Parallel Local Graph Clustering

Kimon Fountoulakis, joint work with J. Shun, X. Cheng, F. Roosta-Khorasani, M. Mahoney, D. Gleich
University of California Berkeley and Purdue University

Based on

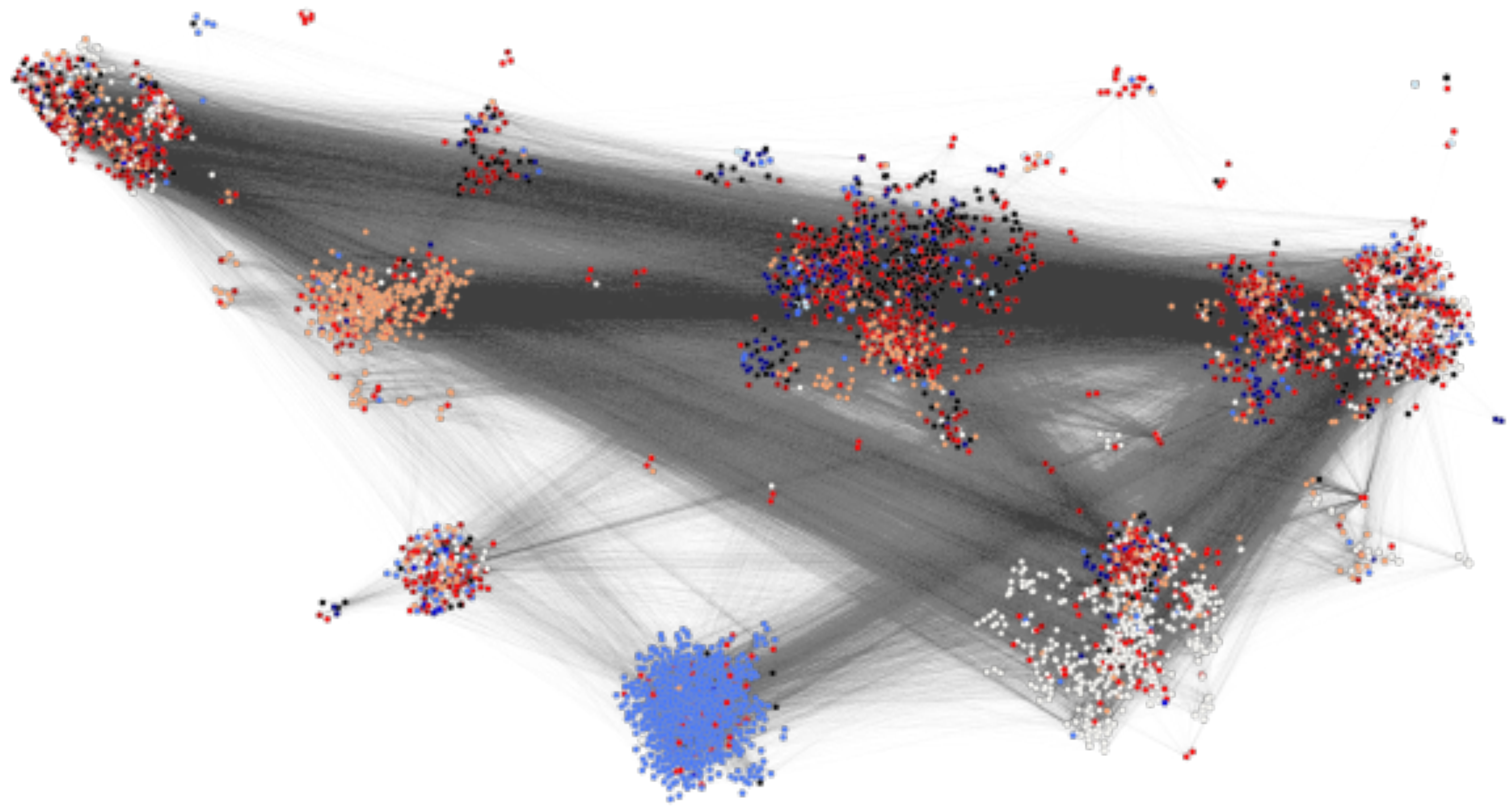
J. Shun, F. Roosta-Khorasani, KF, M. Mahoney. Parallel local graph clustering, VLDB 2016.

KF, X. Cheng, J. Shun, F. Roosta-Khorasani, M. Mahoney. Exploiting optimization for local graph clustering, arXiv:1602.01886v1.

KF, D. Gleich, M. Mahoney. An optimization approach to locally-biased graph algorithms, arXiv:1607.04940.

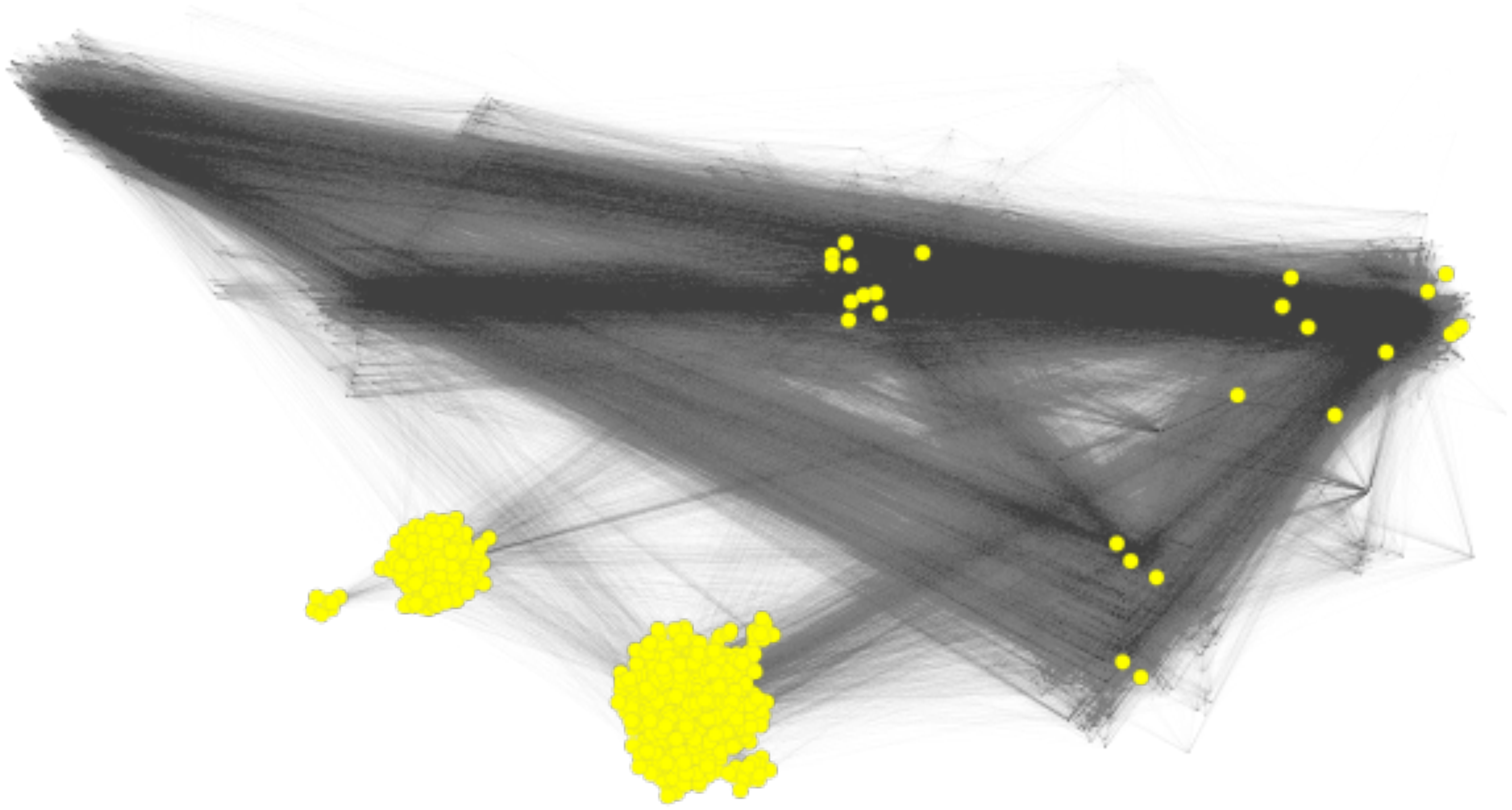
Local graph clustering: motivation

Facebook social network: colour denotes class year



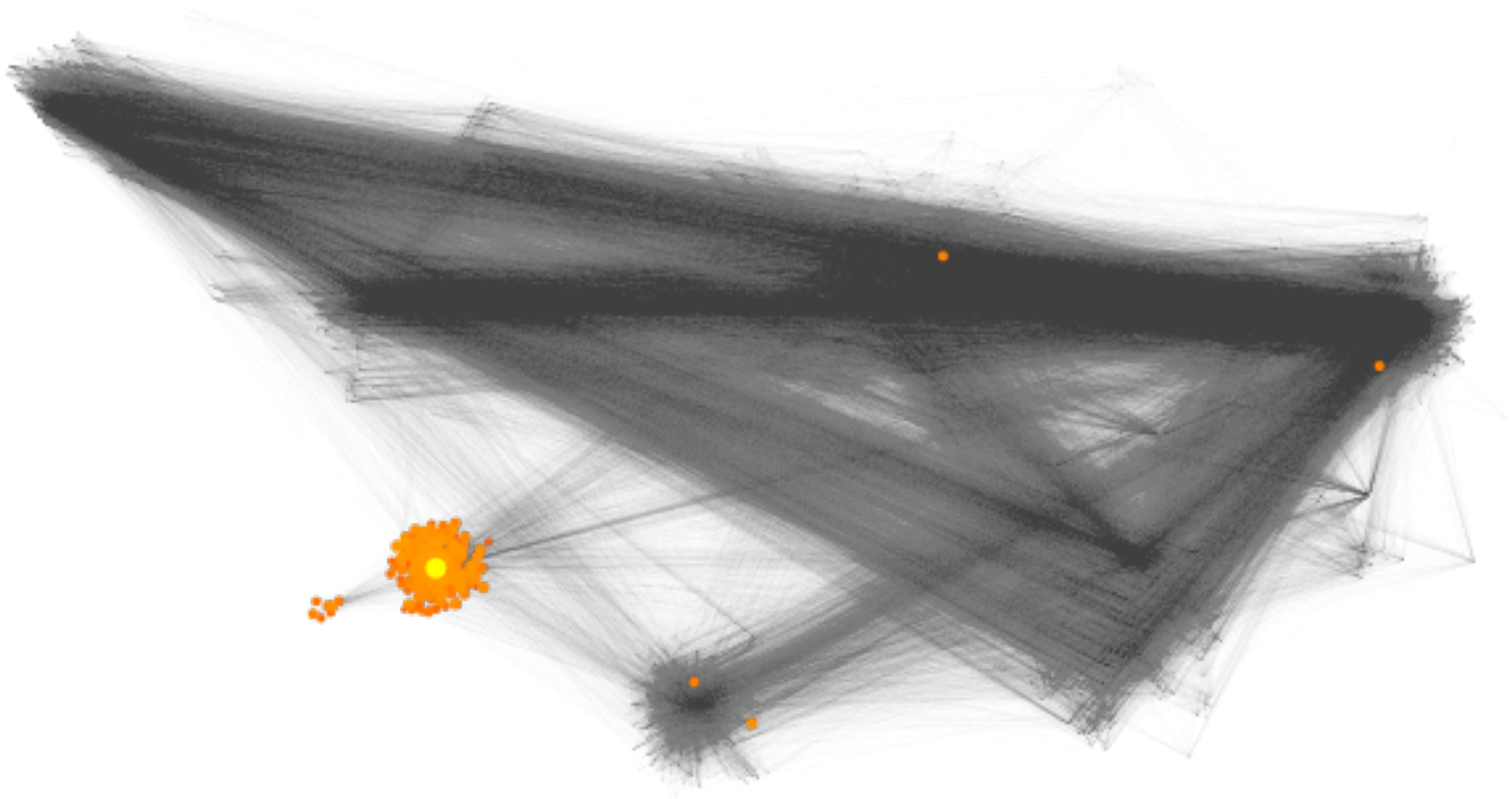
Data: Facebook John Hopkins, A. L. Traud, P. J. Mucha and M. A. Porter, Physica A, 391(16), 2012

Normalized cuts: finds 20% of the graph



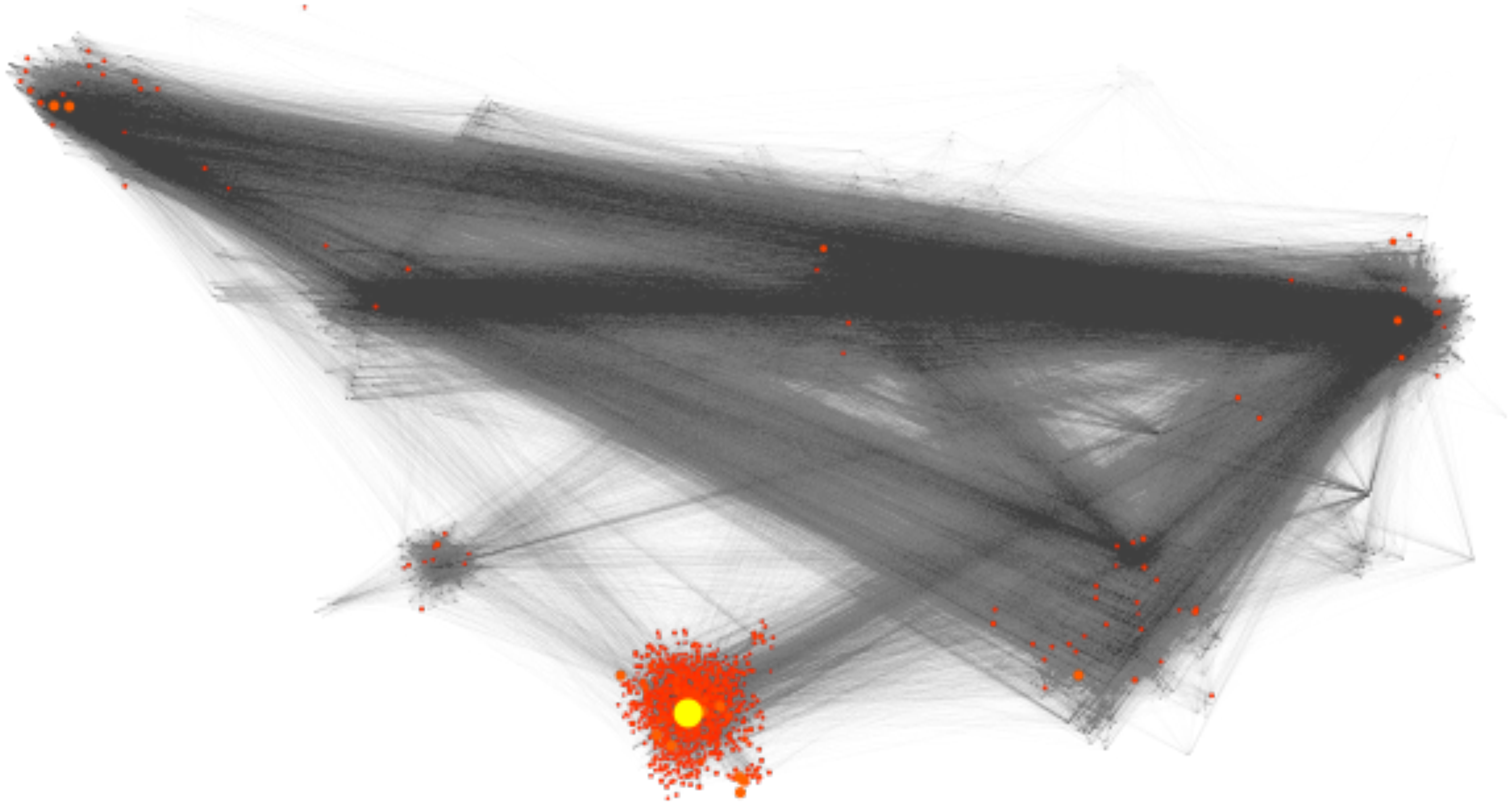
Data: Facebook John Hopkins, A. L. Traud, P. J. Mucha and M. A. Porter, Physica A, 391(16), 2012

Local graph clustering: finds 3% of the graph



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

Local graph clustering: finds 17% of the graph



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

Current algorithms and running time

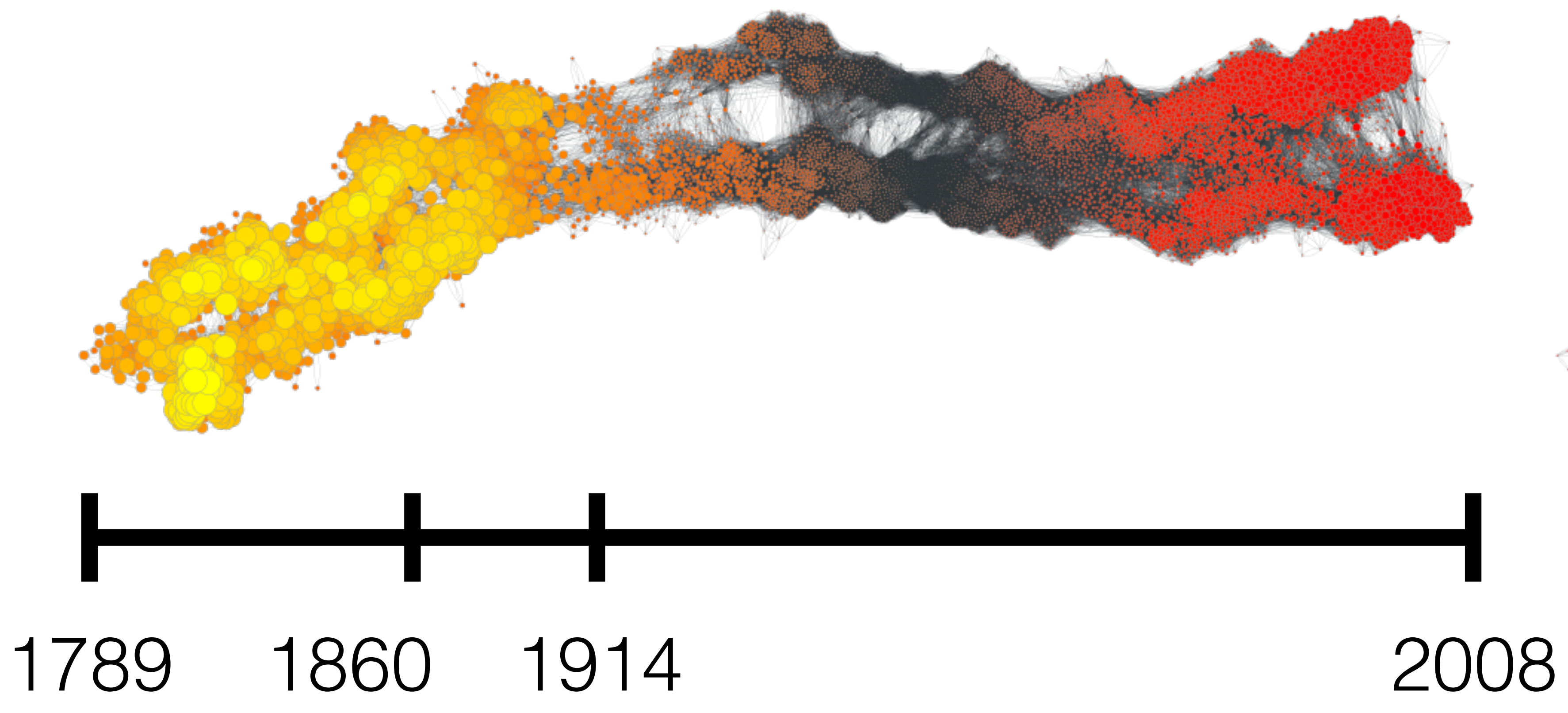
Global, weakly and strongly local methods

Global methods: O(volume of graph)
- The workload depends on the size of the graph

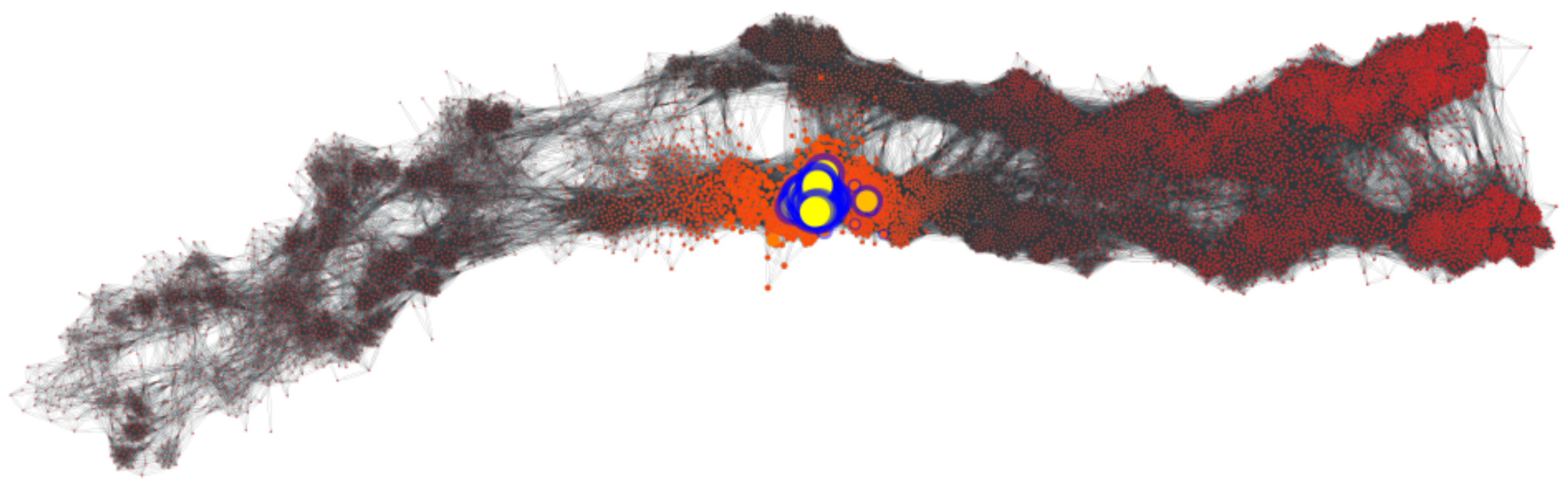
Weakly local methods: O(volume of graph)
- A seed set of nodes is given
- The solution is locally biased to the input seed set
- The workload depends on the size of the graph

Strongly local methods: O(volume of output cluster)
- A seed set of nodes is given
- The solution is locally biased to the input seed set

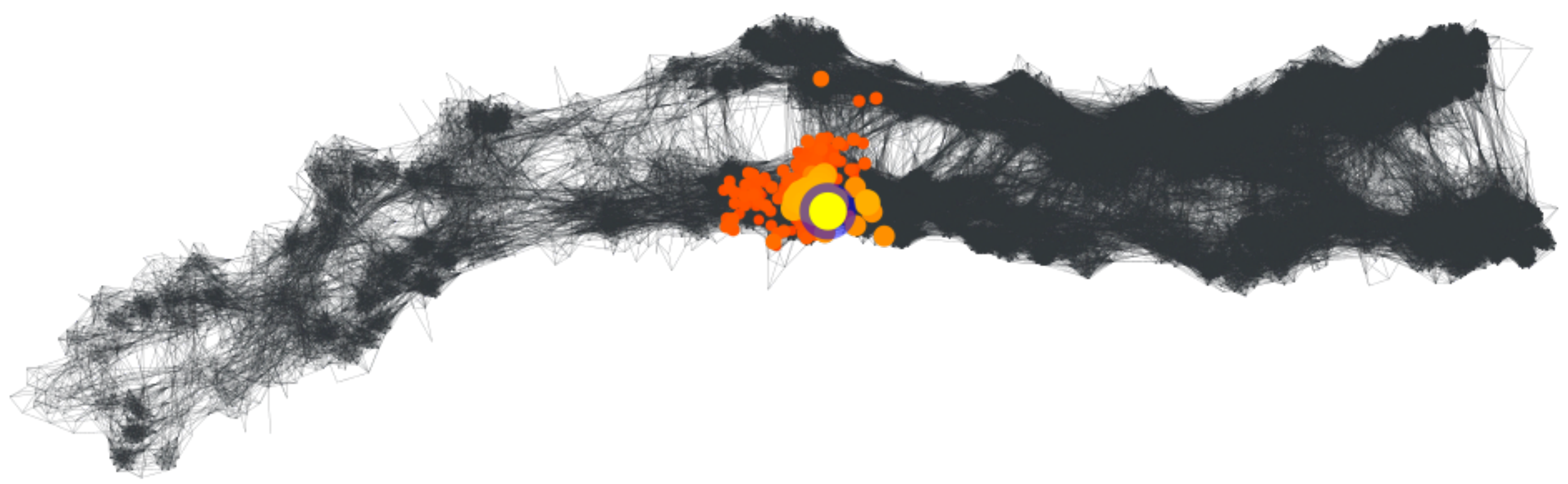
Global, weakly and strongly local methods



Global



Weakly local



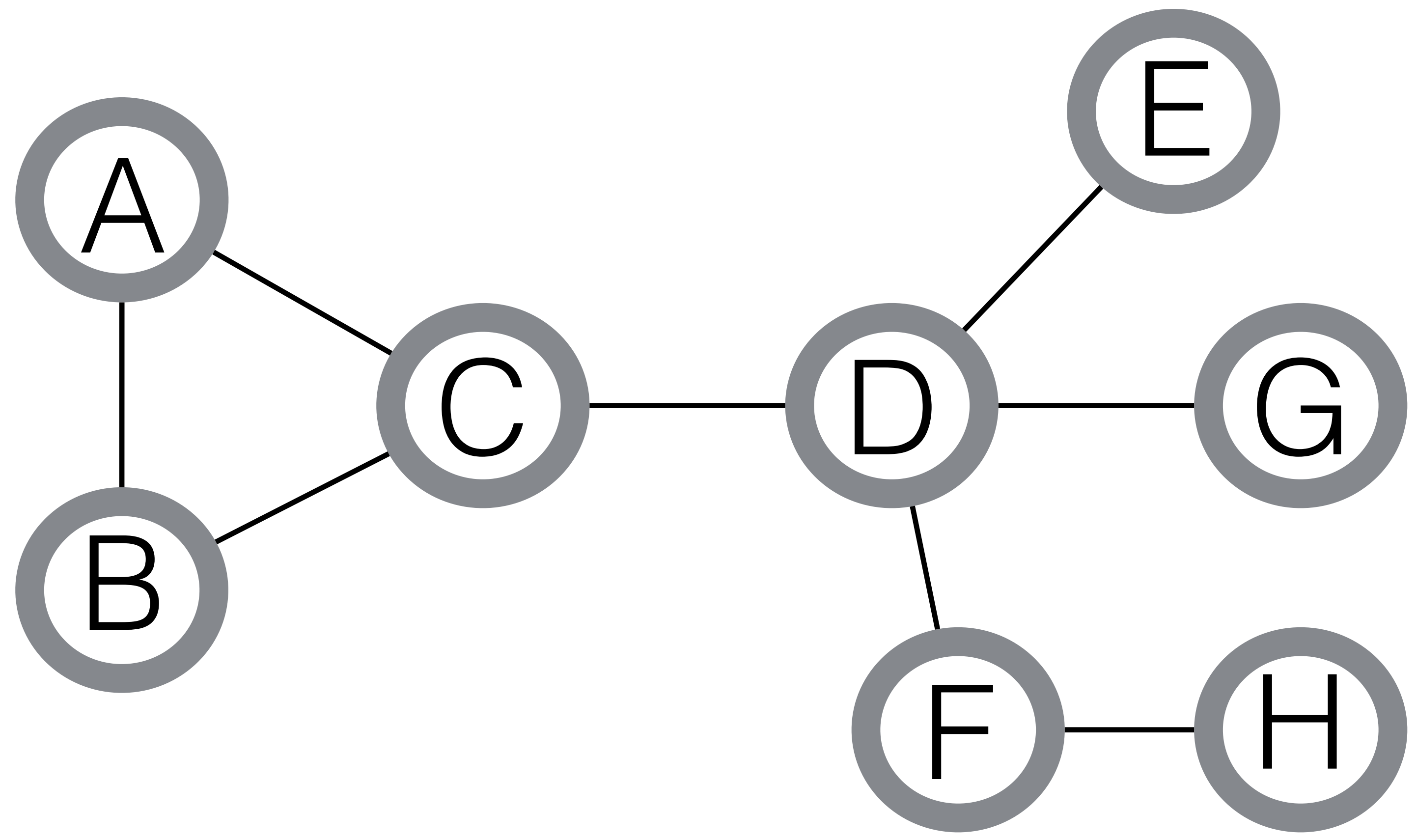
Strongly local

Data: US Senate, P. Mucha, T. Richardson, K. Macon, M. Porter and J. Onnela, Science, vol. 328, no. 5980, pp. 876-878, 2010

Cluster quality

We measure cluster quality using

$$\text{Conductance} := \frac{\text{number of edges leaving cluster}}{\text{sum of degrees of vertices in cluster}}$$



Conductance({A,B}) = 2/(2 + 2) = 1/2

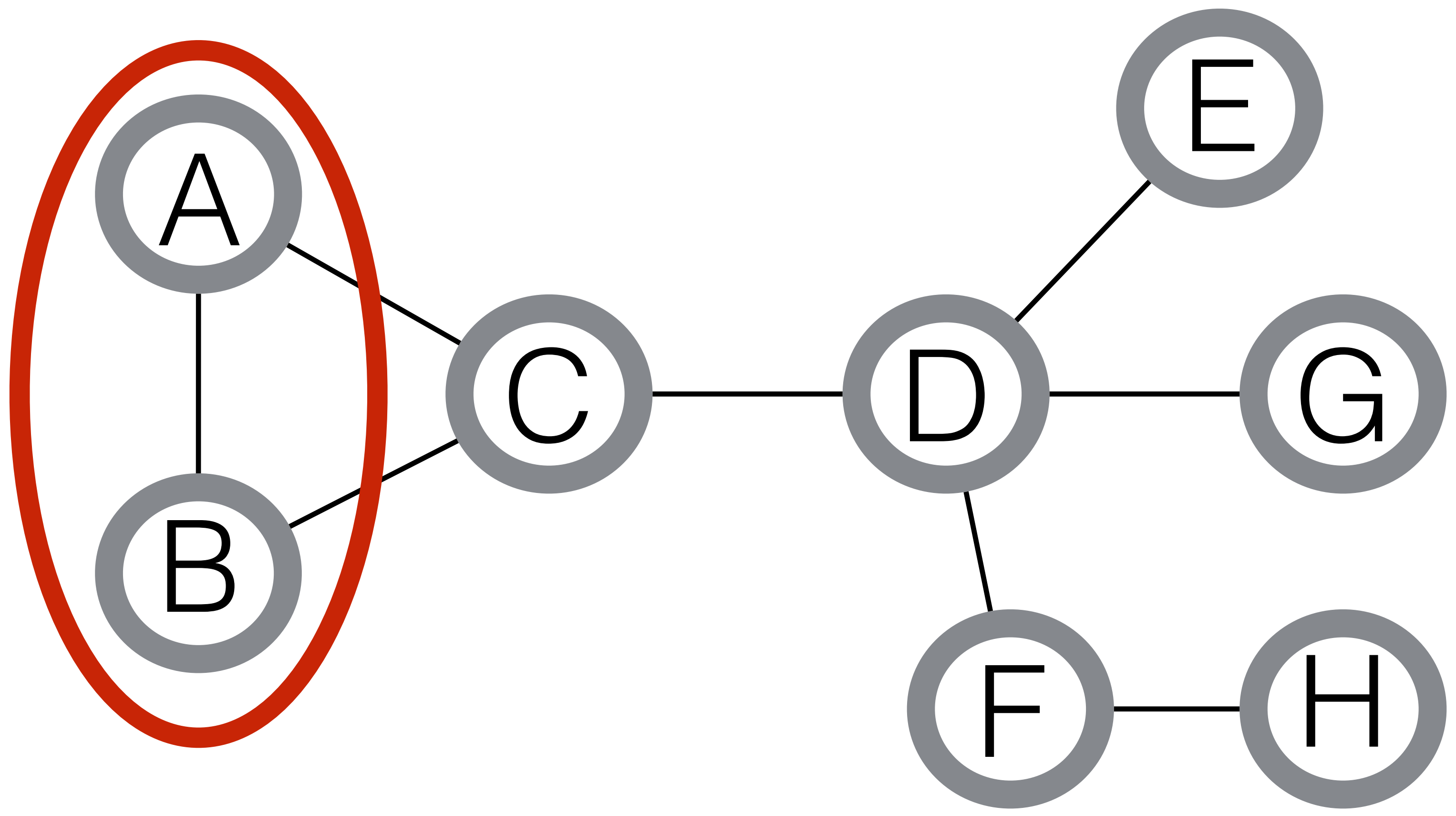
Conductance({A,B,C}) = 1/(2 + 2 + 3) = 1/7

- The smaller the conductance value the better
- Minimizing conductance is NP-hard, we use approximation algorithms

Cluster quality

We measure cluster quality using

$$\text{Conductance} := \frac{\text{number of edges leaving cluster}}{\text{sum of degrees of vertices in cluster}}$$



Conductance(**{A,B}**) = 2/(2 + 2) = 1/2

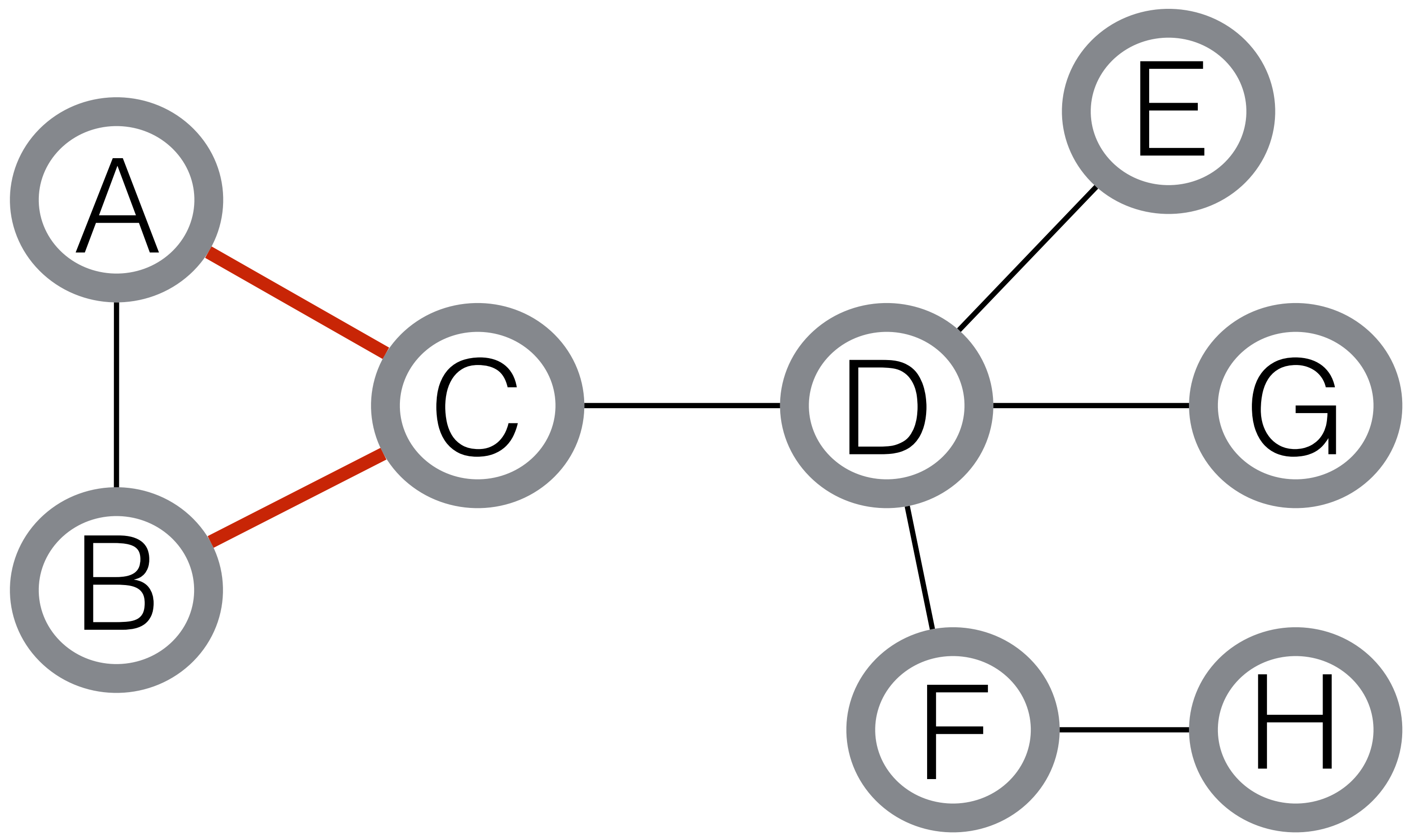
Conductance({A,B,C}) = 1/(2 + 2 + 3) = 1/7

- The smaller the conductance value the better
- Minimizing conductance is NP-hard, we use approximation algorithms

Cluster quality

We measure cluster quality using

$$\text{Conductance} := \frac{\text{number of edges leaving cluster}}{\text{sum of degrees of vertices in cluster}}$$



Conductance({A,B}) = **2**/(2 + 2) = 1/2

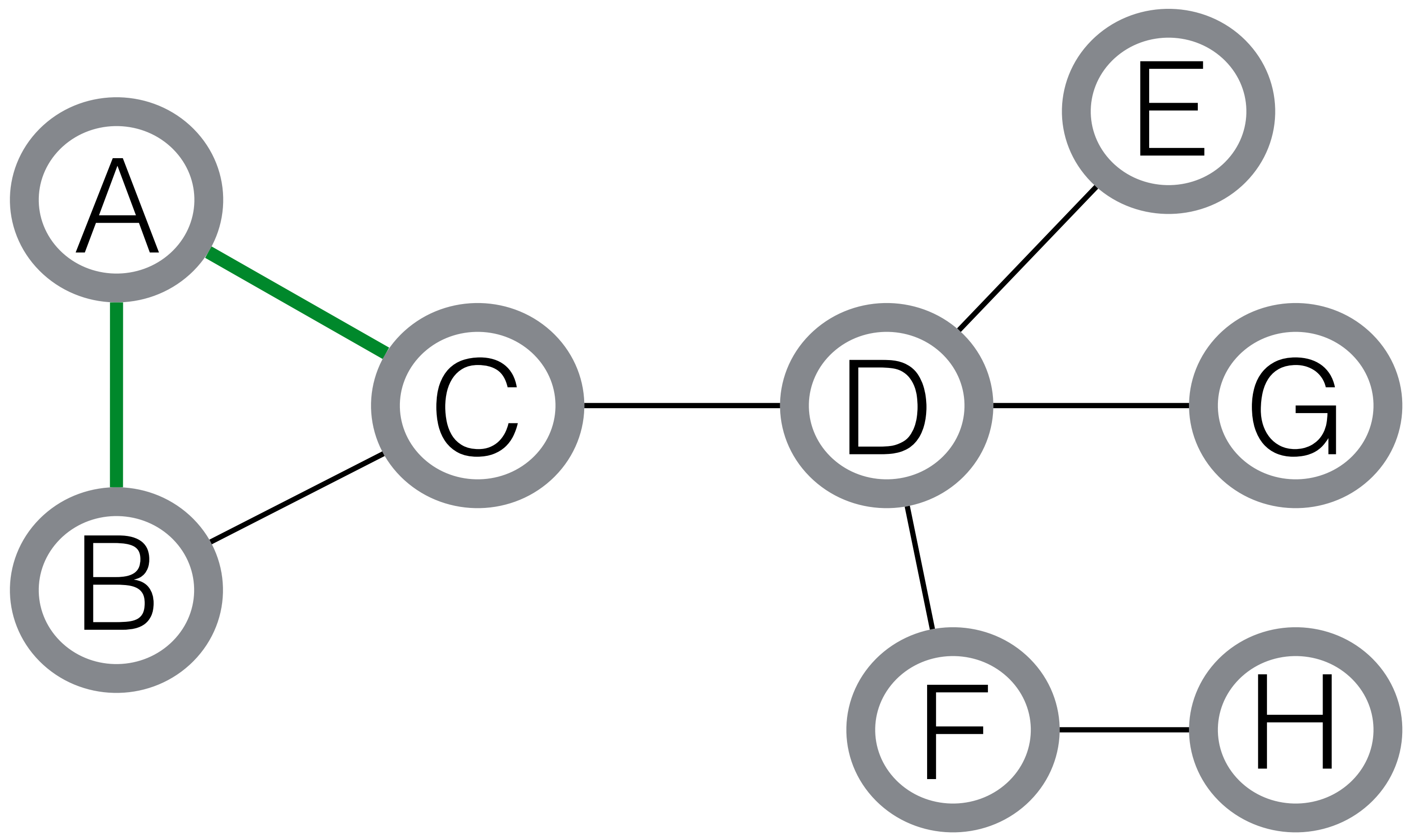
Conductance({A,B,C}) = 1/(2 + 2 + 3) = 1/7

- The smaller the conductance value the better
- Minimizing conductance is NP-hard, we use approximation algorithms

Cluster quality

We measure cluster quality using

$$\text{Conductance} := \frac{\text{number of edges leaving cluster}}{\text{sum of degrees of vertices in cluster}}$$



Conductance({A,B}) = 2/(**2** + 2) = 1/2

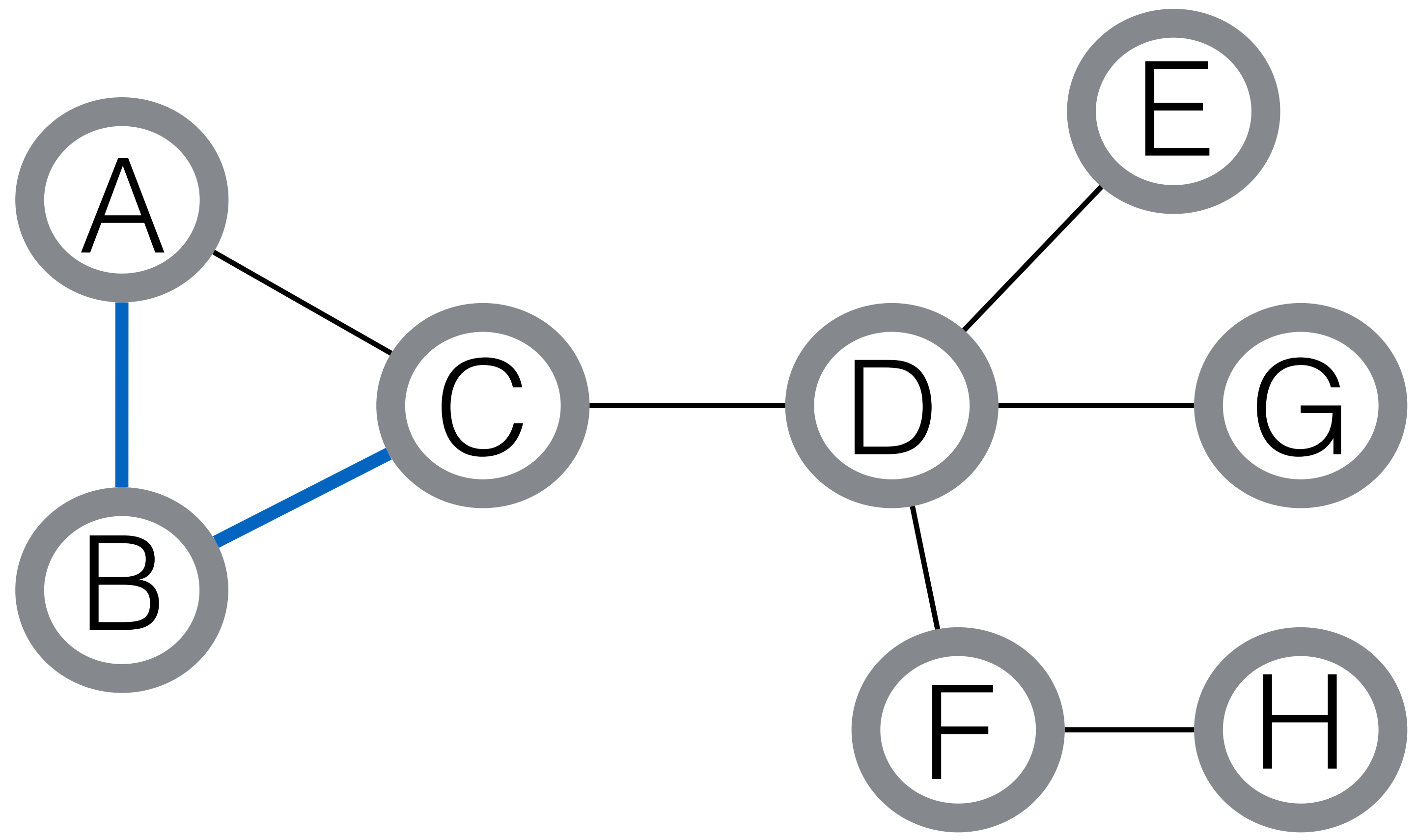
Conductance({A,B,C}) = 1/(2 + 2 + 3) = 1/7

- The smaller the conductance value the better
- Minimizing conductance is NP-hard, we use approximation algorithms

Cluster quality

We measure cluster quality using

$$\text{Conductance} := \frac{\text{number of edges leaving cluster}}{\text{sum of degrees of vertices in cluster}}$$



Conductance({A,B}) = 2/(2 + **2**) = 1/2

Conductance({A,B,C}) = 1/(2 + 2 + 3) = 1/7

- The smaller the conductance value the better
- Minimizing conductance is NP-hard, we use approximation algorithms

Local graph clustering methods

- **MQI (strongly local):** Lang and Rao, 2004
- **Approximate Page Rank (strongly local):** Andersen, Chung, Lang, 2006
- **spectral MQI (strongly local):** Chung, 2007
- **Flow-Improve (weakly local):** Andersen and Lang, 2008
- **MOV (weakly local):** Mahoney, Orecchia, Vishnoi, 2012
- **Nibble (strongly local):** Spielman and Teng, 2013
- **Local Flow-Improve (strongly local):** Orecchia, Zhu, 2014
- **Deterministic HeatKernel PR (strongly local):** Kloster, Gleich, 2014
- **Randomized HeatKernel PR (strongly local):** Chung, Simpson, 2015

- **Sweep cut rounding algorithm**

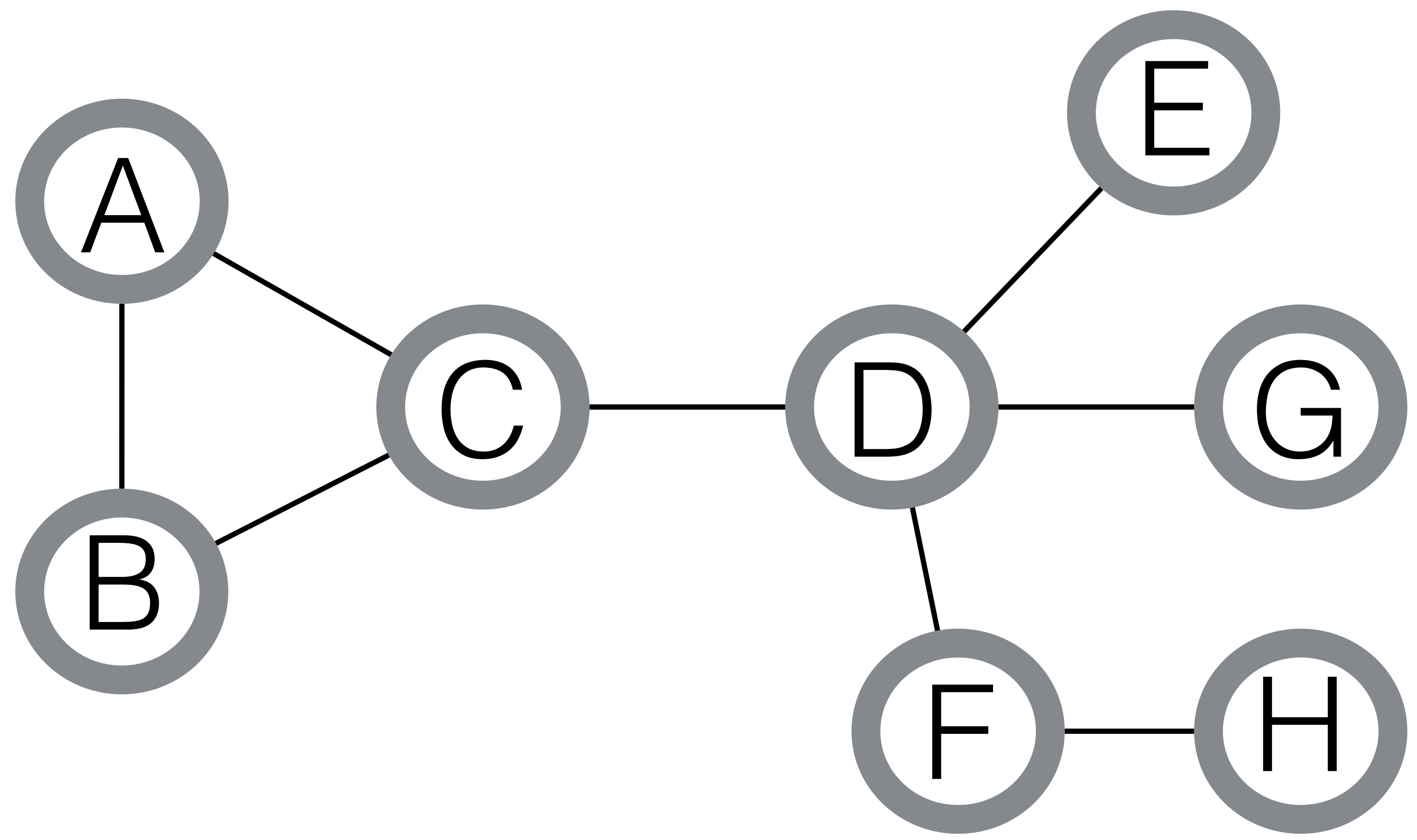
Shared memory parallel methods

- **We parallelize 4 strongly local spectral methods + rounding**
  1. Approximate Page Rank ← - - - this talk
  2. Nibble
  3. Deterministic HeatKernel Approximate Page-Rank
  4. Randomized HeatKernel Approximate Page-Rank
  5. Sweep cut rounding algorithm ← - - - this talk

- **All local methods take various parameters**
  - Parallel method 1: try different parameters independently in parallel
  - Parallel method 2: **parallelize algorithm for individual run**
    - Useful for **interactive** setting where tweaking of parameters is needed

Approximate Page-Rank

Personalized Page-Rank vector



Adjacency matrix A

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

Degree matrix D

	A	B	C	D	E	F	G	H
A	2							
B		2						
C			3					
D				4				
E					1			
F						2		
G							1	
H								1

Pick a vertex u of interest and define a vector:

$$s[u] = 1, \quad s[v] = 0 \quad \forall v \neq u$$

a teleportation parameter $0 \leq \alpha \leq 1$ and $W = AD^{-1}$ then the PPR vector is given by solving:

$$((1-\alpha)W + \alpha s e^T)p = p \quad \Leftrightarrow \quad (I - (1-\alpha)W)p = \alpha s$$

Approximate Personalized Page-Rank

R. Andersen, F. Chung and K. Lang. Local graph partitioning using Page-Rank, FOCS, 2006

Algorithm idea: iteratively spread probability mass from vector s around the graph.

- r is the residual vector, p is the solution vector
- $\rho>0$ is tolerance parameter

Run a coordinate descent solver for PPR until: any vertex u satisfies $r[u] \geq -\alpha\rho d[u]$

Initialize: $p = 0$, $r = -\alpha s$

While termination criterion is not met do

1. Choose any vertex u where $r[u] < -\alpha\rho d[u]$

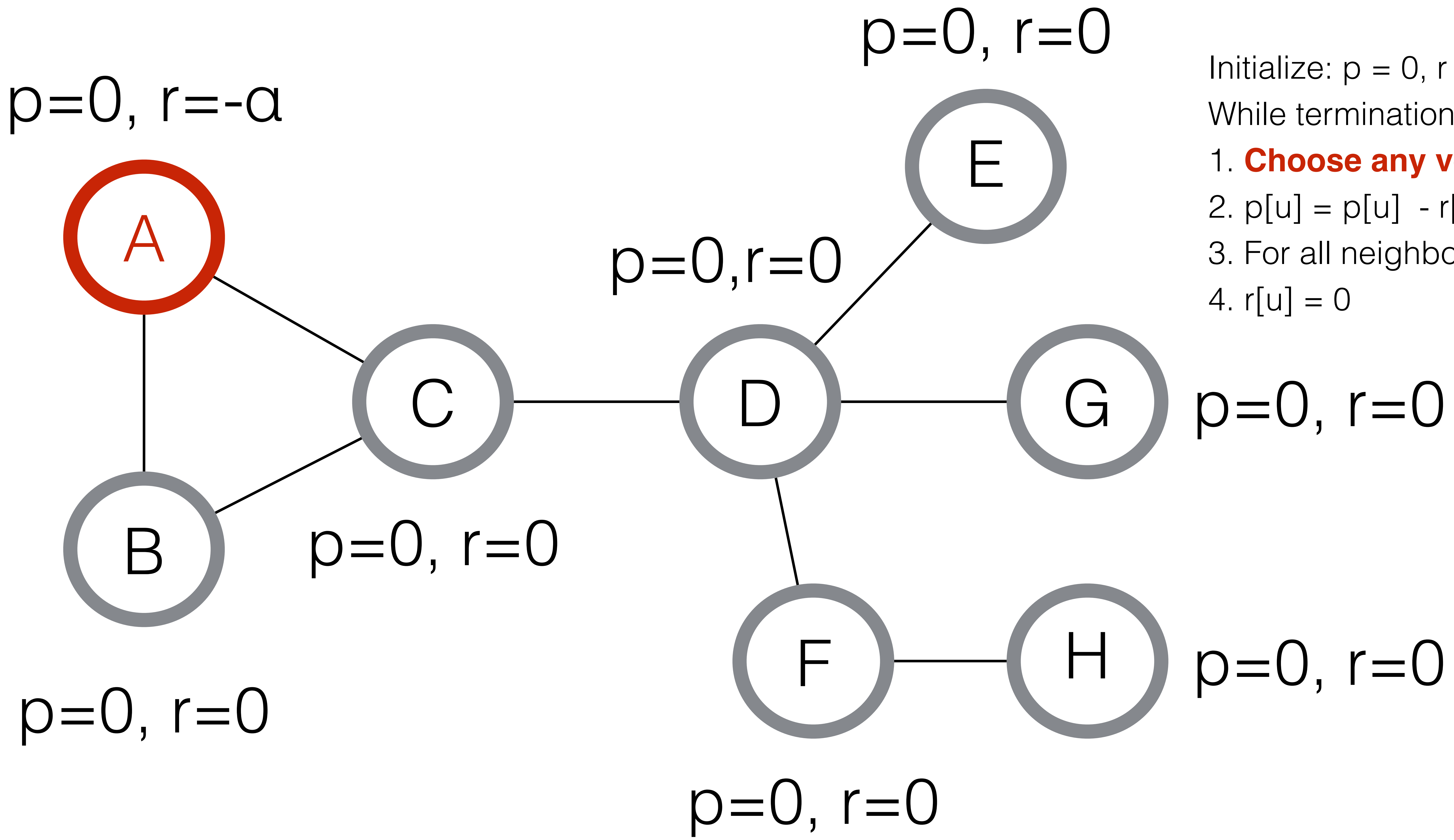
2. $p[u] = p[u] - r[u]$

residual update $\begin{cases} \textbf{3. For all neighbours v of u: } r[v] = r[v] + (1-\alpha)r[u]A[u,v]/d[u] \\ \textbf{4. } r[u] = 0 \end{cases}$

Final step: round the solution p using sweep cut.

Approximate Personalized Page-Rank

R. Andersen, F. Chung and K. Lang. Local graph partitioning using Page-Rank, FOCS, 2006



Initialize: p = 0, r = -αs

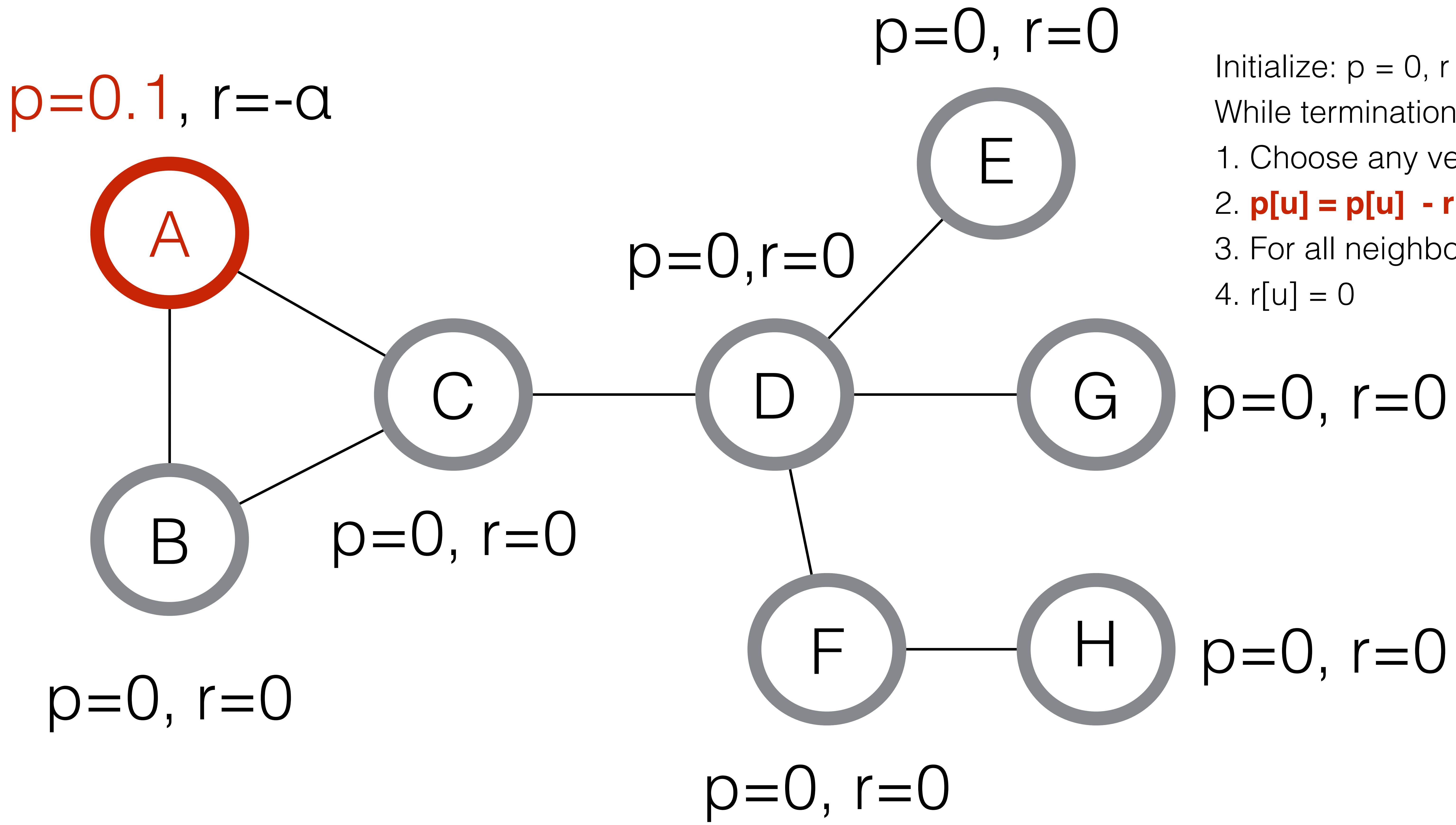
While termination criterion is not met do

1. **Choose any vertex u where r[u] < -αpd[u]**
2. p[u] = p[u] - r[u]
3. For all neighbours v of u: r[v] = r[v] + (1-α)r[u]A[u,v]/d[u]
4. r[u] = 0

$$\frac{\|r\|_1}{\alpha} = 1, \quad \|p\|_1 = 0, \quad \frac{\|r\|_1}{\alpha} + \|p\|_1 = 1$$

Approximate Personalized Page-Rank

R. Andersen, F. Chung and K. Lang. Local graph partitioning using Page-Rank, FOCS, 2006



Initialize: p = 0, r = -αs

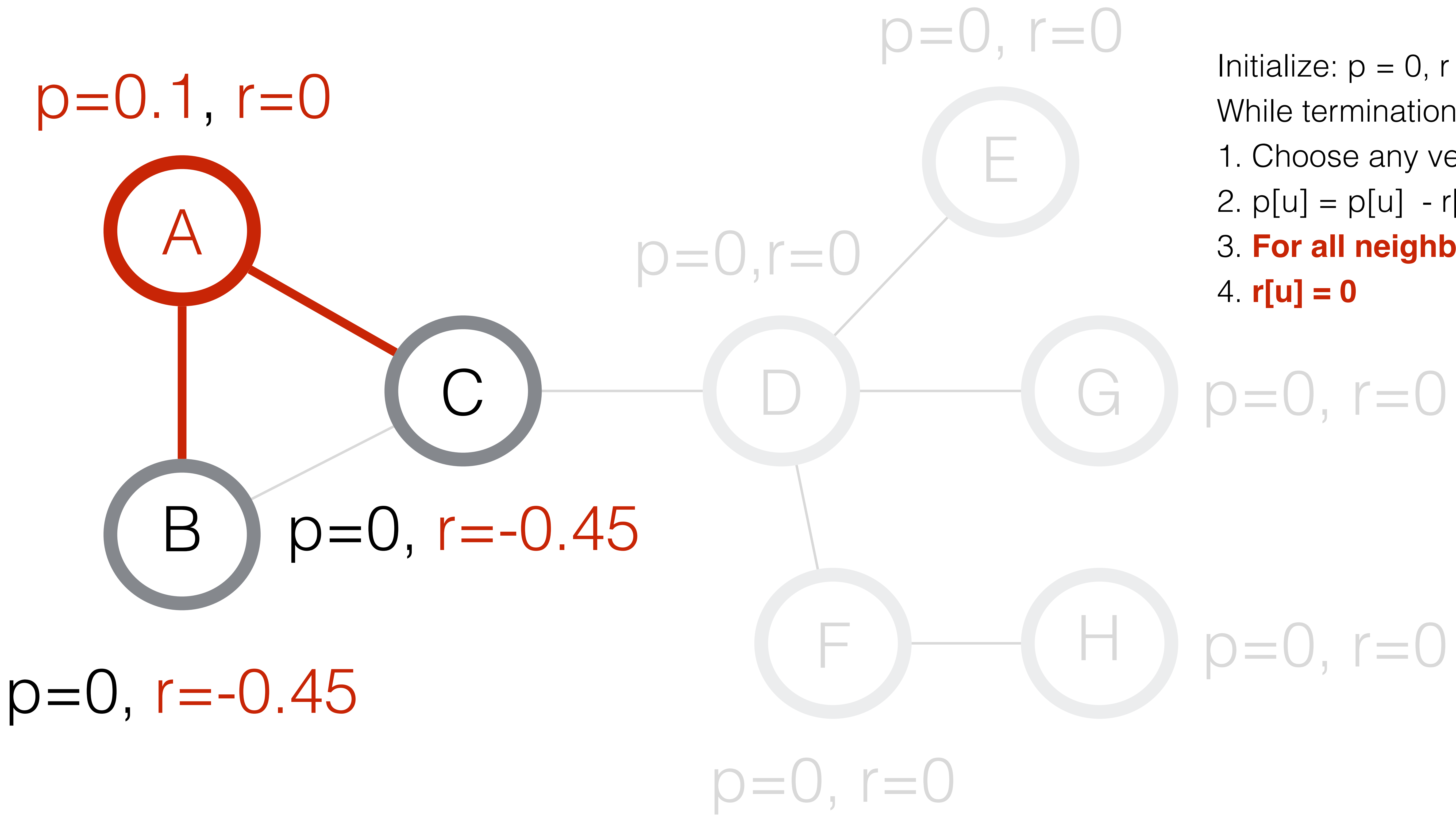
While termination criterion is not met do

1. Choose any vertex u where r[u] < -αpd[u]
2. **p[u] = p[u] - r[u]**
3. For all neighbours v of u: r[v] = r[v] + (1-α)r[u]A[u,v]/d[u]
4. r[u] = 0

$$\frac{\|r\|_1}{\alpha} = 1, \quad \|p\|_1 = 0.1, \quad \frac{\|r\|_1}{\alpha} + \|p\|_1 = 1.1$$

Approximate Personalized Page-Rank

R. Andersen, F. Chung and K. Lang. Local graph partitioning using Page-Rank, FOCS, 2006



Initialize: p = 0, r = -αs, where s is a probability vector

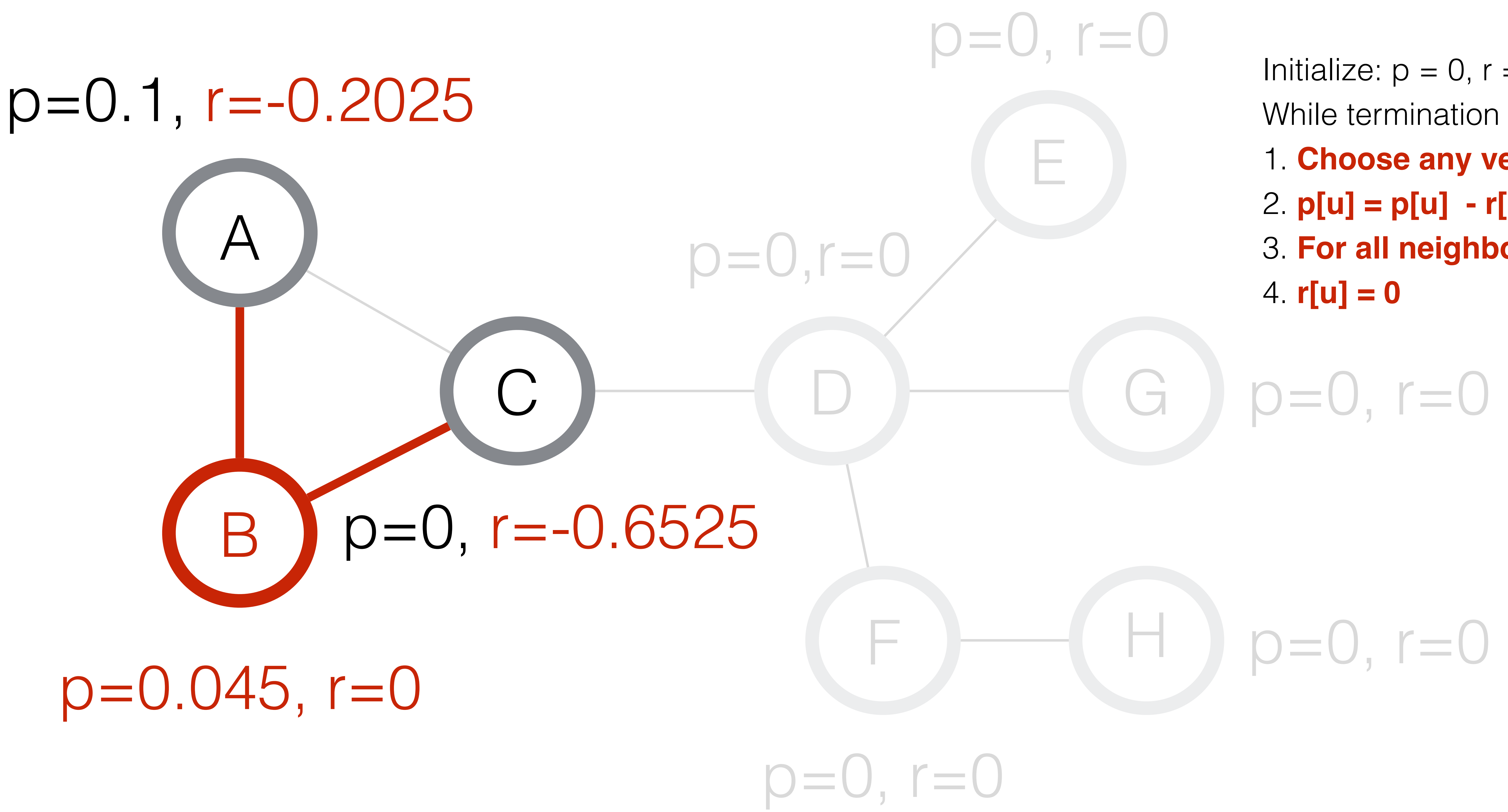
While termination criterion is not met do

1. Choose any vertex u where r[u] < -αpd[u]
2. p[u] = p[u] - r[u]
3. **For all neighbours v of u: r[v] = r[v] + (1-α)r[u]A[u,v]/d[u]**
4. **r[u] = 0**

$$\frac{\|r\|_1}{\alpha} = 0.9, \quad \|p\|_1 = 0.1, \quad \frac{\|r\|_1}{\alpha} + \|p\|_1 = 1.0$$

Approximate Personalized Page-Rank

R. Andersen, F. Chung and K. Lang. Local graph partitioning using Page-Rank, FOCS, 2006



Initialize: p = 0, r = -αs

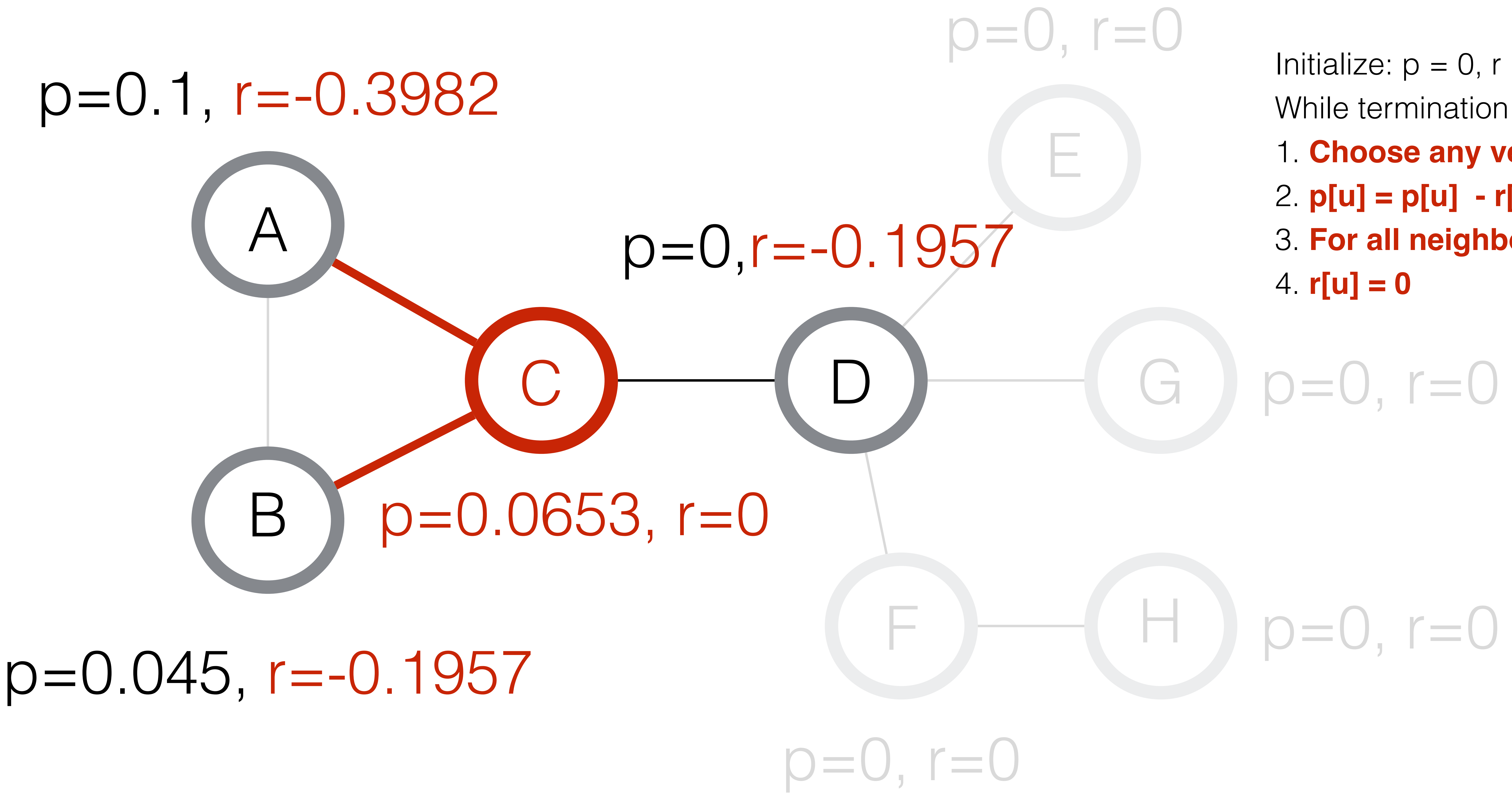
While termination criterion is not met do

1. **Choose any vertex u where r[u] < -αρd[u]**
2. **p[u] = p[u] - r[u]**
3. **For all neighbours v of u: r[v] = r[v] + (1-α)r[u]A[u,v]/d[u]**
4. **r[u] = 0**

$$\frac{\|r\|_1}{\alpha} = 0.855, \quad \|p\|_1 = 0.145, \quad \frac{\|r\|_1}{\alpha} + \|p\|_1 = 1.0$$

Approximate Personalized Page-Rank

R. Andersen, F. Chung and K. Lang. Local graph partitioning using Page-Rank, FOCS, 2006



Initialize: p = 0, r = -αs, where s is a probability vector

While termination criterion is not met do

1. **Choose any vertex u where r[u] < -αpd[u]**
2. **p[u] = p[u] - r[u]**
3. **For all neighbours v of u: r[v] = r[v] + (1-α)r[u]A[u,v]/d[u]**
4. **r[u] = 0**

$$\frac{\|r\|_1}{\alpha} = 0.7897, \quad \|p\|_1 = 0.2103, \quad \frac{\|r\|_1}{\alpha} + \|p\|_1 = 1.0$$

Running time APPR

- At each iteration APPR touches a single node and its neighbours
  - Let supp(p) be the support of vector p at termination which satisfies **vol(supp(p)) ≤ 1/(αρ)**
  - **Overall until termination the work is: *O*(1/(αρ))** [Andersen, Chung, Lang, FOCS, 2006]

- We store vectors p and r using sparse sets
  - **We can only afford to do work proportional to nodes and edges currently touched**
  - We used *unordered_map* data structure in STL (Standard Template Library)
  - Guarantees *O*(1/(αρ)) work

L1-regularized Page-Rank

APPR is an approximation algorithm but what is it minimizing?

$$\text{minimize } \frac{1-\alpha}{2}\|Bp\|_2^2 + \alpha\|H(\mathbf{1}-p)\|_2^2 + \alpha\|Zp\|_2^2 + \rho\alpha\|Dp\|_1$$

where

- B: is the incidence matrix
- Z, H: are diagonal scaling matrices

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

KF, X. Cheng, J. Shun, F. Roosta-Khorasani, M. Mahoney. Exploiting optimization for local graph clustering, arXiv:1602.01886v1.

KF, D. Gleich, M. Mahoney. An optimization approach to locally-biased graph algorithms, arXiv:1607.04940.

Shared memory

Running time: work depth model

Work depth model: J. Jaja. Introduction to parallel algorithms. Addison-Wesley Profesional, 1992

Note that our results are not model dependent.

Model

- Work: number of operations required
- Depth: longest chain of sequential dependencies

Let P be the number of cores available.

By Brent's theorem [1] **an algorithm with work W and depth D has overall running time: W/P + D**.

In practice W/P dominates. Thus parallel efficient algorithms require the same work as its sequential version.

Brent's theorem: [1] R. P. Brent. The parallel evaluation of general arithmetic expressions. J ACM (JACM), 21(2):201-206, 1974

Parallel Approximate Personalized Page-Rank

While termination criterion is not met do

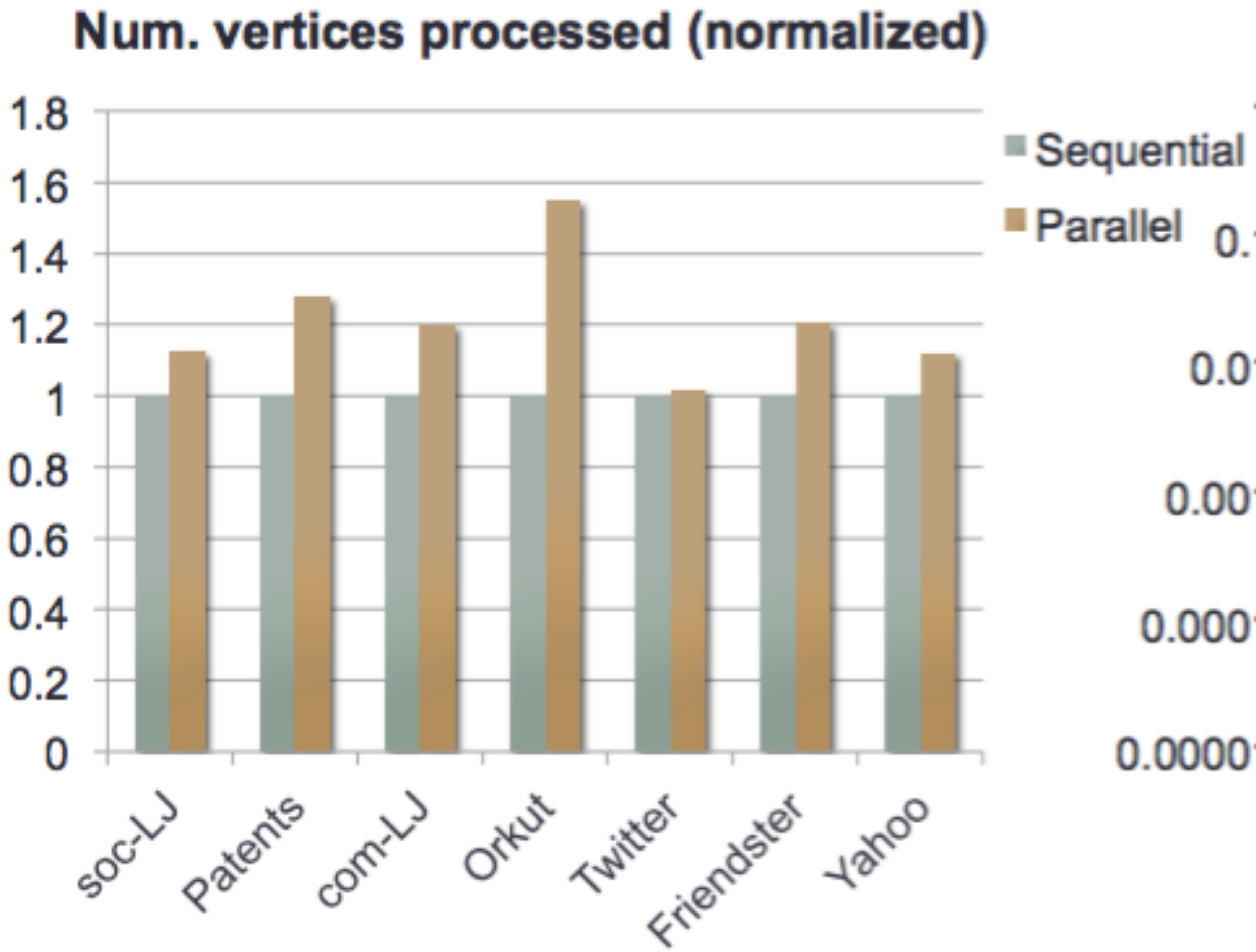
1. Choose **ALL (instead of any)** vertex u where $r[u] < -\alpha\rho deg[u]$
2. $p[u] = p[u] - r[u]$
3. For all neighbours v of u: $r[v] = r[v] + (1-\alpha)/(2deg[u])r[u]$
4. $r[u] = (1-\alpha)r[u]/2$

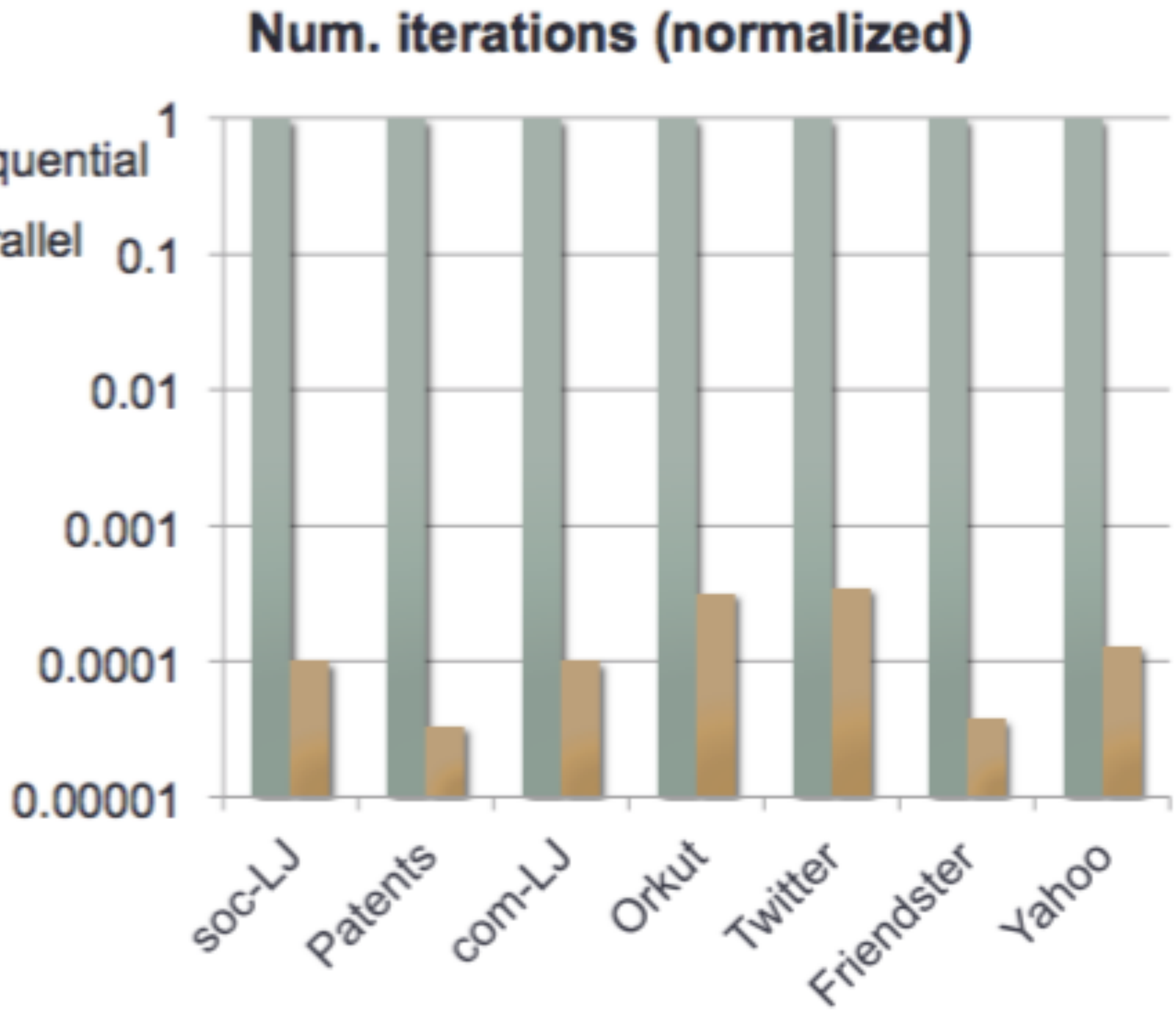
- **Asymptotic work remains the same:** $O(1/(\alpha\rho))$.
- **Parallel randomized implementation: work $O(1/(\alpha\rho))$ and depth $O(\log(1/(\alpha\rho))$.**
  - Keep track of two **sparse** copies of p and r
  - Concurrent hash table for sparse sets <— important for $O(1/(\alpha\rho))$ work
  - Use atomic increment to deal with conflicts
  - Use of Ligra (Shun and Blelloch 2013) to process only “active” vertices and their edges
- **Same theoretical graph clustering guarantees**, Fountoulakis et al. 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

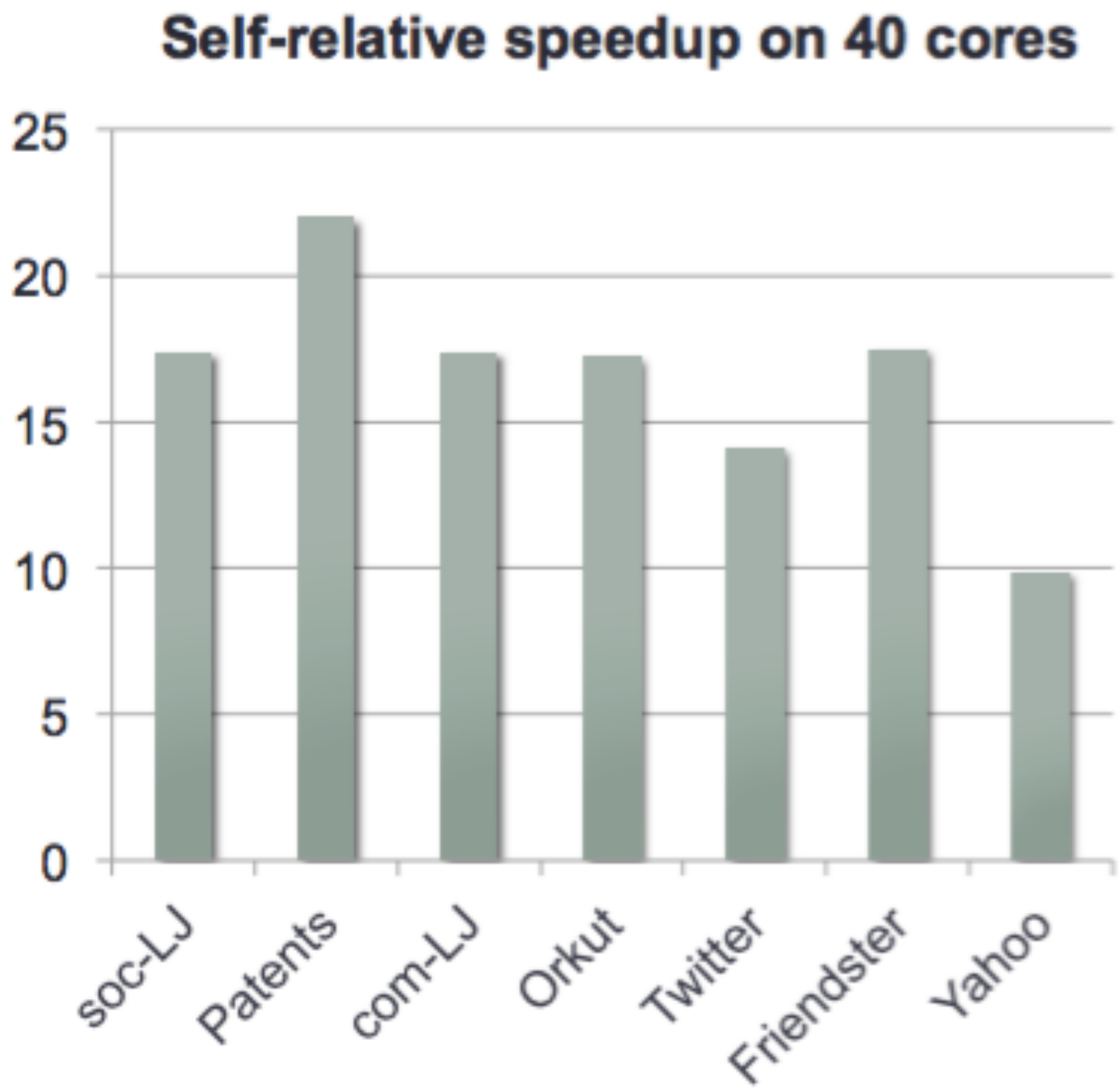
Performance

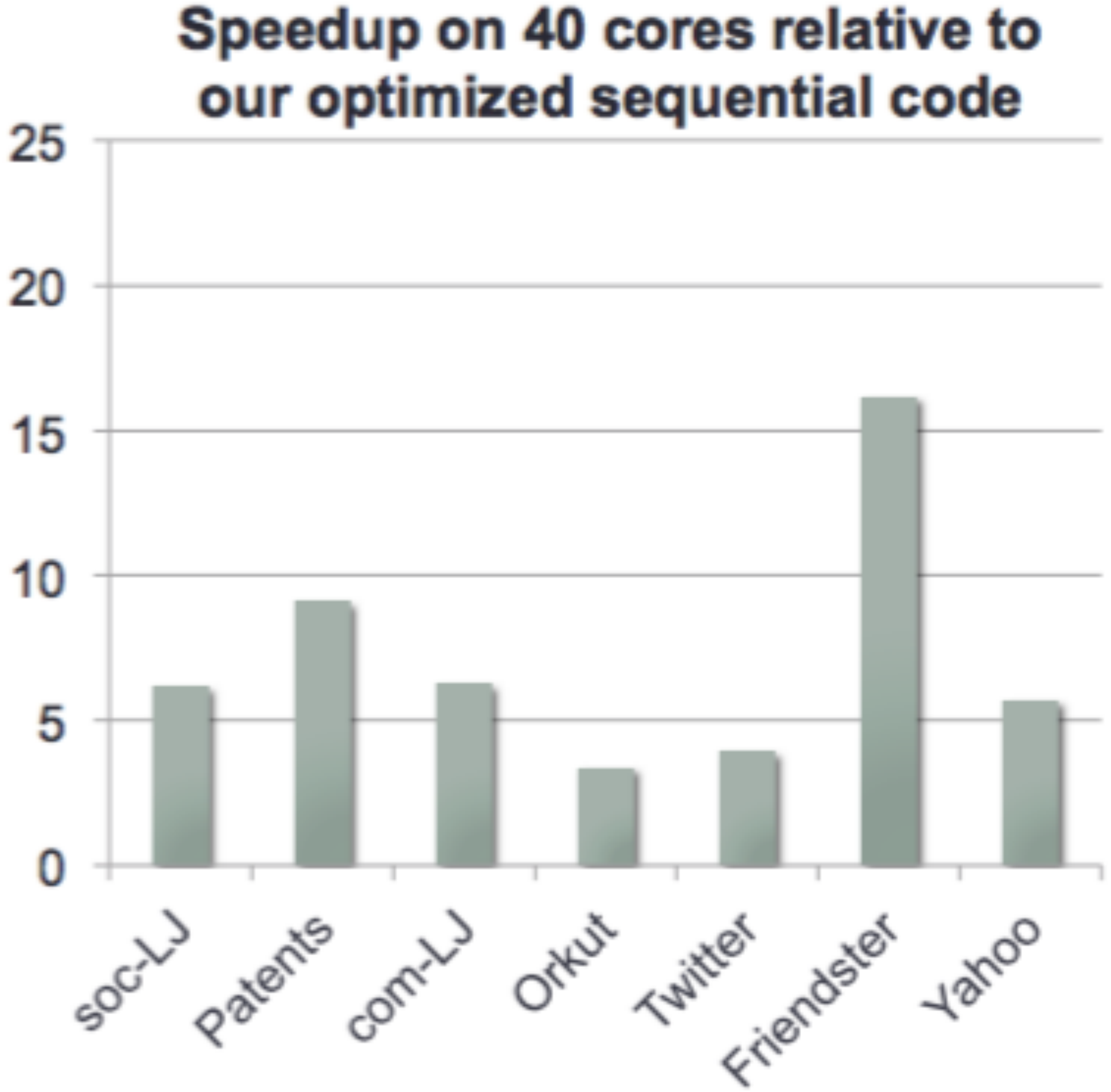




- Slightly more work for the parallel version
- Number of iterations is significantly less

Performance

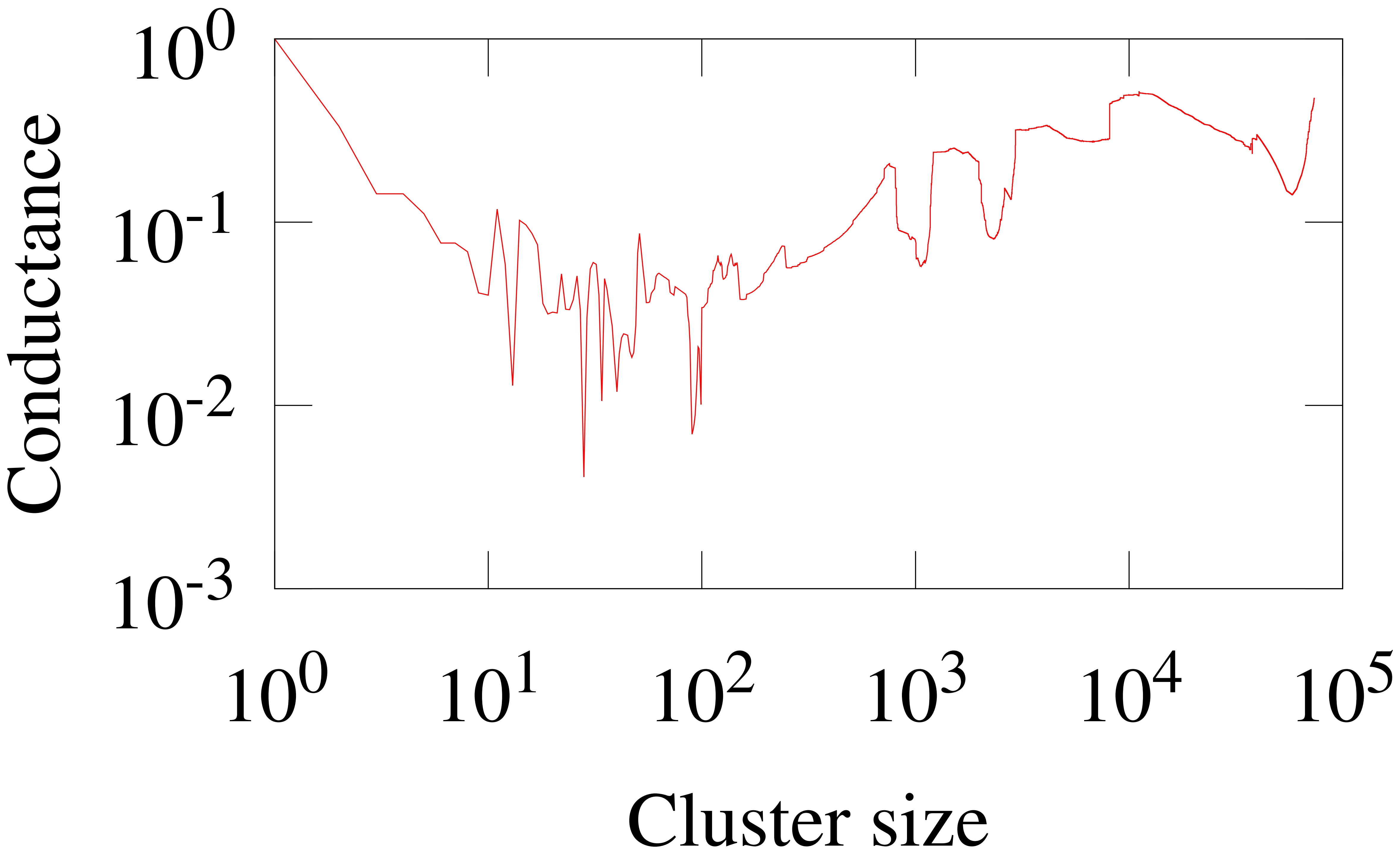




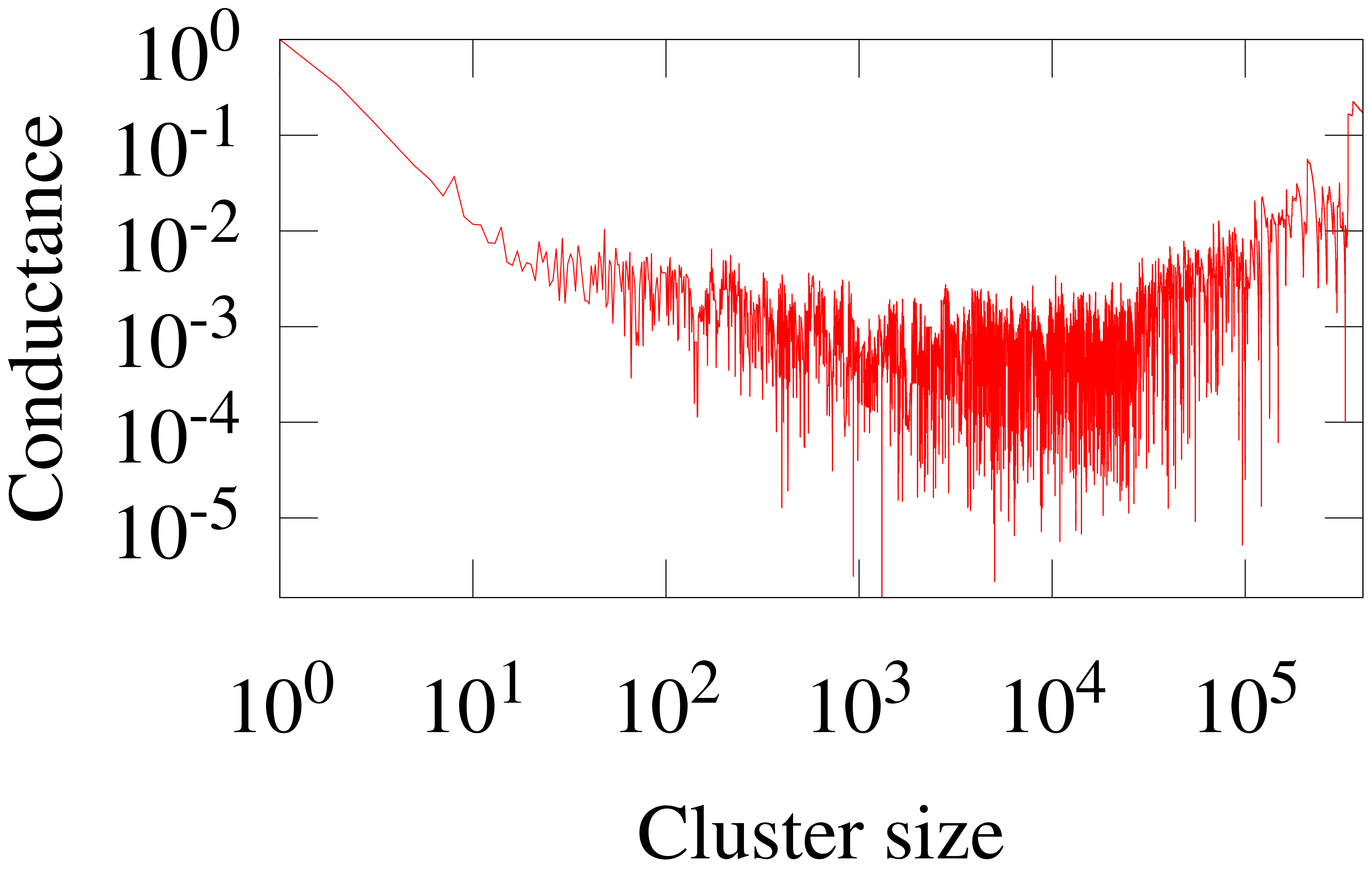
- 3-16x speed up
- Speedup is limited by small active set in some iterations and memory effects

Network community profile plots

Friendster, 124M nodes, 1.8B edges



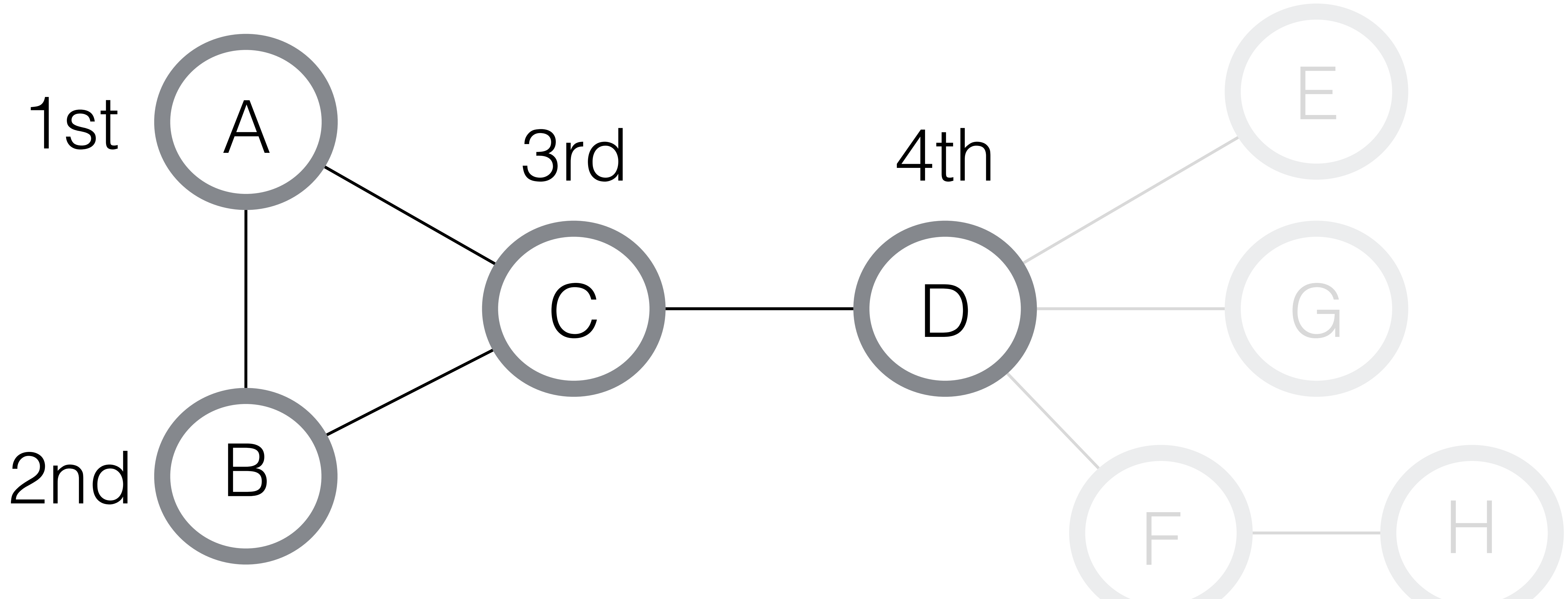
Yahoo, 1.4B nodes, 6.4B edges



- $O(10^5)$ approximate PPR problems were solved in parallel for each plot,
- Agrees with conclusions of [Leskovec et al. 2008], i.e., good clusters tend to be small.

Rounding: sweep cut

- Round returned vector p of approximate PPR
  - **1st step (O(1/(αρ) log(1/(αρ))) work):** Sort vertices by non-increasing value of non-zero p[u]/d[u]
  - **2nd step (O(1/(αρ)) work):** Look at all prefixes of sorted order and return the cluster with minimum conductance,



Sorted vertices: {A,B,C,D}

Cluster	Conductance
{A}	1
{A,B}	1/2
{A,B,C}	**1/7**
{A,B,C,D}	3/11

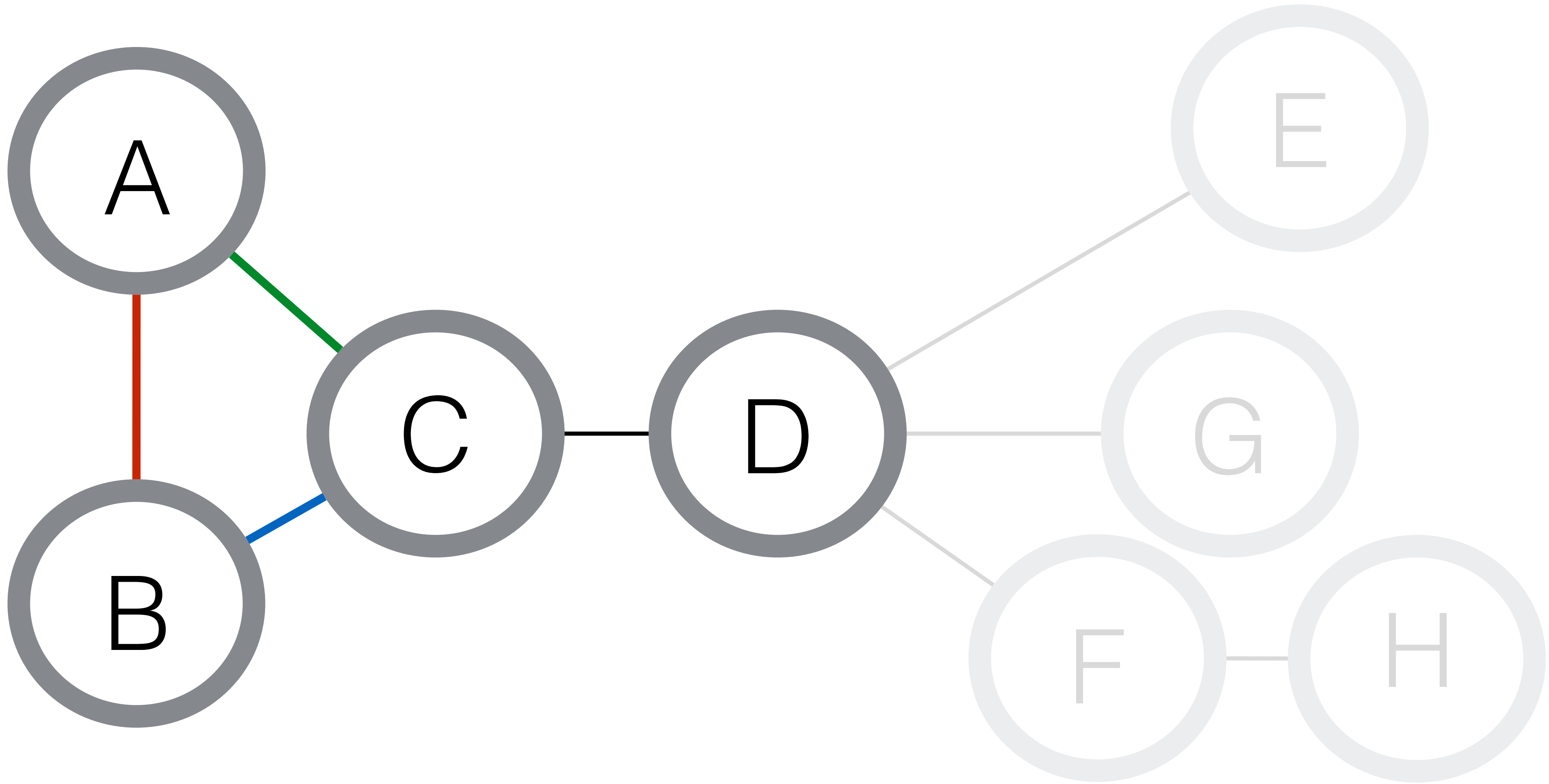
Parallel sweep cut

- **1st step:** Sort vertices by non-increasing value of non-zero p[u]/d[u].
  - Use parallel sorting algorithm, $O(1/(\alpha\rho) \log(1/(\alpha\rho)))$ work and $\boldsymbol{O}(\log(1/(\alpha\rho))$ depth.
- **2nd step:** Look at all prefixes of sorted order and return the cluster with minimum conductance.
  - Naive implementation: for each sorted prefix compute conductance, $O((1/(\alpha\rho))^2)$.
  - **We design a parallel algorithm based on integer sorting and prefix sums that takes $O(1/(\alpha\rho))$ time.**
  - **The algorithm computes the conductance of ALL sets with a single pass over the nodes and the edges.**

Parallel sweep cut: 2nd step

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

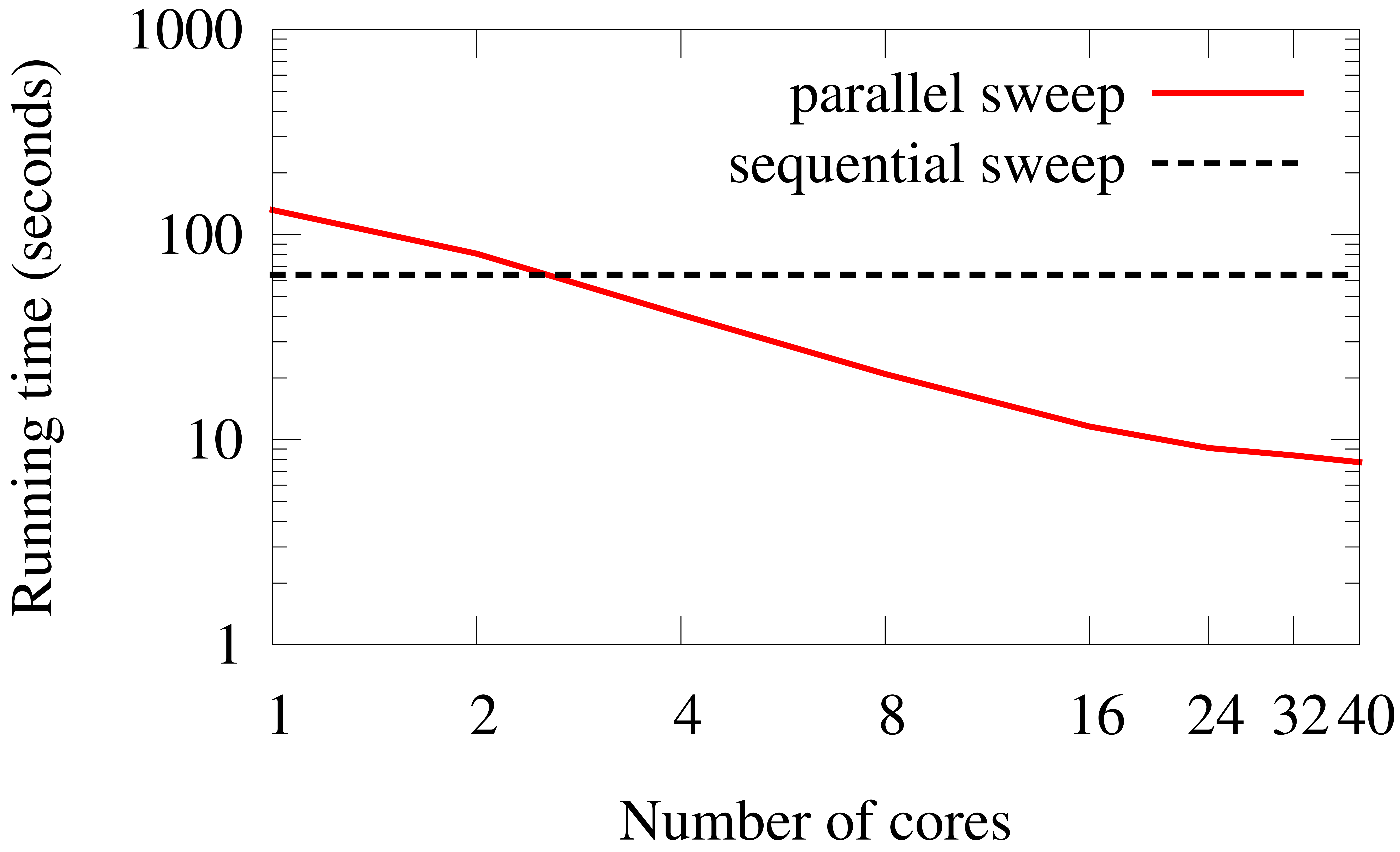


Sorted vertices: {A,B,C,D}

Cluster	Sum cols B	Volume	Conductance
{A}	**2**	**2**	2/2=1
{A,B}	2	4	2/4=1/2
{A,B,C}	1	7	1/7
{A,B,C,D}	3	11	3/11

- Sort vertices
  - work: $O(1/(\alpha\rho) \log(1/(\alpha\rho)))$, depth: $O(\log(1/(\alpha\rho)))$
- Represent matrix B with a sparse set using vertex identifiers and the order of vertices
  - work: $O(1/(\alpha\rho))$, depth: $O(\log(1/(\alpha\rho)))$
- Use prefix sums to sum elements of the columns
  - work: $O(1/(\alpha\rho))$, depth: $O(\log(1/(\alpha\rho)))$

Parallel sweep cut: performance



Summary

- We parallelise 4 spectral algorithms for local graph clustering.
- The proposed algorithms are work efficient, i.e., same worst-case work.
- We parallelise the rounding procedure to obtain the clusters.
- Useful in interactive setting where one has to experiment with parameters.
- 3-15x faster than sequential version
- Parallelisation allowed us to solve problems of billions of nodes and edges.

Further work: distributed block coordinate descent

- Generalization to l2-regularized least-squares and kernel learning:

  A. Devarakonda, KF, J. Demmel, M. Mahoney: Avoiding communication in primal and dual block coordinate descent methods (work in progress $\leq$ month)

- Given a positive integer k
  - We reduce latency for BCD by a factor of k
  - at the expense of a factor of k more work and number of words.

Thank you!