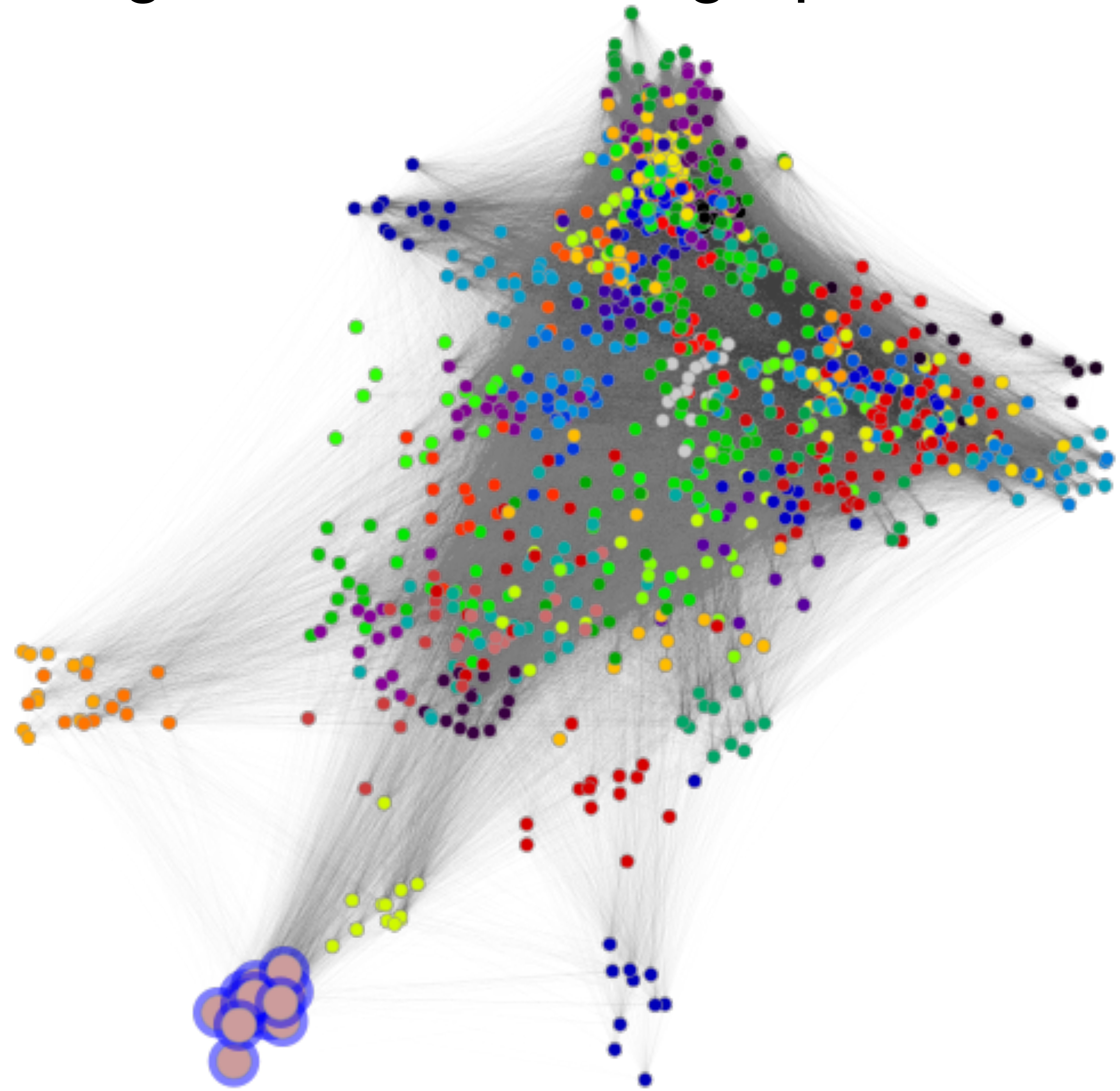
Data: Facebook Johns Hopkins, A. L. Traud, P. J. Mucha and M. A. Porter, Physica A, 391(16), 2012

Protein structure similarity: color denotes similar function



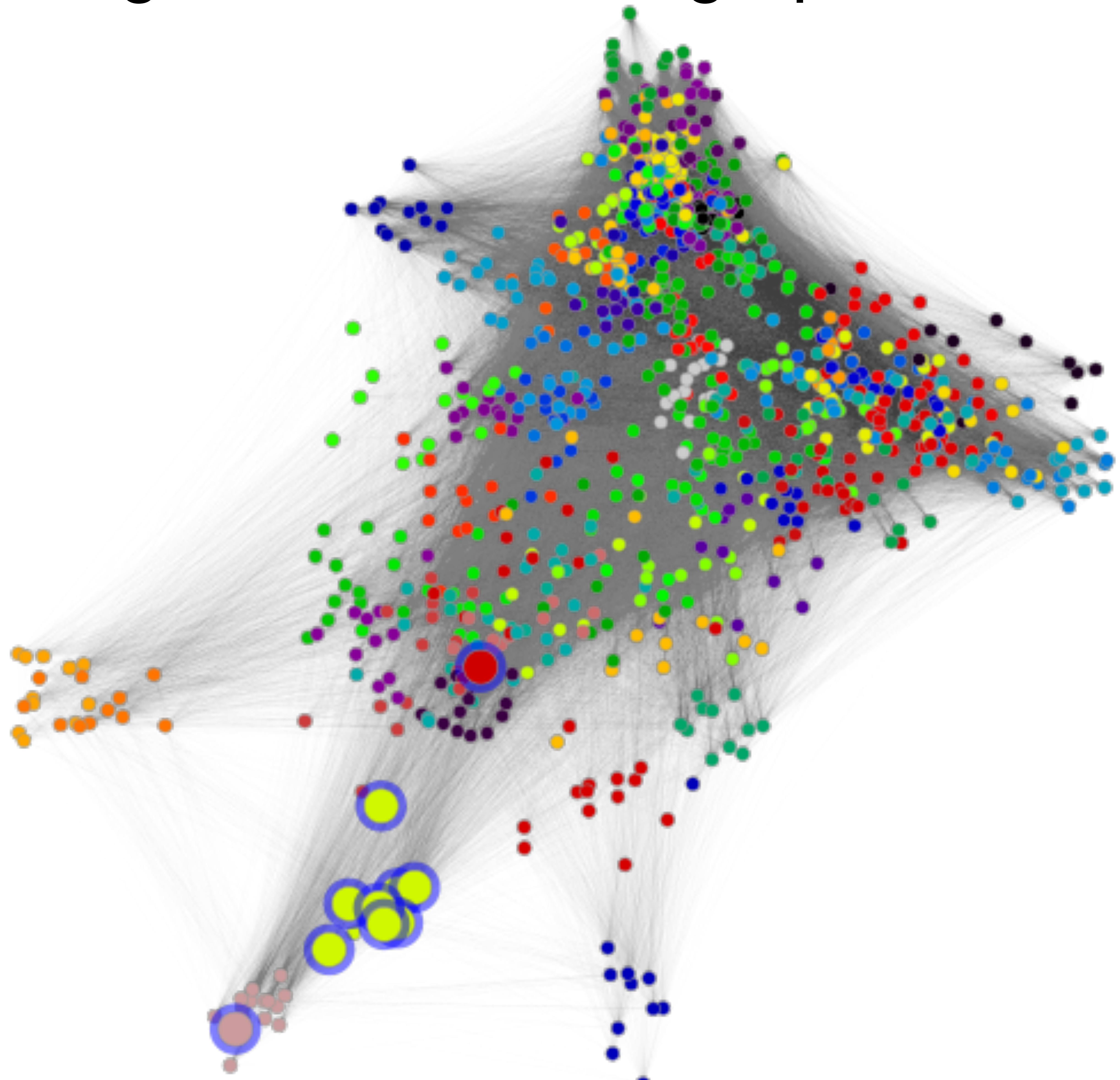
Data: The MIPS mammalian protein-protein interaction database. *Bioinformatics, 21(6):832-834*, 2005

Local graph clustering finds 2% of the graph



Data: The MIPS mammalian protein-protein interaction database. *Bioinformatics, 21(6):832-834*, 2005

Local graph clustering finds 1% of the graph



Data: The MIPS mammalian protein-protein interaction database. *Bioinformatics, 21(6):832-834*, 2005

Warm-up: Local Spectral Methods

Some definitions

Graph: $G = (\underbrace{V}_{nodes}, \underbrace{E}_{edges}), |V| = n, |E| = m$

- n x n adjacency matrix: A
- An element of A is equal to 1 if two nodes are connected
- Degree matrix: $D = diag(A1_n)$, 1_n is a vector of all ones.
- Each element of D shows the number of neighbors of a node
- Random walk matrix: AD^{-1}
- Lazy random walk matrix: $W = \frac{1}{2}\left(I + AD^{-1}\right)$
- Graph Laplacian: $L = D - A$

Personalized random walk

- Let $\alpha \in (0, 1)$

- Consider a diffusion process where we perform lazy random walk with probability 1-alpha, and jump to a given seed node with probability alpha:

$$\alpha s 1_n^T + (1 - \alpha) W$$

- where s is an indicator vector of the seed node and alpha is the teleportation parameter.

- **Simple idea:** use a random walk from a seed node. The nodes with the highest probability after k steps consist a cluster.

Let’s get rid off the tail

- For the stationary personalized PageRank vector most of the probability mass is concentrated around the seed node.

- This means that the ordered personalized PageRank vector has long tail for nodes far away from the seed node.

- We can efficiently cut the tale using l1-regularized PageRank without even having to compute the long tail.

L1-regularized PageRank

$$\text{minimize} \quad \underbrace{\frac{1}{2}x^T Q x - \alpha x^T s}_{f(x)} + \underbrace{\rho\alpha \|Dx\|_1}_{g(x)}$$

where

$$Q = \alpha D + \frac{1-\alpha}{2} L$$

- Simple check: for ρ=0 the optimality conditions of the l1-regularized problem give the personalized PageRank linear system.

Fountoulakis et al. Variational Perspective of Local Graph Clustering, Mathematical Programming, 2017

Properties of the l1-regularized optimal solution

- **Theorem**

- If the graph is unweighted then the number of nonzero nodes in the optimal solution is bounded by 1/ρ.

- If the graph is weighted then the volume of nonzero nodes in the optimal solution is bounded by 1/ρ.

Sketch of proof

- **Theorem:** Let $\hat{x}$ be the optimal solution. Then the volume of the support of the solution is bounded Bounded volume of the support

$$\mathrm{vol}(\hat{S}) = \sum_{i \in \hat{S}} d_i \leq \frac{1}{\rho} \qquad \text{where} \qquad \hat{S} = \{i \mid \hat{x}_i \neq 0\}$$

Result 1: Negative partial derivatives are bounded from below

$$\rho \alpha^{1/2} \leq -\nabla_i f(\hat{x}) \quad \forall i \in \hat{S} \rightarrow \mathrm{vol}(\hat{S}) \rho \alpha \leq \|D^{1/2} \nabla f(\hat{x})\|_1$$

Result 2: Gradients are bounded from above

$$\|D^{1/2} \nabla f(\hat{x})\|_1 \leq \alpha$$

Results 1 + 2 use proof-by-algorithm + strong convexity, Results 1 + 2 give the final result.

The solution path is monotonic

- **Theorem**

- Let $\hat{x}(\rho)$ be the solution of the l1-regularized problem as a function of ρ.

- Then $\hat{x}(\rho)$ is a component-wise monotone function

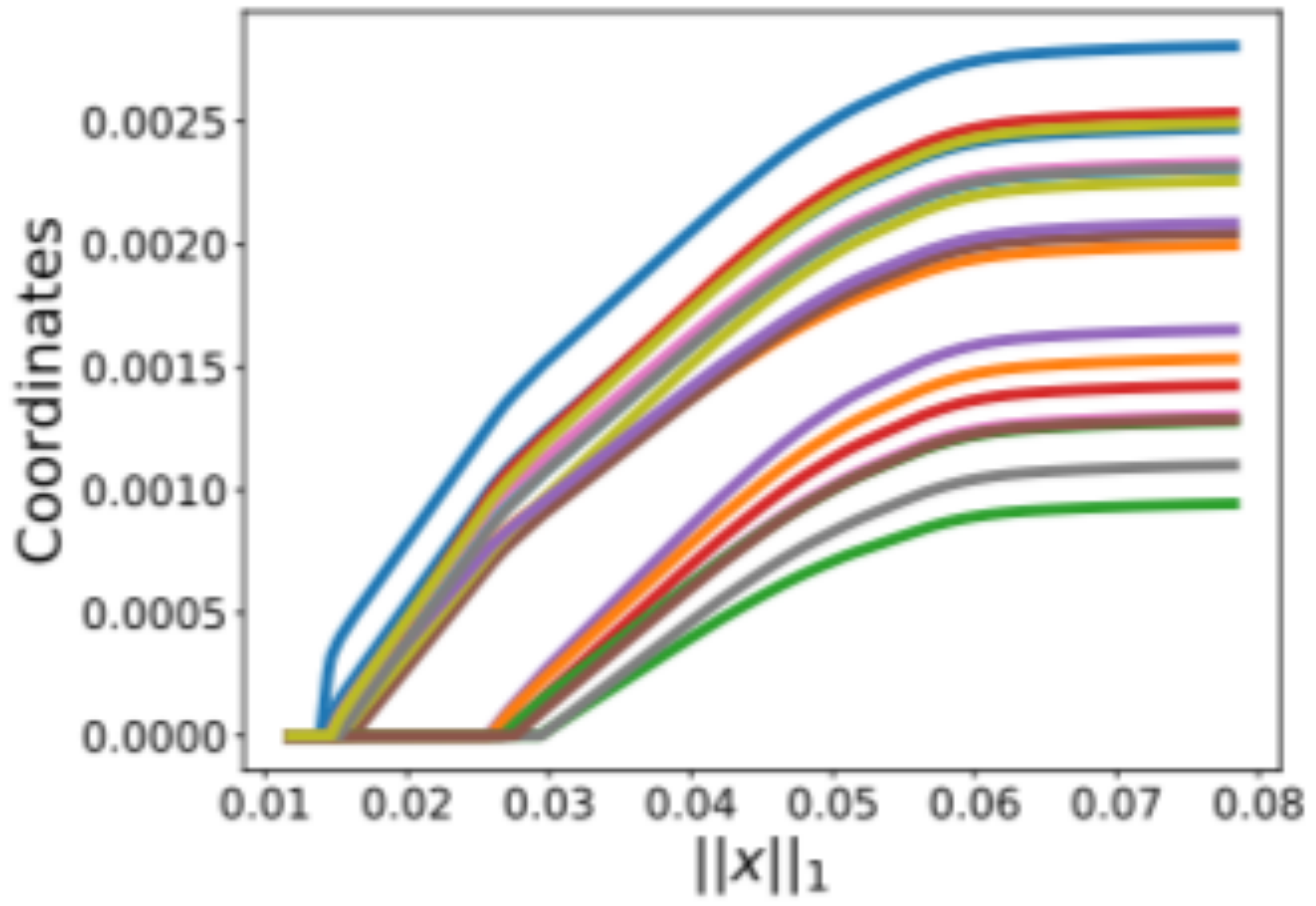
$$\hat{x}(\rho_0) \leq \hat{x}(\rho_1) \text{ for } \rho_0 > \rho_1$$

- The inequality becomes strict when a component is positive.

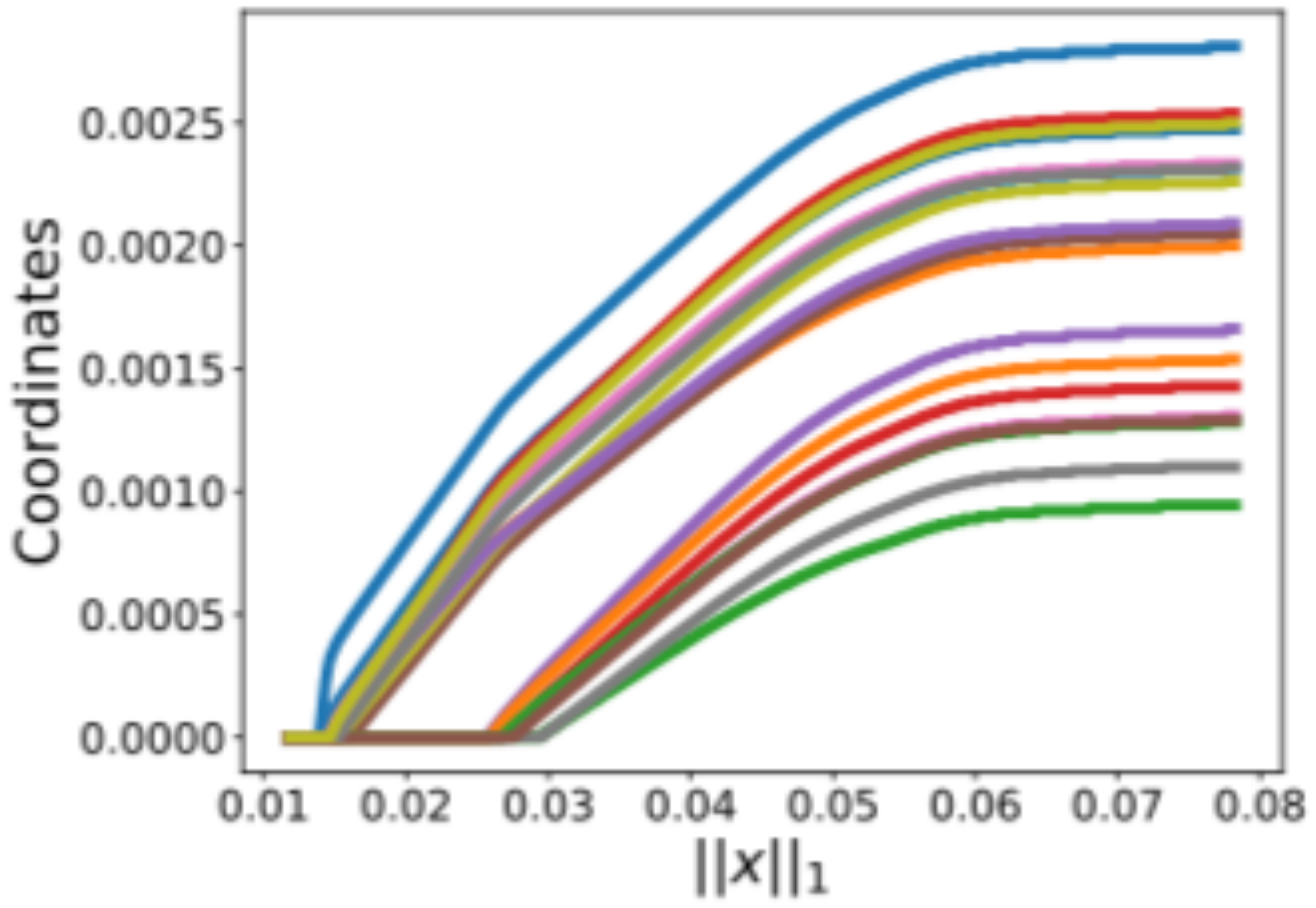
Recover the whole path with the cost of one solve (in worst-case)

- **Corollary**

  - The stage-wise algorithm (next slide) converges to the l1-regularized solution path if we drag the step-size of the algorithm to zero.



(a) ℓ_1-reg. path



(b) Stage. path $\eta = 10^{-4}$

W. Ha, K. Fountoulakis, M. Mahoney. Statistical Guarantees of Local Graph Clustering. AISTATS 2020

Stage-wise for recovering the whole path

- **Stage-wise algorithm**

$$1)\ \text{Choose } i \text{ such that } \left|d_i^{-1}\nabla_i f(x_k)\right| \text{ is the largest among } [n]$$

$$2)\ \text{Update } [x_{k+1}]_i = [x_k]_i + \frac{\eta}{d_i}$$

- **Corollary**

- The stage-wise algorithm converges to the l1-regularized solution path if we drag the step-size η of the algorithm to zero.

- The running time of stage-wise depends on the nonzero nodes and its neighbors and not on the size of the whole graph.

What if we do not want to recover the whole path?

$$\text{minimize}\ \underbrace{\frac{1}{2}x^TQx - \alpha x^Ts}_{f(x)} + \underbrace{\rho\alpha\|Dx\|_1}_{g(x)}$$

Proximal gradient descent

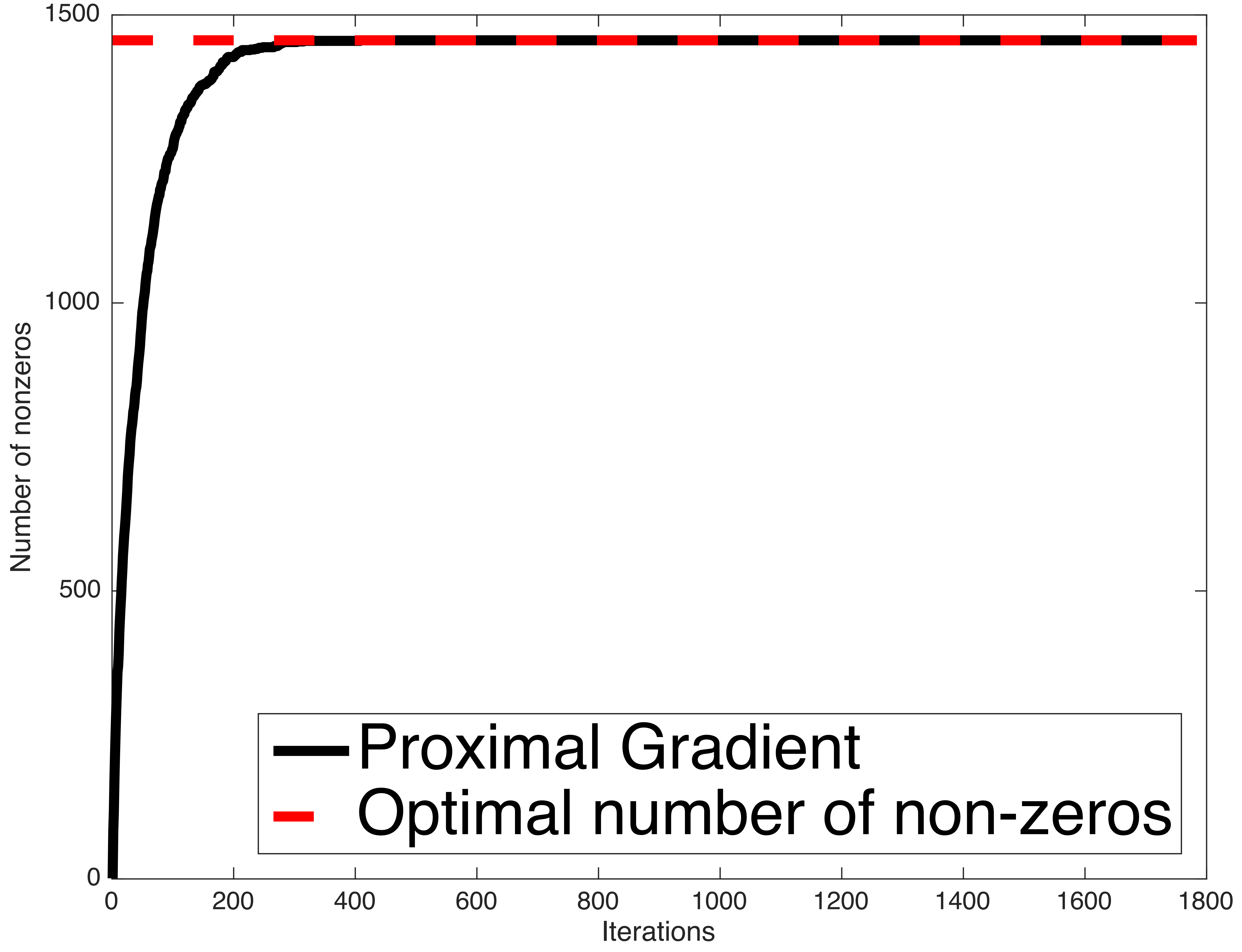
$$x_{k+1} := \text{argmin}\ g(x) + \underbrace{f(x_k) + \langle \nabla f(x_k), x - x_k\rangle}_{\text{first-order Taylor approximation}} + \underbrace{\frac{1}{2}\|x - x_k\|_2^2}_{\substack{\text{upper bound on the}\\ \text{approximation error}}}$$

Requires careful implementation to avoid excessive running time

- Need to maintain a set of non-zero nodes
- Update x and gradient only for non-zero nodes and their neighbors at each iteration

Fountoulakis et al. Variational Perspective of Local Graph Clustering, Mathematical Programming, 2017

Theorem: non-decreasing non-zero nodes



Fountoulakis et al. Variational Perspective of Local Graph Clustering, Mathematical Programming, 2017

Sketch of proof

-**Theorem:** non-decreasing non-zero nodes

Result 1: Using induction we get that negative partial derivatives are bounded

$$-\nabla_i f(x_k) \geq \rho\alpha d_i^{1/2} \ \forall i \in S_k \quad \text{and} \quad -\nabla_i f(x_k) < \rho\alpha d_i^{1/2} \ \forall i \in [n]\backslash S_k \quad \forall k$$

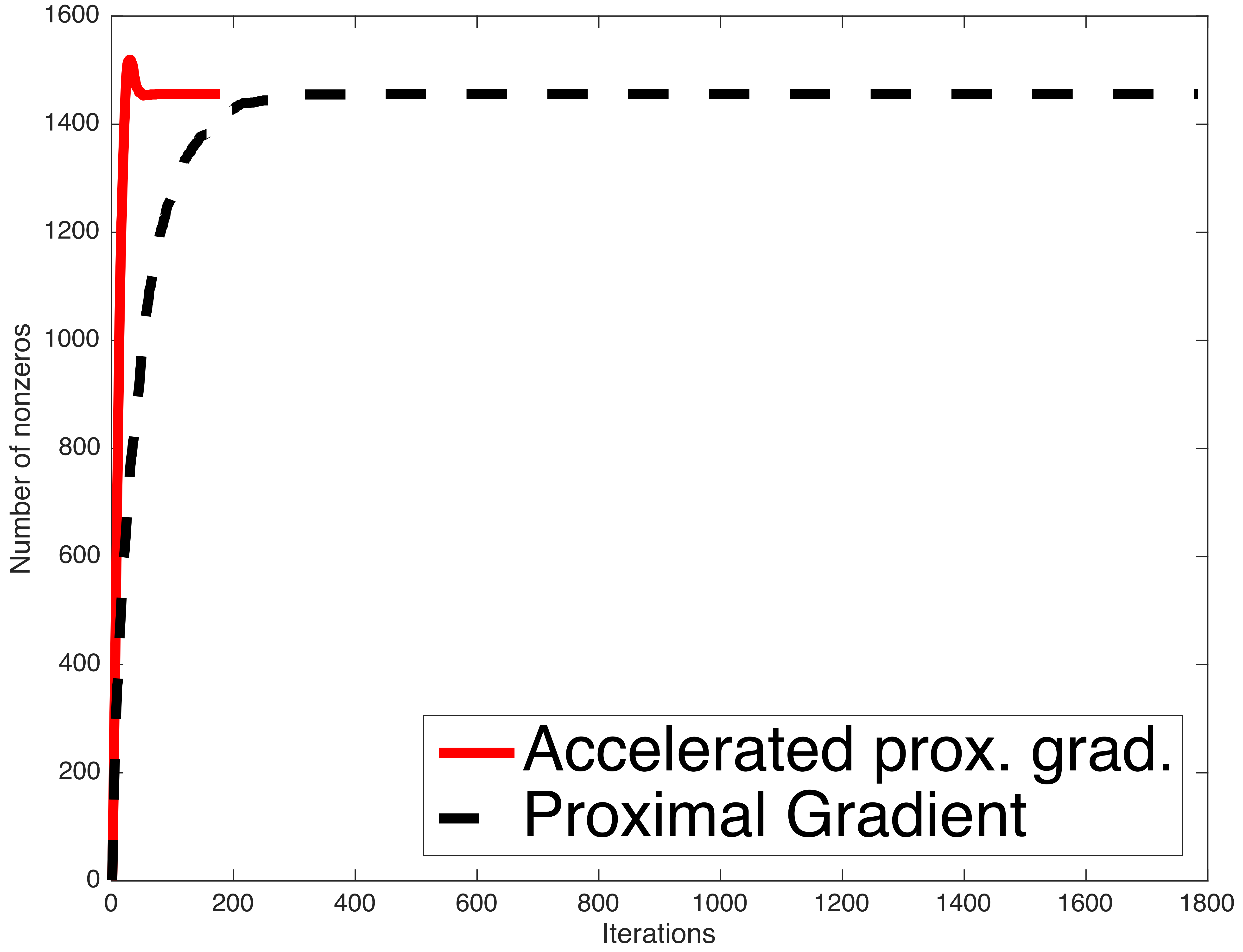
$\Big\downarrow$ Using the definition of a proximal step

Result 2: The mass of the variables is non-decreasing

$$x_k \leq x_{k+1} \quad \forall k$$

Results 1 + 2 give $S_k \subseteq S_{k+1}$

Open problem: is accelerated prox. grad. a local algorithm?



Gradient descent running time

$$\tilde{\mathcal{O}}\left(\frac{\text{vol}(\hat{S})}{\mu}\right)$$

Accel. gradient descent

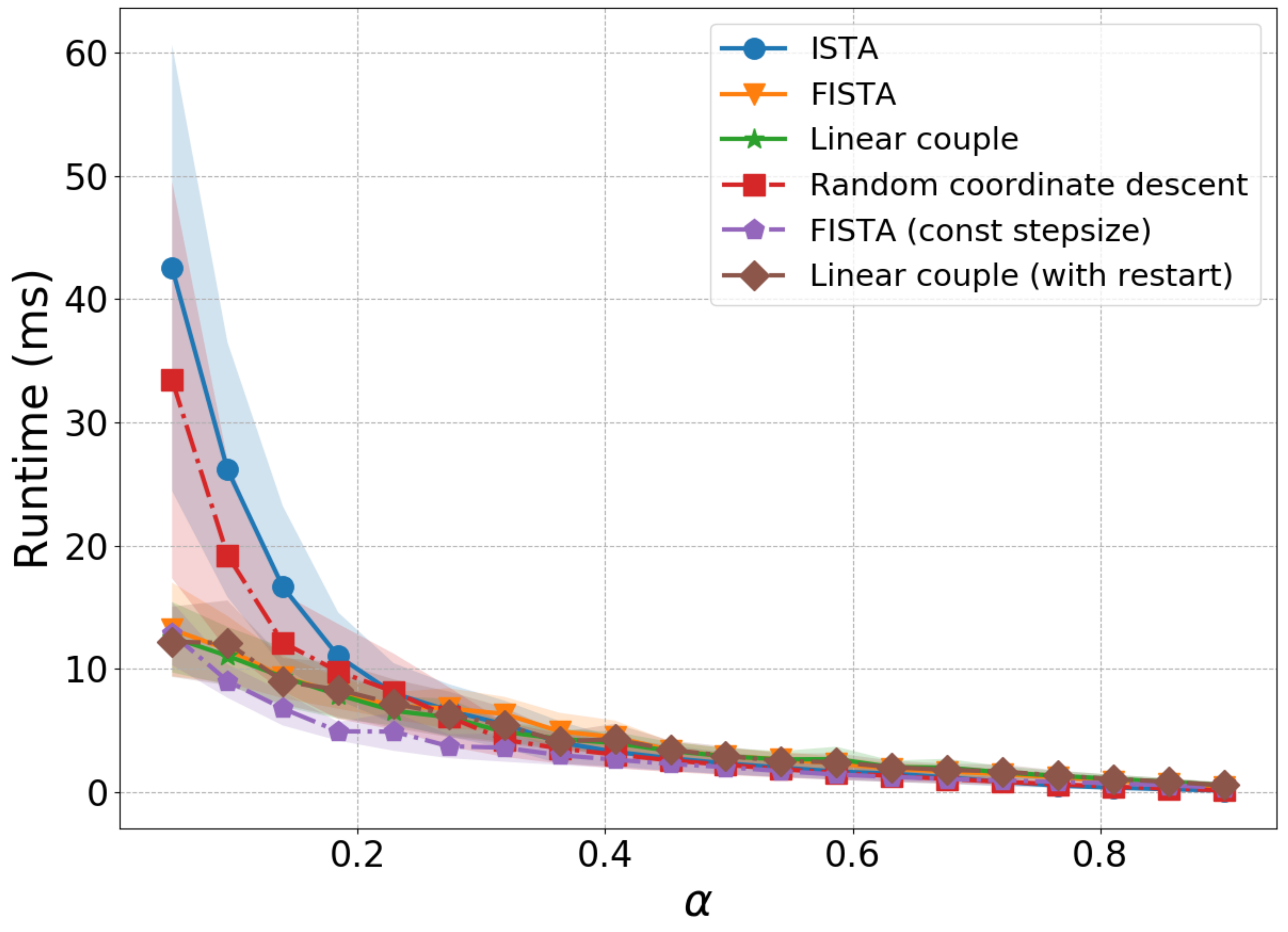
$$\tilde{\mathcal{O}}\left(\frac{\text{vol}(G)}{\sqrt{\mu}}\right)$$

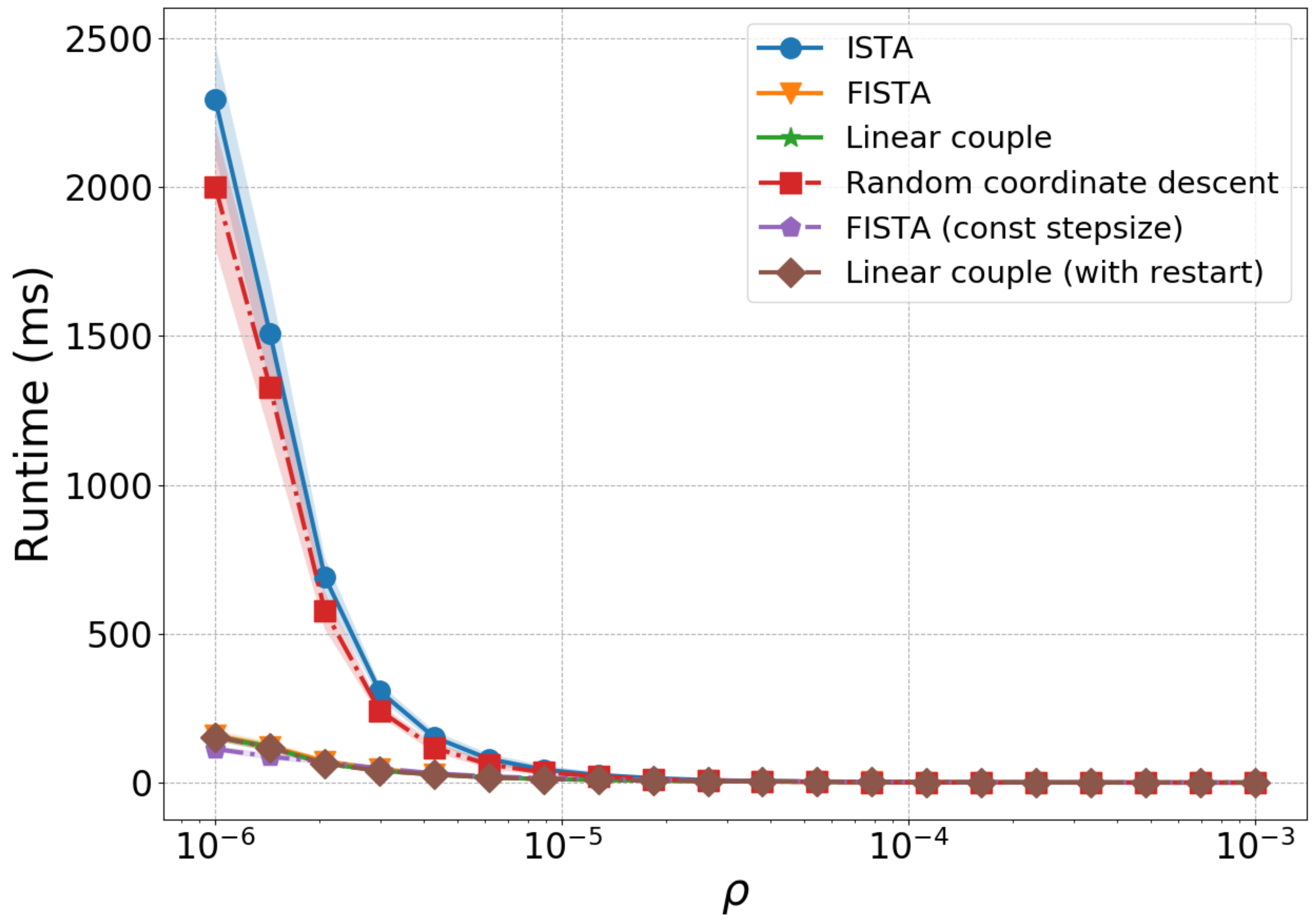
$$\Downarrow$$

$$\tilde{\mathcal{O}}\left(\frac{\text{vol}(\hat{S})}{\sqrt{\mu}}\right)$$

- $\hat{S}$: support of optimal solution, i.e., non-zero nodes.
- μ strong convexity parameter of the problem.

Accelerated prox. grad. is the fastest in practice





Two ways to measure performance of the l1-regularized PageRank model

Average-case

- Performance under stochastic block model - recover a cluster using the output of l1-regularized PageRank.

W. Ha, K. Fountoulakis, M. Mahoney. Statistical Guarantees of Local Graph Clustering. AISTATS 2020

Worst-case

- Use conductance to measure quality of the output. Show that the output has conductance value similar to a target cluster around the seed node.

Fountoulakis et al. Variational Perspective of Local Graph Clustering, Mathematical Programming, 2017

Average-case guarantees

Average-case performance

Local random model

- Given a graph G with n nodes, let K be a target cluster inside G.
- Two nodes in K are connected with probability p
- Nodes in K are connected K^c with probability q.
- The rest of edges can be drawn using **any** other model.

Expected l1-regularized PageRank

- The optimal solution of the expected problem identifies the target cluster.

- **Theorem**

- Suppose that the seed node is selected from target cluster K. The optimal solution of

$$x^* := \operatorname{argmin} \ \frac{1}{2} x^T \mathbb{E}[Q] x - \alpha x^T s + \rho\alpha \|\mathbb{E}[D]x\|_1$$

- satisfies

$$\operatorname{supp}(x^*) = K$$

- as long as $\rho = \mathcal{O}\left(\frac{p}{\bar{d}^2}\right)$

- where $\bar{d}$ is the expected degree of nodes in the target cluster.

Results for l1-regularized PageRank for noisy data

- In practice, we do not have access to the expected graph. We are given a realization of the local random model that includes “noise”, i.e., edges from the target cluster to the rest of the graph.

- We have two results for the noisy case.
- **First result.**
- Zero false negatives.
- Bounded false positives.
- **Second result.**
- With additional assumptions on the seed nodes we can show exact recovery.

W. Ha, K. Fountoulakis, M. Mahoney. Statistical Guarantees of Local Graph Clustering. AISTATS 2020

Results for l1-regularized PageRank for noisy data

- **Theorem (bounded false positives)**
- Suppose $p^2 k \geq \mathcal{O}(\log k)$, where k is the size of the target cluster.
- and $\rho = \mathcal{O}(\frac{\gamma p}{\bar{d}^2})$
- where $\gamma = \frac{pk}{\bar{d}}$, i.e., the probability of staying inside the target cluster in one step.
- Then with probability $1 - 6\exp(-\mathcal{O}(p^2 k))$ the optimal solution of the realized problem has **zero false negatives** and the false positives are bounded

$$\mathrm{vol}(FP) \leq \mathrm{vol}(K) \left(\mathcal{O} \left(\frac{1}{\gamma^2} \right) - 1 \right)$$

Noisy results for l1-regularized PageRank

- **Theorem (exact recovery)**
- Let $q = \mathcal{O}\left(\frac{1}{n}\right)$
- Then with probability at least $1 - \mathcal{O}(e^{-k})$ there exists a good seed node such that if we use that seed node we get

$$\mathrm{supp}(\hat{x}) = K$$

- As long as

$$d_j \geq \mathcal{O}\left(\frac{1}{\gamma p}\right) \quad \forall j \in K^c$$

W. Ha, K. Fountoulakis, M. Mahoney. Statistical Guarantees of Local Graph Clustering. AISTATS 2020

Noisy results for l1-regularized PageRank

- The assumption that $q = \mathcal{O}\left(\frac{1}{n}\right)$

- implies that there are constant number of edges leaving the cluster, which sounds artificial.

- but it is not, because it also covers the case were the size of the target cluster is $k = \mathcal{O}(1)$

- This is a realistic local graph clustering setting where we attempt to recover a very small target cluster of constant size with constant number of edges leaving the cluster.

Worst-case guarantees

Some definitions

- **Conductance of target cluster B:**

$$\Phi(B) := \left(\frac{\text{number of edges leaving B}}{\text{sum of edges of vertices in B}} \right)$$

Assuming B is the smaller part of the graph

- **Internal connectivity of target cluster B**

IC(B):=the minimum conductance of the subgraph induced by B

Worst-case performance

- **Theorem (by Zhu et al.)**
- Assume that the internal connectivity of the target cluster K is larger than its conductance

$$\frac{\mathrm{IC}^2(K)}{\Phi(K)\log \mathrm{vol}(K)} \geq \Omega(1)$$

- False positives are bounded by

$$\mathrm{vol}(FP) \leq \mathcal{O}\left(\frac{\Phi(K)}{\mathrm{IC}(K)}\right)\mathrm{vol}(K)$$

- True positives are bounded by

$$\mathrm{vol}(TP) \leq \mathcal{O}\left(\frac{\Phi(K)}{\mathrm{IC}(K)}\right)\mathrm{vol}(K)$$

Zhu et al. A local algorithm for finding well-connected clusters, ICML, 2013

Compare average- and worst-case

	False Positives	**True Negatives**
Average-case	$\mathrm{vol}(FP) \leq \mathrm{vol}(K)\left(\mathcal{O}\left(\frac{1}{\gamma^2}\right) - 1\right)$	zero
Worst-case	$\mathrm{vol}(FP) \leq \mathrm{vol}(K)\mathcal{O}\left((1-\gamma)\log k\right)$	$\mathrm{vol}(TP) \leq \mathrm{vol}(K)\mathcal{O}\left((1-\gamma)\log k\right)$

- The average-case result on FP is stronger for large values of gamma.
- Also for the average-case we can also prove exact recovery.

Parallel local spectral methods in shared memory

- Why shared memory? Currently the largest publicly available graphs can be stored in computers with shared memory

- We parallelize 4 local spectral methods + rounding
  1. **Gradient descent for L1-regularized PageRank** (in this talk)
  2. Nibble
  3. Deterministic HeatKernel Approximate PageRank
  4. Randomized HeatKernel Approximate PageRank
  5. Sweep cut rounding algorithm

J. Shun, K. Fountoulakis, F. Khorasani, M. Mahoney. Parallel Local Graph Clustering, VLDB, 2016

Overview of results

- 3-16x faster than serial version

- Parallelization allowed us to solve problems of billions of nodes and edges.

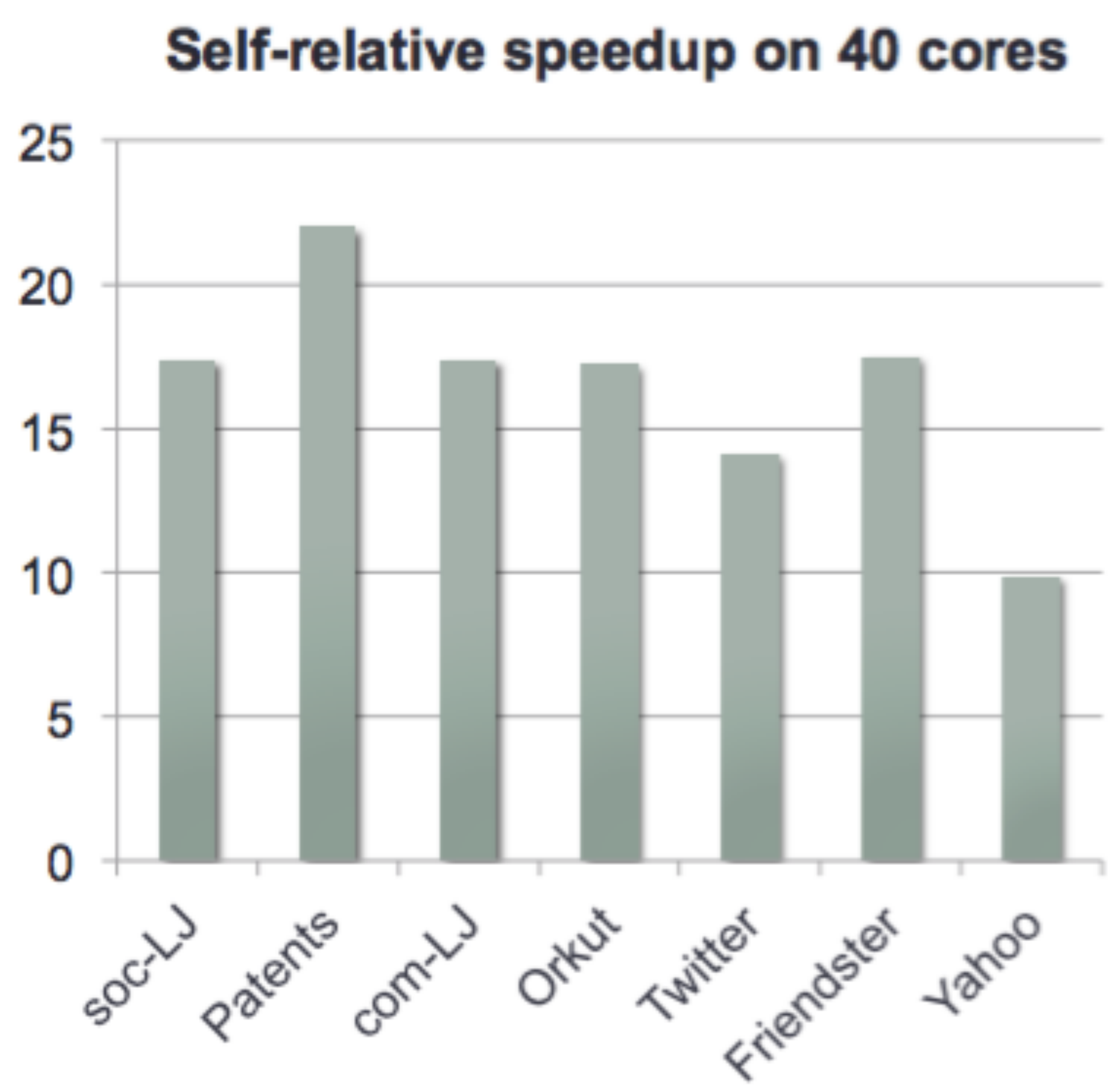
J. Shun, K. Fountoulakis, F. Khorasani, M. Mahoney. Parallel Local Graph Clustering, VLDB, 2016

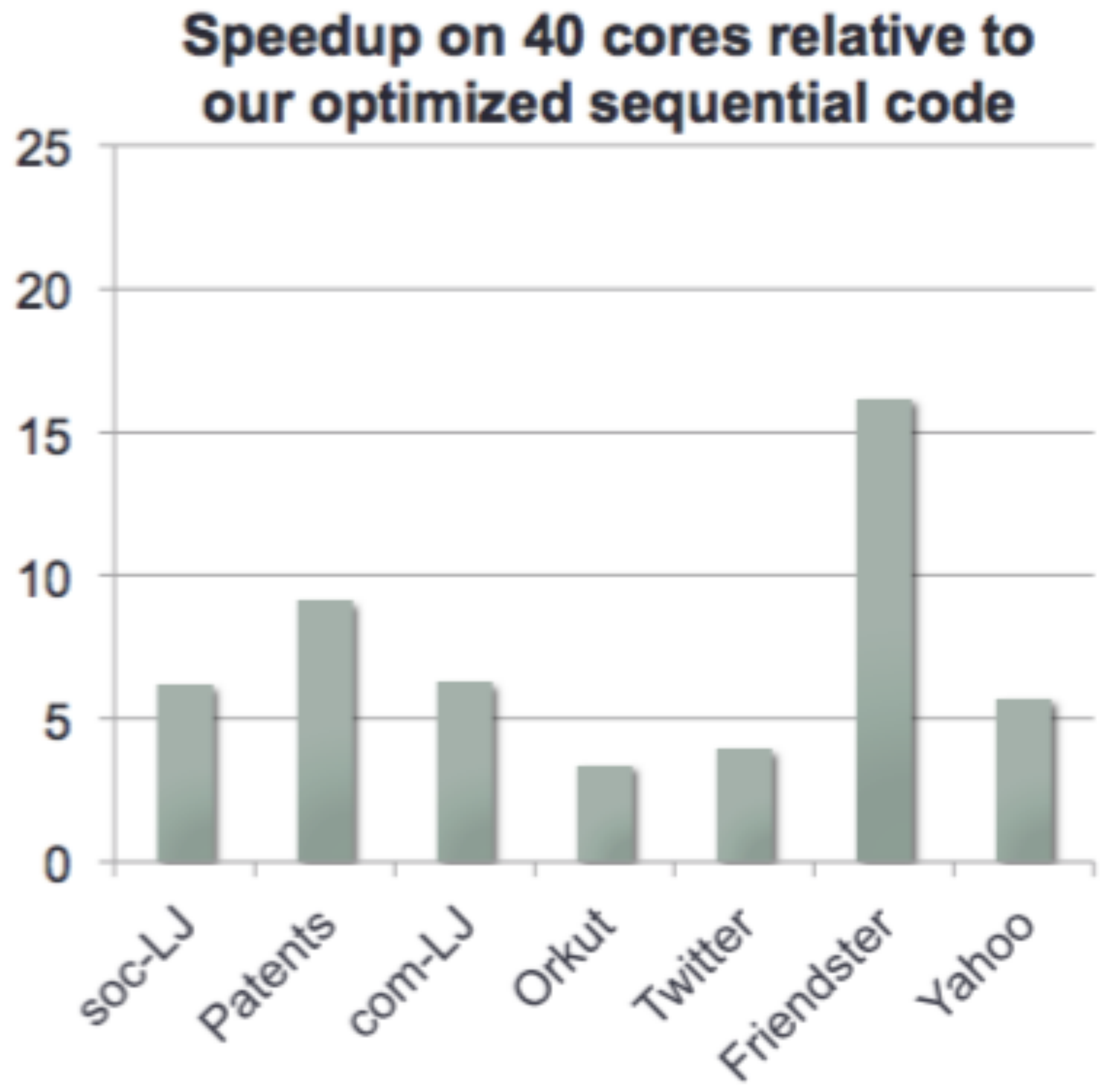
Data

Input graph	Num. vertices	Num. edges
soc-JL	4,847,571	42,851,237
cit-Patents	6,009,555	16,518,947
com-LJ	4,036,538	34,681,189
com-Orkut	3,072,627	117,185,083
Twitter	41,652,231	1,202,513,046
Friendster	124,836,180	1,806,607,135
Yahoo	1,413,511,391	6,434,561,035

J. Shun, K. Fountoulakis, F. Khorasani, M. Mahoney. Parallel Local Graph Clustering, VLDB, 2016

Performance



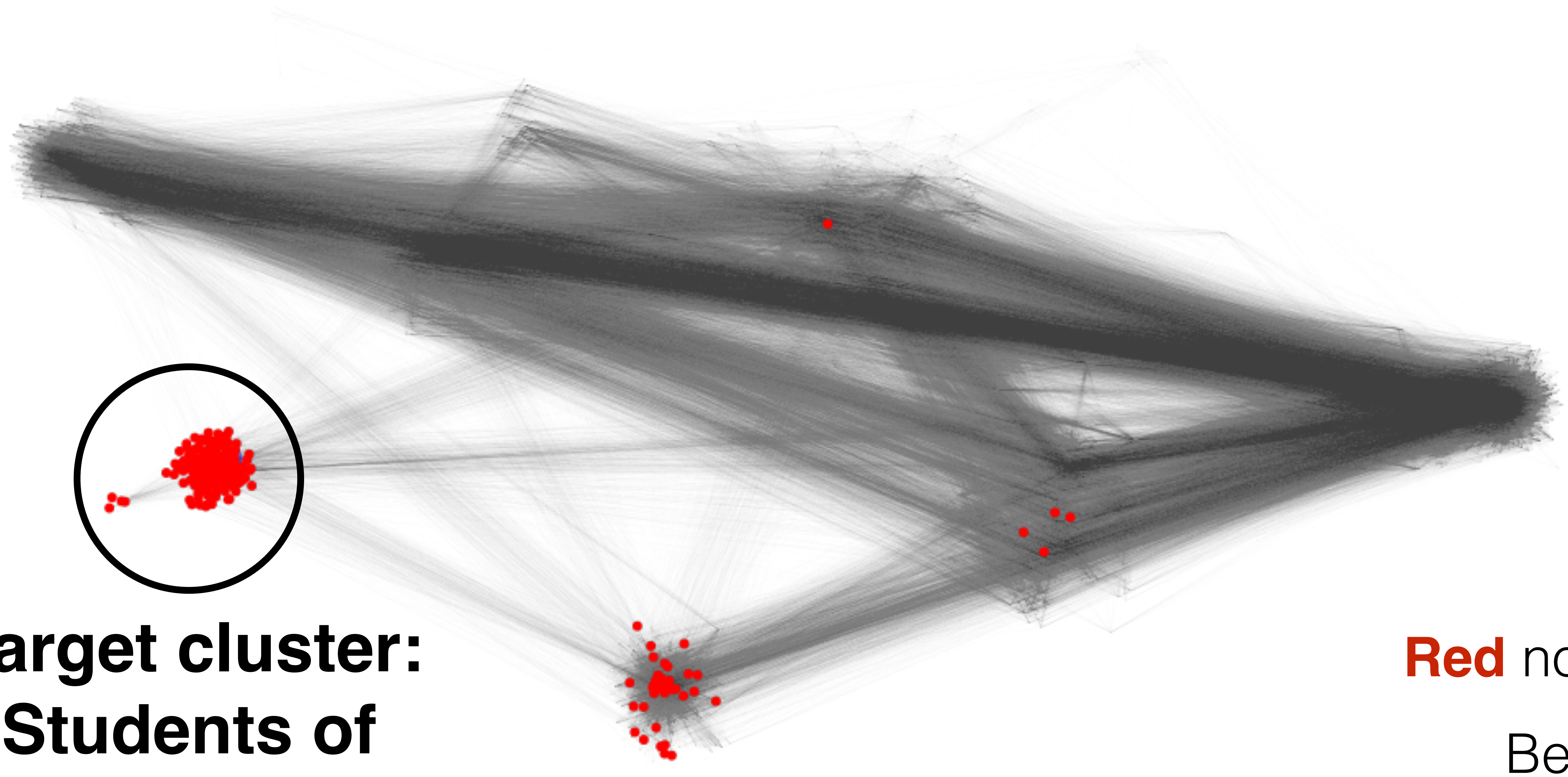


- 3-16x speed up
- Speedup is limited by small active set in some iterations and memory effects

J. Shun, K. Fountoulakis, F. Khorasani, M. Mahoney. Parallel Local Graph Clustering, VLDB, 2016

Local Flow Methods:
Capacity Releasing Diffusion

Problem: spectral diffusions might leak mass



Target cluster: Students of same major

Red nodes: output of the algorithm

Best spectral (best tuning)
Precision=0.71, Recall=0.91

Data: Facebook Johns Hopkins, A. L. Traud, P. J. Mucha and M. A. Porter, Physica A, 391(16), 2012

Solving the problem of spreading mass indiscriminately by gradual release of edge capacity

Local spectral methods

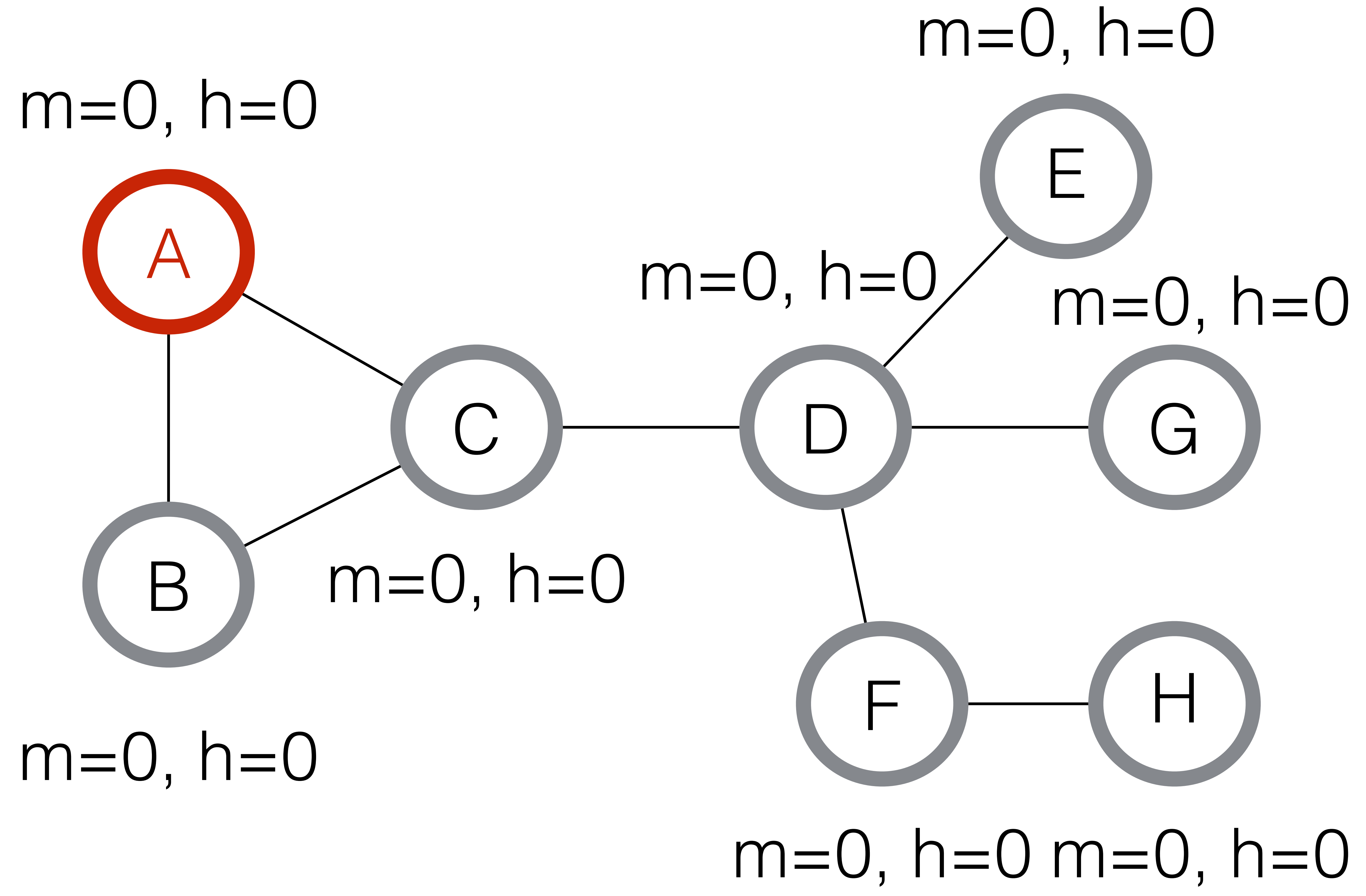
- Even distribution of the residual probability mass to neighbors

Local flow method: Capacity Releasing Diffusion

- Controls the amount of mass to be send over an edge by using the height “h” of a node
- In theory this results in bounded mass leaked outside of the target cluster
- In practice this results in much better precision and recall

Capacity Releasing Diffusion for Speed and Locality, D. Wang, K. Fountoulakis, M. Mahoney, S. Rao, ICML 2017

A new **combinatorial** diffusion: Capacity Releasing Diffusion algorithm



Maintain mass “m” and height “h” for each node

degree(v): #edges of node v

Saturated nodes: m(v) >= deg(v)

Excess mass = max(m(v) - deg(v),0)

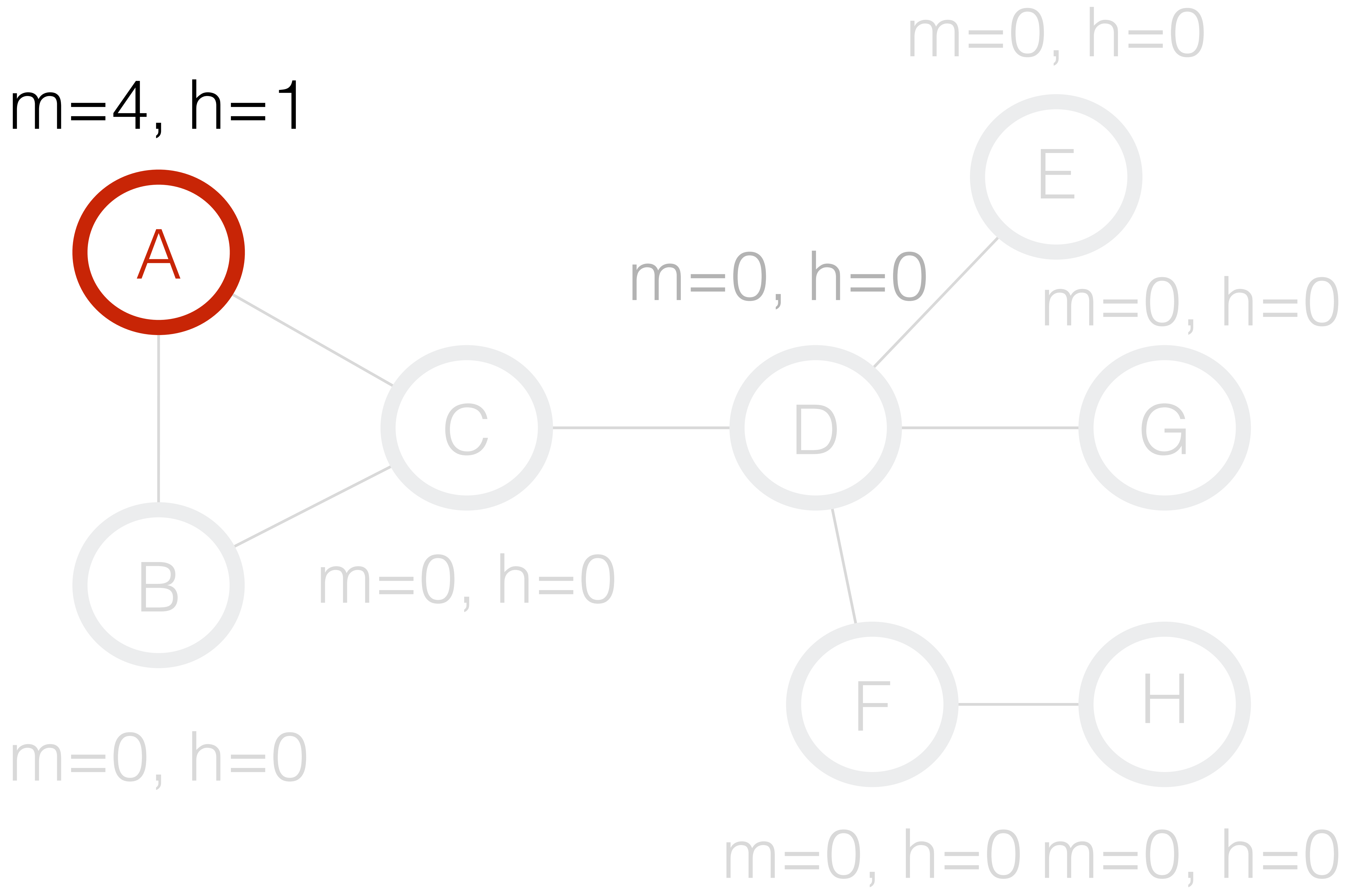
Capacity Releasing Diffusion for Speed and Locality, D. Wang, K. Fountoulakis, M. Mahoney, S. Rao, ICML 2017

A new **combinatorial** diffusion: Capacity Releasing Diffusion algorithm

degree(v): #edges of node v

Saturated nodes: m(v) >= deg(v)

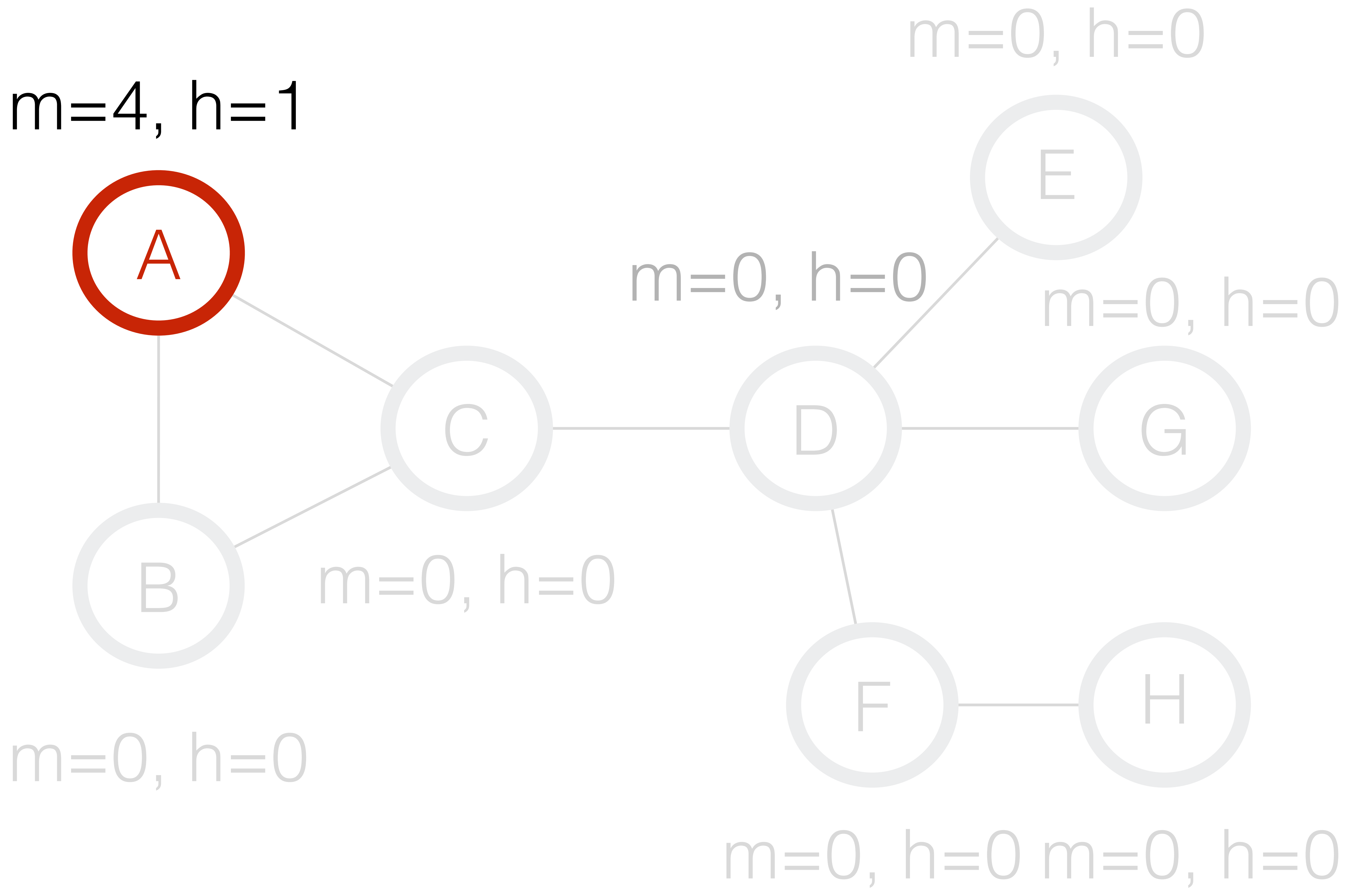
Excess mass = max(m(v) - deg(v),0)



Algorithm

```
**Overflow the seed: m(A) = 2deg(A)**
Iterate
    m(v) <= 2deg(v) for all nodes v
        Push excess mass to
        unsaturated nodes with
        lower height
    m(v) <= deg(v) for all nodes v
    Overflow: m(v) = 2m(v)
```

A new **combinatorial** diffusion: Capacity Releasing Diffusion algorithm



degree(v): #edges of node v

Saturated nodes: m(v) >= deg(v)

Excess mass = max(m(v) - deg(v),0)

Algorithm

Overflow the seed: m(A) = 2deg(A)

Iterate

Push excess mass to nodes with lower height

m(v) <= deg(v) for all nodes v

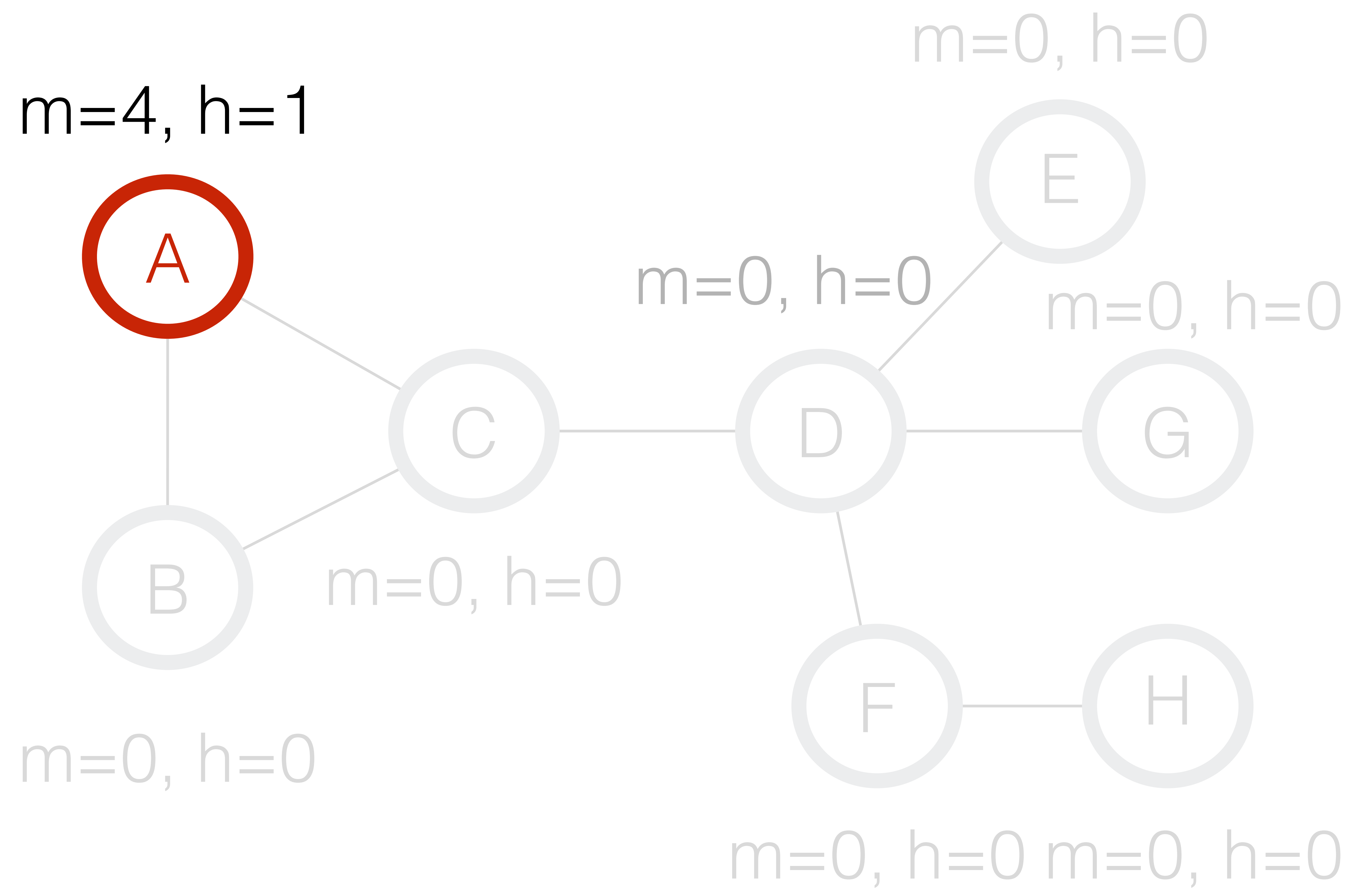
Overflow: m(v) = 2m(v)

A new **combinatorial** diffusion: Capacity Releasing Diffusion algorithm

degree(v): #edges of node v

Saturated nodes: m(v) >= deg(v)

Excess mass = max(m(v) - deg(v),0)



Algorithm

Overflow the seed: m(A) = 2deg(A)

Iterate

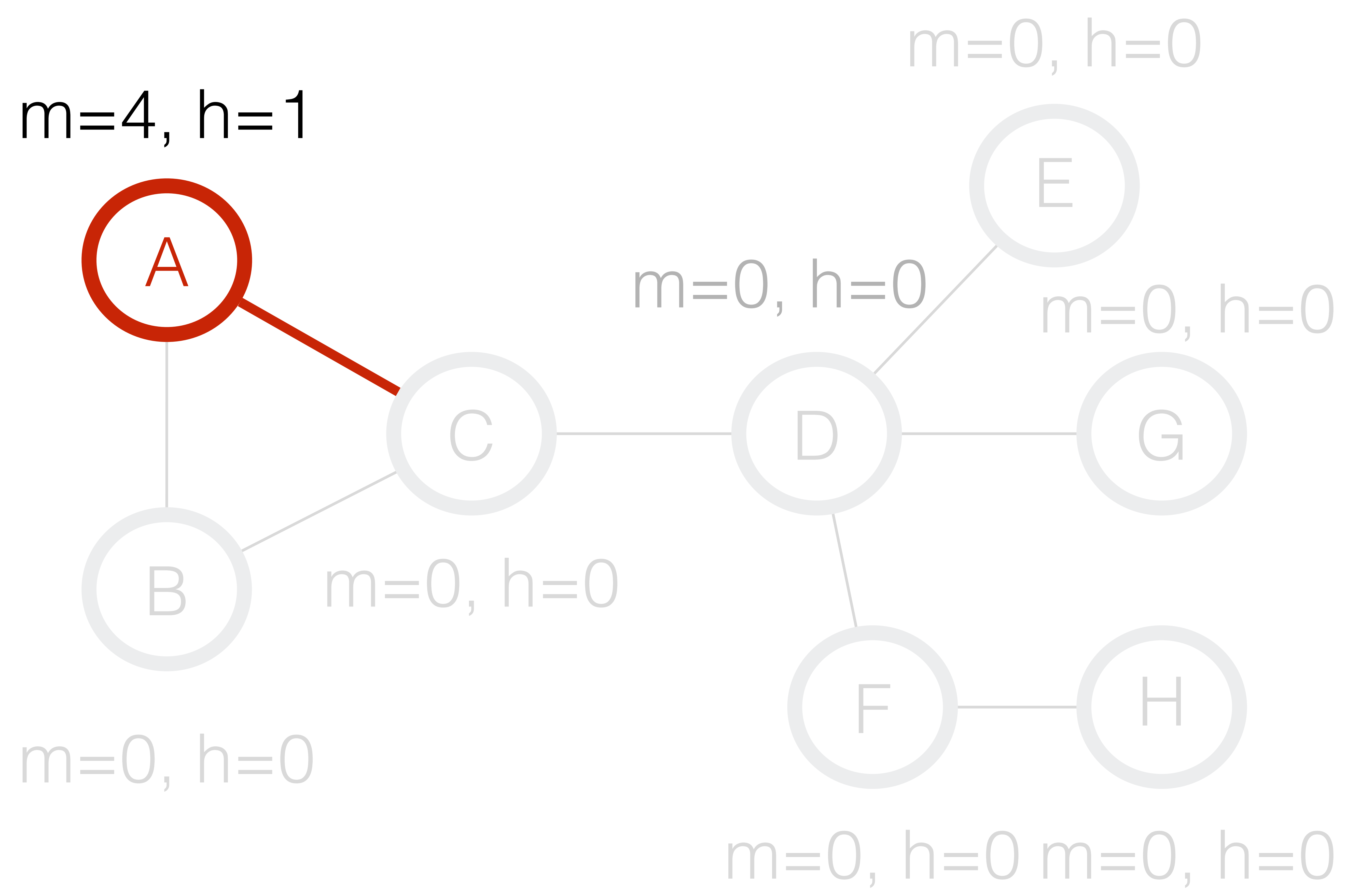
Pick node A (has excess mass)

and a neighbor of A with lower height "h"

m(v) <= deg(v) for all nodes v

Overflow: m(v) = 2m(v)

A new **combinatorial** diffusion: Capacity Releasing Diffusion algorithm



degree(v): #edges of node v

Saturated nodes: m(v) >= deg(v)

Excess mass = max(m(v) - deg(v),0)

Algorithm

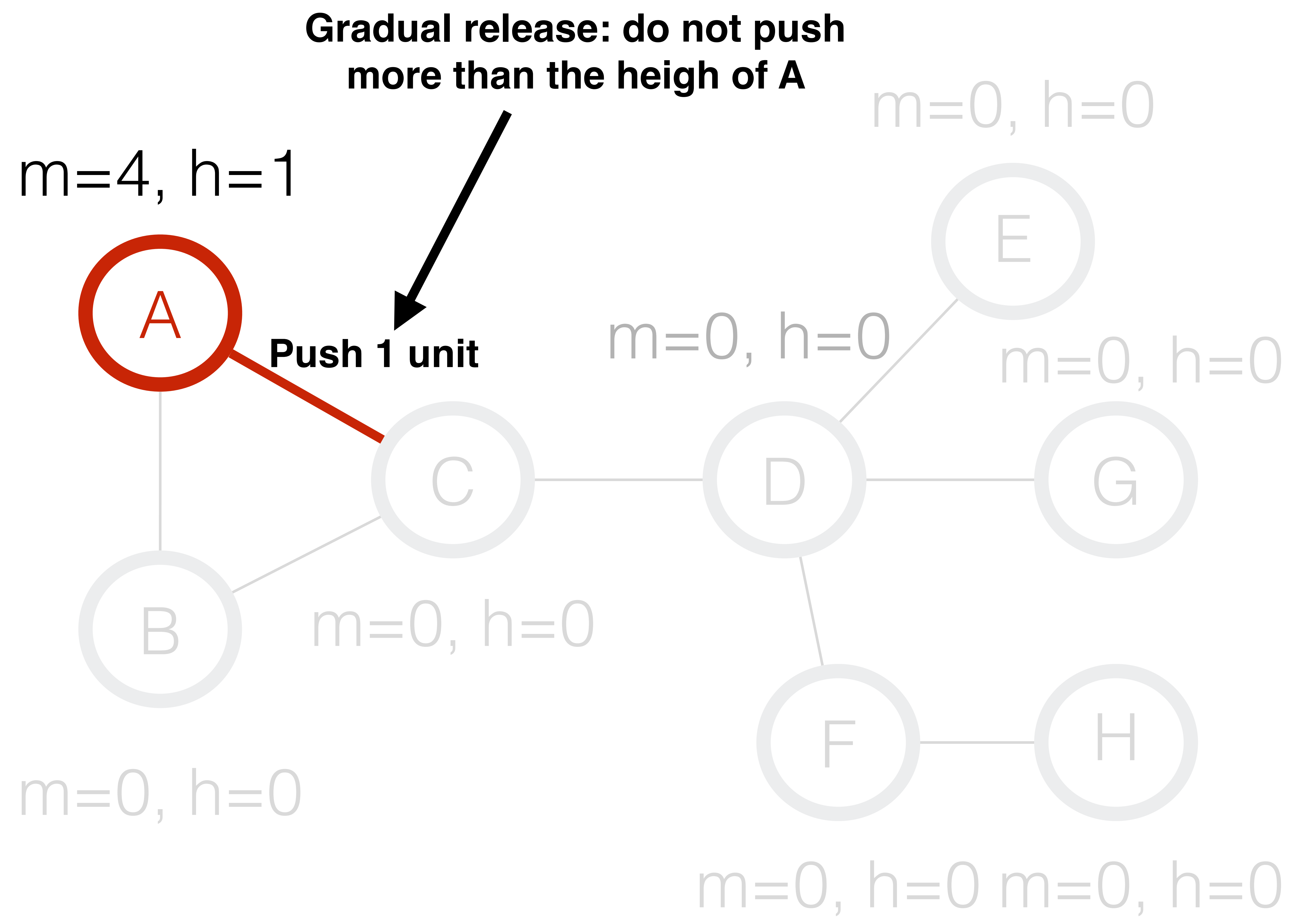
```
Overflow the seed: m(A) = 2deg(A)
Iterate
    **Pick node C**



    m(v) <= deg(v) for all nodes v
    Overflow: m(v) = 2m(v)
```

A new **combinatorial** diffusion: Capacity Releasing Diffusion algorithm

degree(v): #edges of node v

Saturated nodes: m(v) >= deg(v)

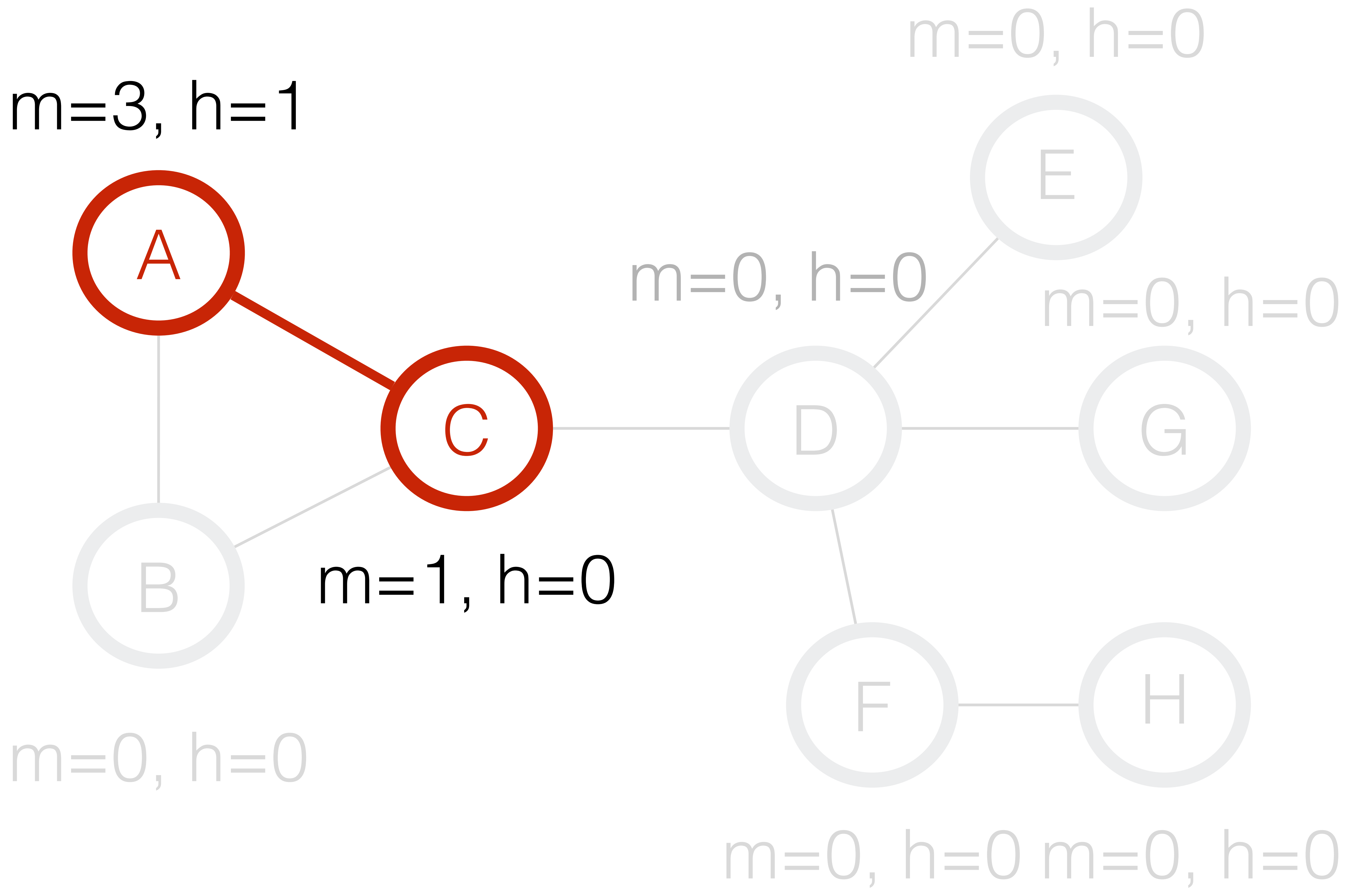
Excess mass = max(m(v) - deg(v),0)



Algorithm

```
Overflow the seed: m(A) = 2deg(A)
Iterate
    Pick node A (has excess mass)
    Push at most "h" flow to a chosen neighbor
    m(v) <= deg(v) for all nodes v
    Overflow: m(v) = 2m(v)
```

A new **combinatorial** diffusion: Capacity Releasing Diffusion algorithm



degree(v): #edges of node v

Saturated nodes: m(v) >= deg(v)

Excess mass = max(m(v) - deg(v),0)

Algorithm

Overflow the seed: m(A) = 2deg(A)

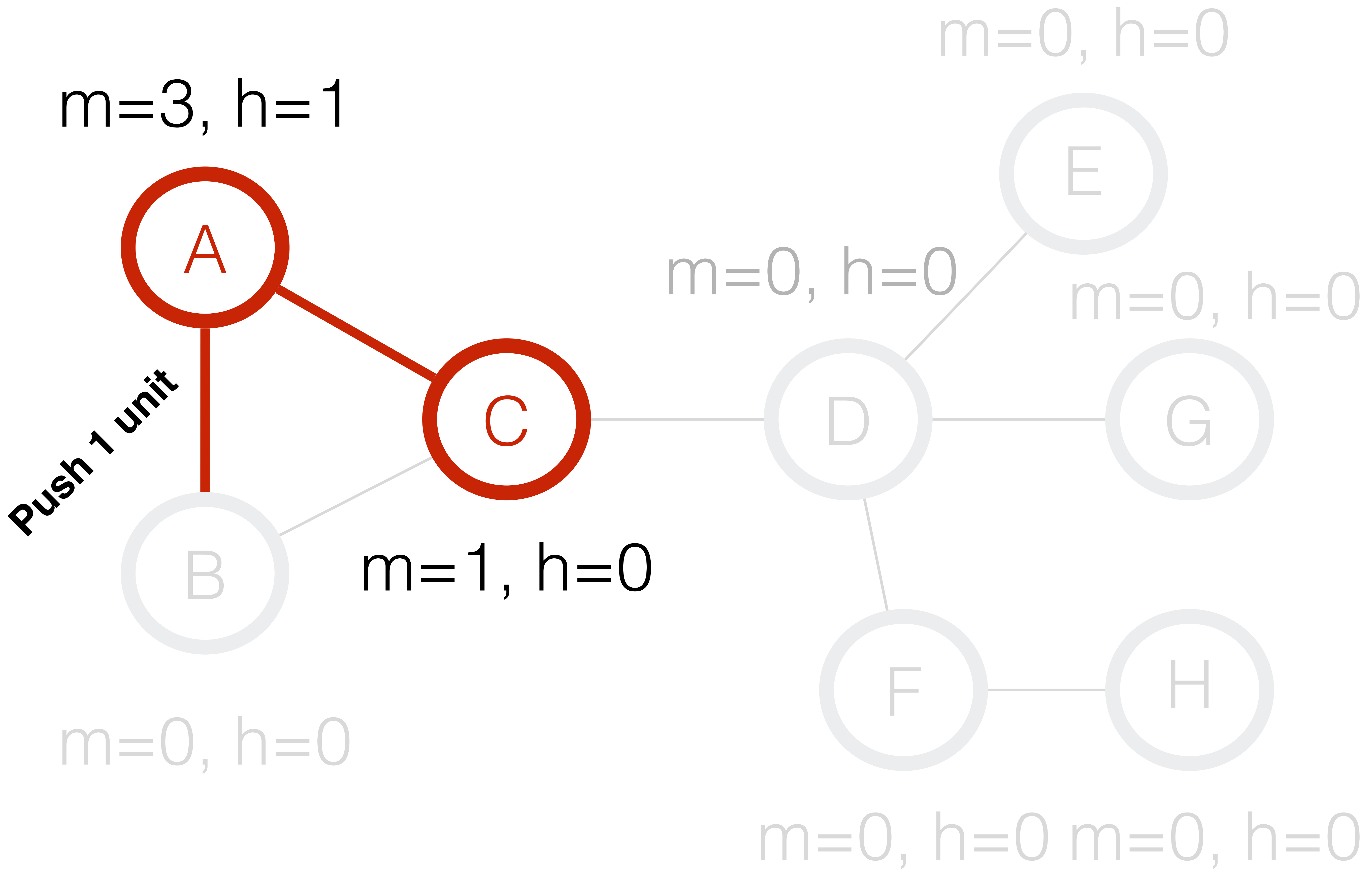
Iterate

Push excess mass to nodes with lower height

m(v) <= deg(v) for all nodes v

Overflow: m(v) = 2m(v)

A new **combinatorial** diffusion: Capacity Releasing Diffusion algorithm



degree(v): #edges of node v

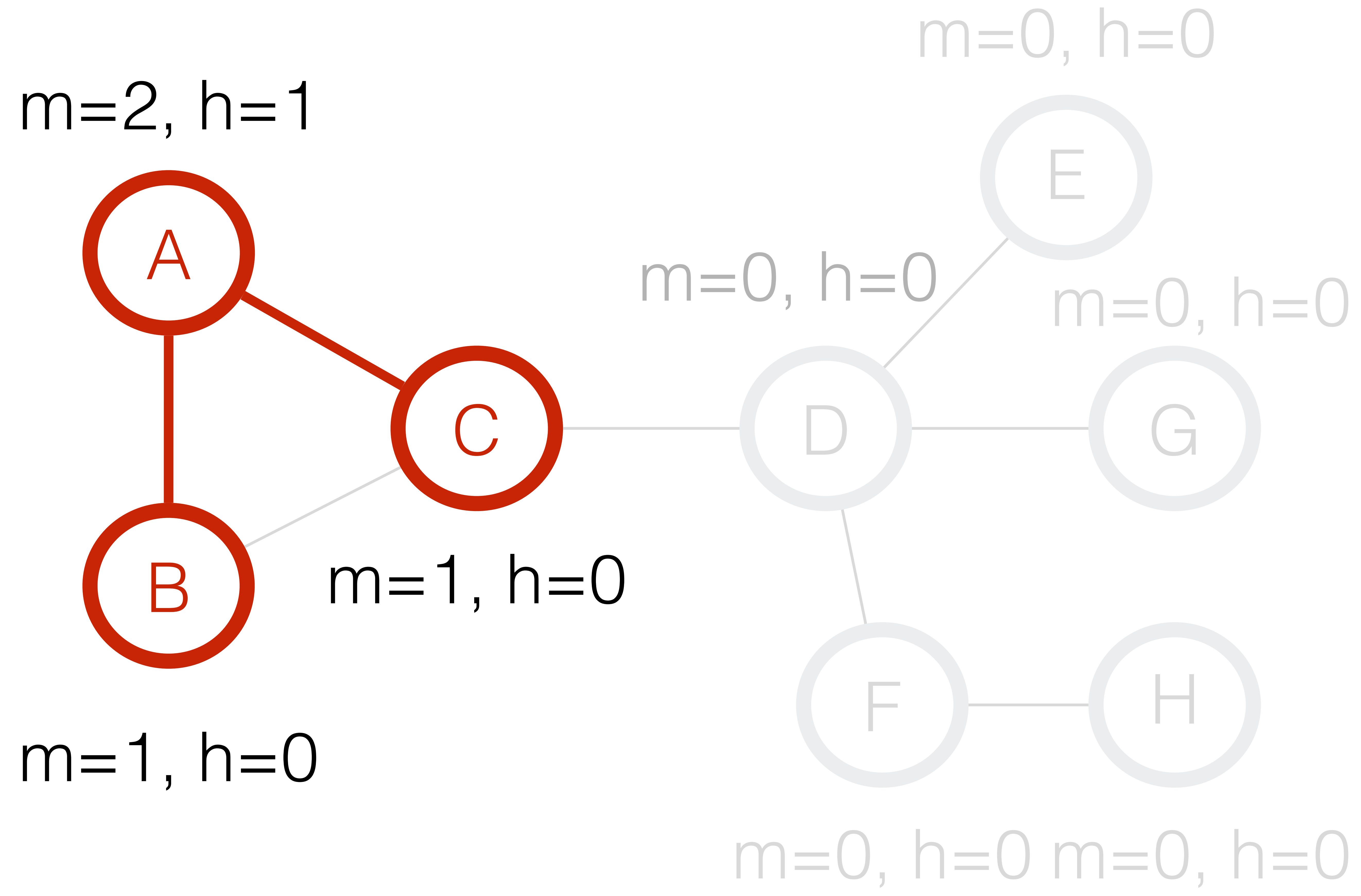
Saturated nodes: m(v) >= deg(v)

Excess mass = max(m(v) - deg(v),0)

Algorithm

```
Overflow the seed: m(A) = 2deg(A)
Iterate
    **Pick node A (has excess mass)**

    **and a new edge of node A of
    residual flow less than "h"**

    m(v) <= deg(v) for all nodes v
    Overflow: m(v) = 2m(v)
```

A new **combinatorial** diffusion: Capacity Releasing Diffusion algorithm



degree(v): #edges of node v

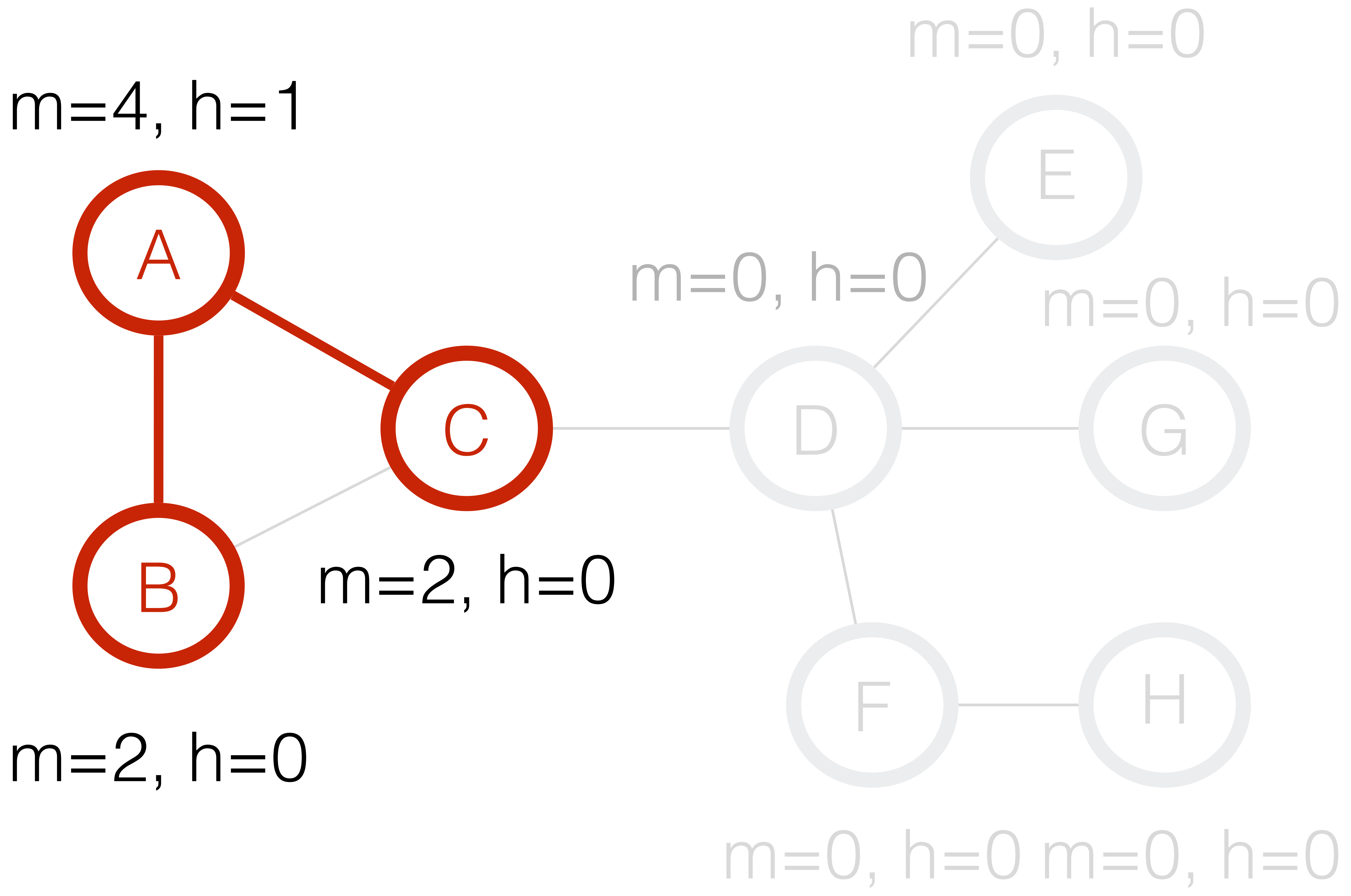
Saturated nodes: m(v) >= deg(v)

Excess mass = max(m(v) - deg(v),0)

Algorithm

```
Overflow the seed: m(A) = 2deg(A)
Iterate
    m(v) <= 2deg(v) for all nodes v
        Push excess mass to
        unsaturated nodes with
        lower height
    **m(v) <= deg(v) for all nodes v**
    Overflow: m(v) = 2m(v)
```

A new **combinatorial** diffusion: Capacity Releasing Diffusion algorithm



degree(v): #edges of node v

Saturated nodes: m(v) >= deg(v)

Excess mass = max(m(v) - deg(v),0)

Algorithm

```
Overflow the seed: m(A) = 2deg(A)
Iterate
    m(v) <= 2deg(v) for all nodes v
        Push excess mass to
        unsaturated nodes with
        lower height
    m(v) <= deg(v) for all nodes v
    **Overflow: m(v) = 2m(v)**
```

Theoretical comparison to spectral diffusions

- **Conductance of target B ("noise"): $\Phi(B) := \left(\dfrac{\text{number of edges leaving B}}{\text{sum of edges of vertices in B}}\right)$** Assuming B is the smaller part of the graph
- **Internal connectivity ("signal") of target B**

IC(B):=the minimum conductance of the subgraph induced by B

Weaker assumptions

- Theoretical bound on precision/recall needs: "signal" polylog stronger than "noise", as opposed to: quadratically stronger for spectral methods

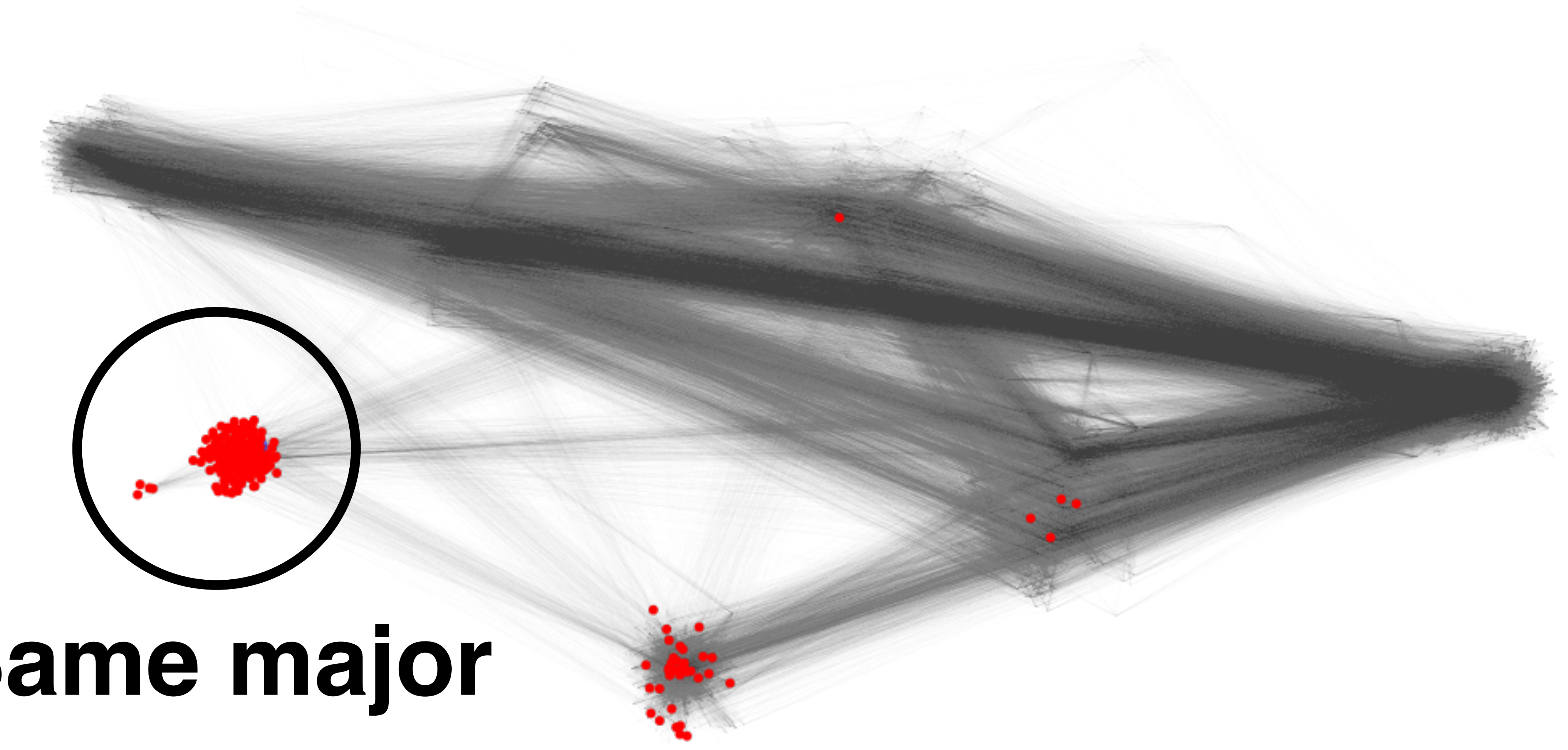
Better worst-case guarantees

- Output A satisfies $\Phi(A) <= O(\Phi(B))$, as opposed to $\Phi(A) <= O(\Phi(B)/IC(B))$

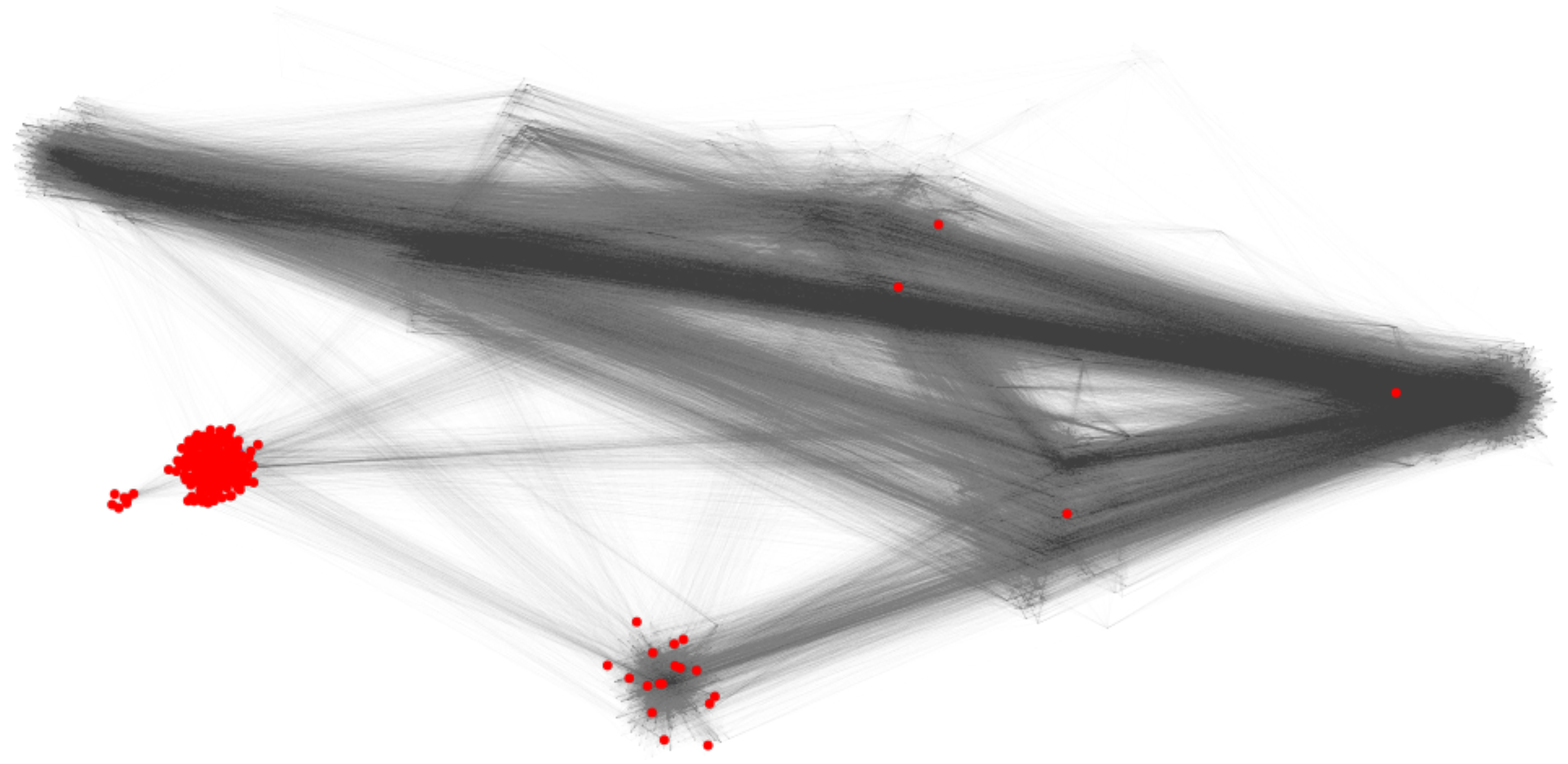
Better running time

- The running time is 1/IC(B) times faster than spectral

Example on Facebook Johns Hopkins social network

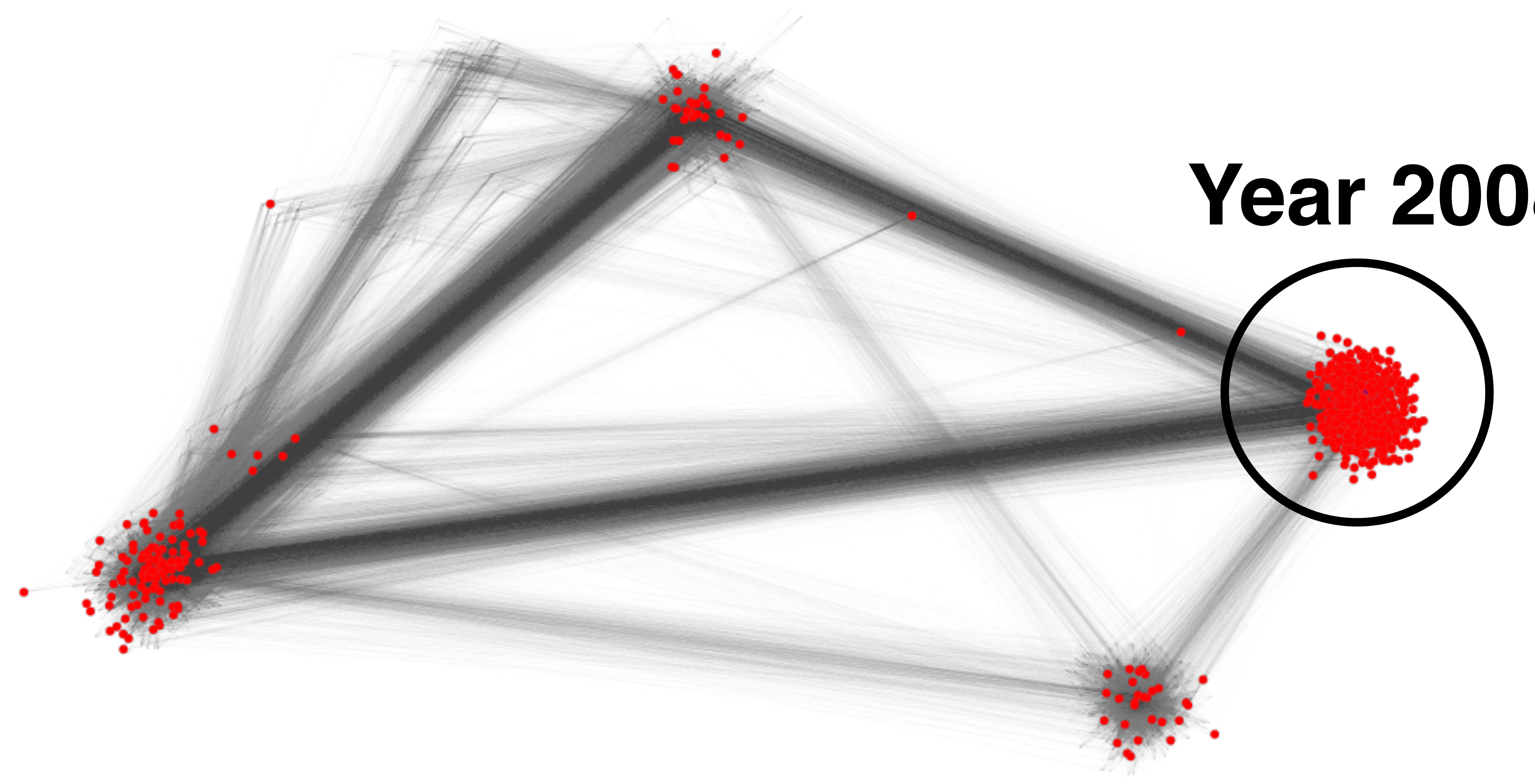


Best spectral (best tuning)
Precision=0.71, Recall=0.91
Conductance ground truth: 0.26
Conductance spectral method: 0.37

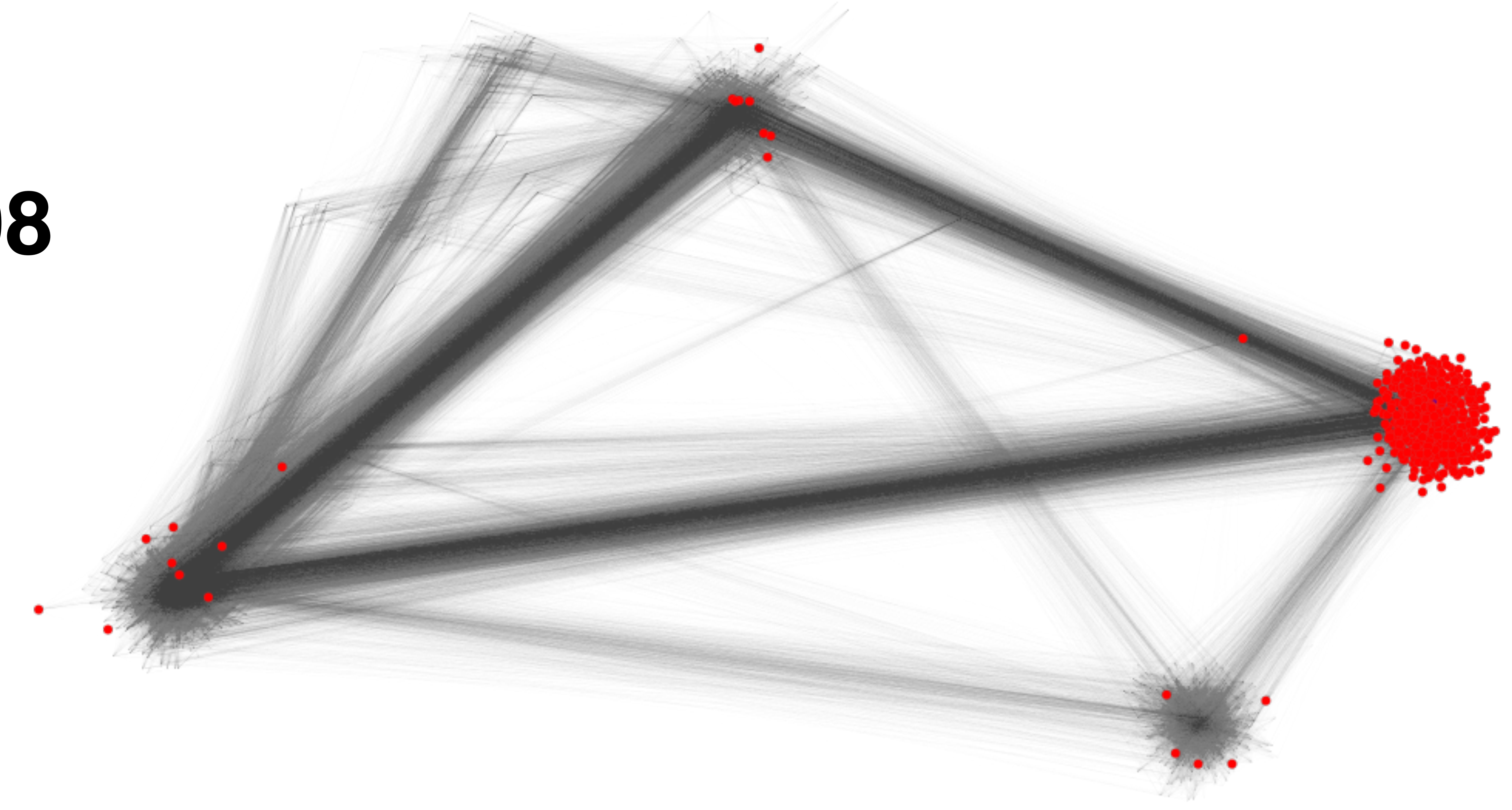


Our flow-based method
Precision=0.87, Recall=0.94
Conductance ground truth: 0.26
Conductance flow method: 0.21

Example on Facebook Colgate University social network



Best spectral (best tuning)
Precision=0.73, Recall=0.94



Our flow-based method
Precision=0.93, Recall=0.94

p-norm Local Flow Diffusion

S. Yang, D. Wang, K. Fountoulakis. p-Norm Flow Diffusion for Local Graph Clustering. Submitted. Manuscript available upon request.

What is missing in spectral and flow methods?

L1-regularized PageRank

- Propagates too much information outside a target cluster, thus leads to many false positives ☹️
- This is inherently unavoidable as long as messages are distributed evenly along edges in graph
- Node should be able to discriminate over their neighborhoods' features to diffusion mass

Capacity Releasing Diffusion

- Has better theoretical guarantees, but its complicated combinatorial operations require a lot of parameter tuning to work well in practice ☹️
- *Simpler and more interpretable algorithms are preferred*

S. Yang, D. Wang, K. Fountoulakis. p-Norm Flow Diffusion for Local Graph Clustering. Submitted. Manuscript available upon request.

Spectrum of methods

Spectral Diffusions

Combinatorial Diffusions

e.g., PageRank

easy to understand

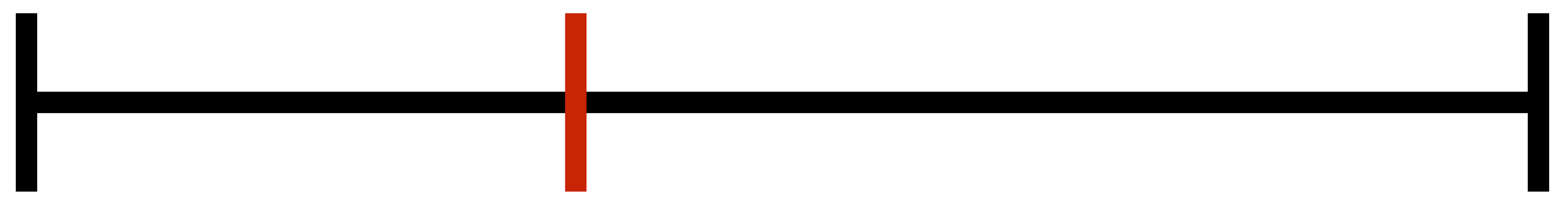
fast in practice

e.g., capacity releasing diffusion

robust to noise

S. Yang, D. Wang, K. Fountoulakis. p-Norm Flow Diffusion for Local Graph Clustering. Submitted. Manuscript available upon request.

Spectrum of methods



Spectral diffusions

Combinatorial diffusions

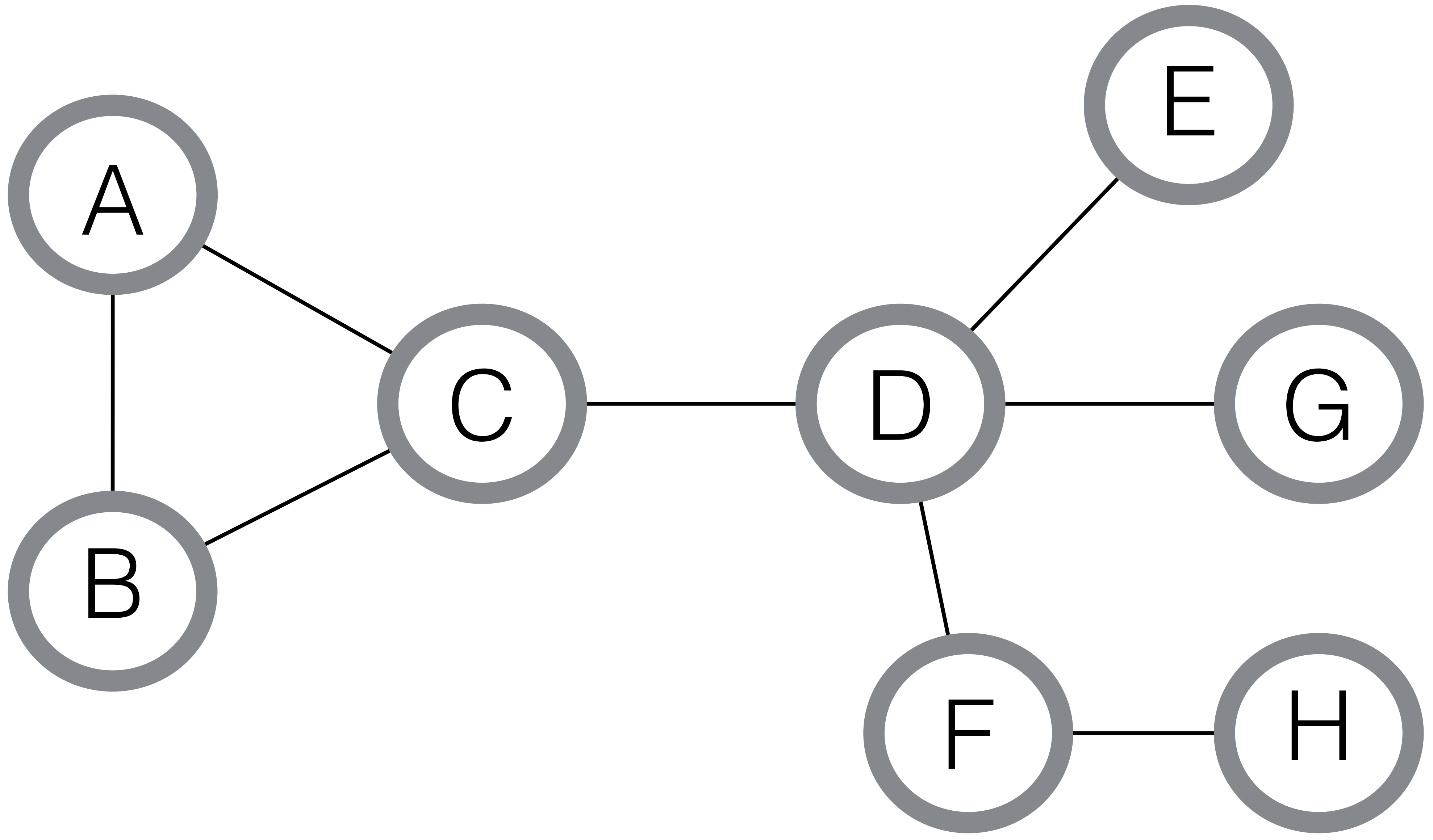
p-norm flow diffusion

- p-norm message passing is a family of convex optimization problems that characterizes the trade-off between spectral and combinatorial diffusions.

- This allows us to define methods that are the best of both worlds.

S. Yang, D. Wang, K. Fountoulakis. p-Norm Flow Diffusion for Local Graph Clustering. Submitted. Manuscript available upon request.

Some definitions - incidence matrix



Incidence matrix B

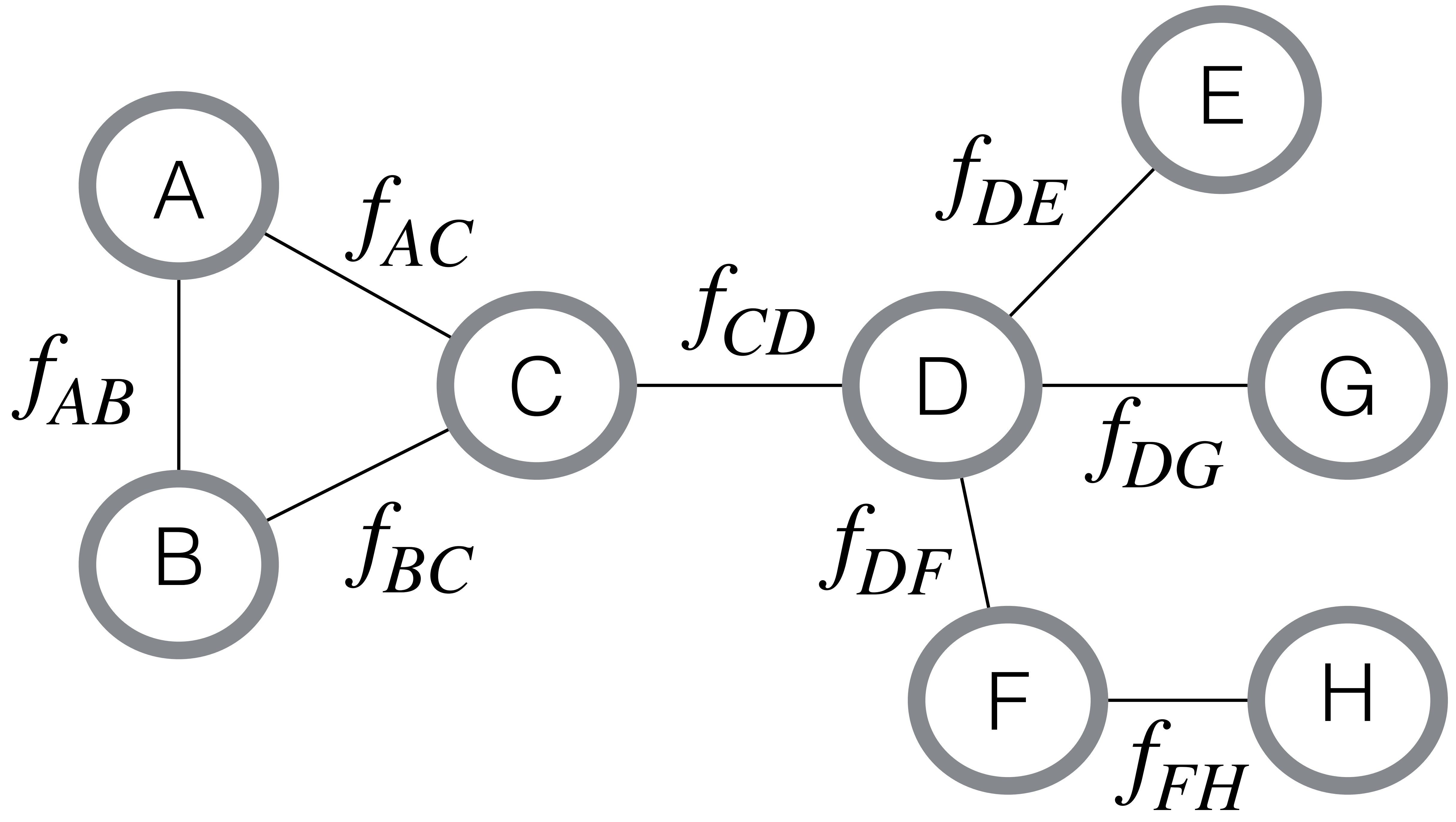
	A	B	C	D	E	F	G	H
A-B	1	-1						
A-C	1		-1					
B-C		1	-1					
C-D			1	-1				
D-E				1	-1			
D-F				1		-1		
D-G				1			-1	
F-H						1		-1

- Ordering of edges and direction is arbitrary

S. Yang, D. Wang, K. Fountoulakis. p-Norm Flow Diffusion for Local Graph Clustering. Submitted. Manuscript available upon request.

Some definitions - flow variables

- Let f be a vector and each component of f corresponds to an edge, for example:



- The magnitude of f is the amount of flow that passes through an edge
- The sign of f is the direction of flow

Some definitions - net flow

- Let Δ be a non-negative vector, each component of Δ indicates the initial mass at a node.
- $B^T f$ is a vector that captures the net flow on a node.
- $B^T f + \Delta$ indicates the net mass on every node.

S. Yang, D. Wang, K. Fountoulakis. p-Norm Flow Diffusion for Local Graph Clustering. Submitted. Manuscript available upon request.

Node capacities

- We will require that each node has capacity equal to its degree d_i
- We will say that the initial mass Δ has been diffused, when the net mass on each node is less than its capacity:

$$\underbrace{B^T f + \Delta}_{\text{net mass per node}} \leq \underbrace{d}_{\text{capacity per node}}$$

Diffusion as an optimization formulation

- Out of all possible flows f that satisfy the capacities we are interested in the one with minimum L_p norm, where $p \in [2,\infty)$.

$$\text{minimize } \|f\|_p$$
$$\text{subject to: } B^T f + \Delta \leq d$$

Relation to other methods

- For $p = 2$ the dual of the 2-norm flow diffusion problem is

$$\text{minimize } \frac{1}{2}\|Bx\|_2^2 - x^T\Delta + \|Dx\|_1$$

- which is a regularized spectral problem, very similar ℓ_1-regularized PageRank.

- For $p \to \infty$ the dual of the ∞-norm flow diffusion problem is

$$\text{minimize } \|Bx\|_1 - x^T\Delta + \|Dx\|_1$$

- which is a regularized min-cut problem, very similar to the so-called flow-improve methods

S. Yang, D. Wang, K. Fountoulakis. p-Norm Flow Diffusion for Local Graph Clustering. Submitted. Manuscript available upon request.

Rounding

- In practice we solve the dual of the p-norm flow problem

$$\begin{aligned}\text{minimize} \quad & -x^T\Delta + \|Dx\|_1 \\ \text{subject to:} \quad & \|Bx\|_q \le 1 \\ & x \ge 0\end{aligned}$$

- So we have direct access to the dual variables

- Sort the dual variables in descending order
- Output the prefix set with smallest conductance.

S. Yang, D. Wang, K. Fountoulakis. p-Norm Flow Diffusion for Local Graph Clustering. Submitted. Manuscript available upon request.

p-norm MP: theoretical guarantee

- Conductance of target B is $\Phi(B) := \left(\frac{\text{number of edges leaving } B}{\text{sum of edges of vertices in } B}\right)$

Assuming B is the smaller part of the graph

- If we set $\mu = \mathcal{O}(\|\Delta\|_1^{(1-p)/p})$ and assume the input source Δ overlaps well with B, then output A satisfies

$$\Phi(A) \leq \mathcal{O}(\Phi(B)^{1-1/p})$$

For comparison:

- L1-reg. PageRank: $\Phi(A) \leq \mathcal{O}(\Phi(B)/IC(B))$ under assumption of strong IC
- Capacity Releasing Diffusion: $\Phi(A) \leq \mathcal{O}(\Phi(B))$ under assumption of good IC

S. Yang, D. Wang, K. Fountoulakis. p-Norm Flow Diffusion for Local Graph Clustering. Submitted. Manuscript available upon request.

p-norm network flow diffusions - algorithm

-Simple randomized coordinate descent

-Running time $$\mathcal{O}\left(\frac{|\Delta|}{\gamma}\left(\frac{|\Delta|}{\epsilon}\right)^{1-2/p}\log\frac{1}{\epsilon}\right)$$

- $|\Delta|$ represents the magnitude of the input source.
-γ is the strong convexity parameter of the dual problem.
-ϵ is the required accuracy

-$p = 2$ gives the usual running time for spectral methods $\tilde{\mathcal{O}}(|\Delta|)$
-$p \to \infty$ gives $\tilde{\mathcal{O}}(|\Delta|^2/\epsilon)$

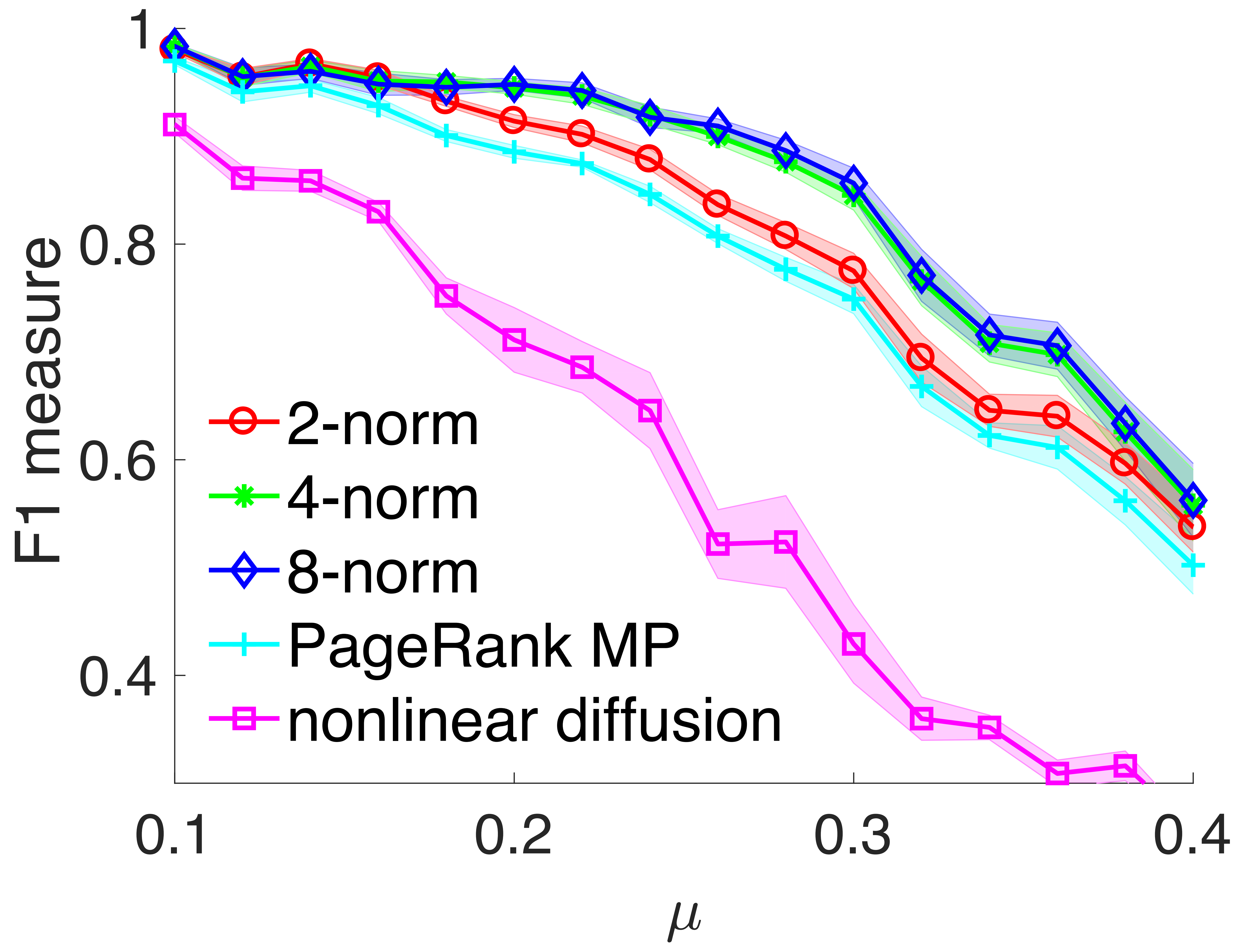
S. Yang, D. Wang, K. Fountoulakis. p-Norm Flow Diffusion for Local Graph Clustering. Submitted. Manuscript available upon request.

p-norm flow diffusion: empirical performance

- Performs very well on theoretically hard instances, e.g., nearly exact recovery on the Cartesian product of a binary tree and a line graph.

- Plot shows results on LFR synthetic model, basically a stochastic block model
- μ is a parameter that controls noise, the higher the more noise.

- Significantly outperforms PageRank and nonlinear diffusion on real world social and biological networks.



S. Yang, D. Wang, K. Fountoulakis. p-Norm Flow Diffusion for Local Graph Clustering. Submitted. Manuscript available upon request.

Open problems regarding Lp local flow.

- Currently we do not have any bounds on false positives and false negatives.
- Currently we apply black-box optimization algorithms to the dual problems. Although we can show that these algorithms are strongly local in terms of running time, we can probably do better by early termination or other specialized algorithms
- Currently, our conductance result requires sufficient overlap with the target cluster. What if we have a single node inside the target cluster? This problem has been solved for Capacity Releasing Diffusion, can we do the same for Lp local flow?
- Finally, it is known that Lp problems with equality constraints can be solved in nearly linear time. In our case we have an Lp problem with inequality constraints, can we still solve this in nearly linear time?

S. Yang, D. Wang, K. Fountoulakis. p-Norm Flow Diffusion for Local Graph Clustering. Submitted. Manuscript available upon request.

Software

Software

LocalGraphClustering on GitHub

- Written in Python with C++ routines when required

- Demonstrations on social and bioinformatics networks

- Multiple Python notebooks with numerous examples and graph visualizations

- Video presentations

- 12 methods and pipelines

Thank you!