Sublinear algorithms in large networks via core-periphery decomposition

Omri Ben-Eliezer

MIT → SIMONS INSTITUTE for the Theory of Computing → TECHNION Israel Institute of Technology

Joint works with:

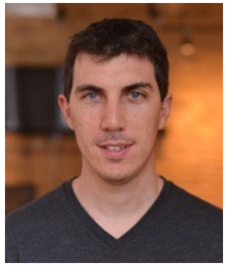
Sabyasachi Basu

Talya Eden

Dimitris Fotakis

Nadia Kōshima

Joel Oren



Tim Rieder

C. Seshadhri

Social & information networks: **properties**

- (approx) power-law degree distribution

$$\# \text{ vtxs of deg. } d \;\sim\; \boldsymbol{d^{-\gamma}}$$

Interesting regime:

$$2 < \gamma < 3$$

https://archive.nytimes.com/krugman.blogs.nytimes.com/2012/02/08/the-power-law-of-twitter/

Paul Krugman | The Power (Law) of Twitter

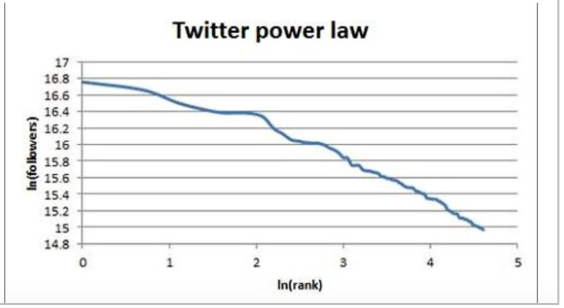
The Power (Law) of Twitter

FEBRUARY 8, 2012 11:24 AM 150

I don't tweet, but I have a robot that does — all it does it tell followers that there's a new entry on this blog. For some marketing purposes I was told to check the number of followers, of which more in a moment; and this had me wasting some time on the math of Twitter followership.

What I knew is that many more or less hierarchical systems — the size of cities, the distribution of income in the upper tail — follow a power law, meaning that number 100 is to number 10 as number 10 is to number 1. Does Twitter?

Not exactly; the rank-followership gradient is relatively flat at the top (Lady Gaga and Justin Bieber have roughly the same number of followers), then steepens once you get past the mega-celebrities:



Paul Krugman @ NY Times, 2012

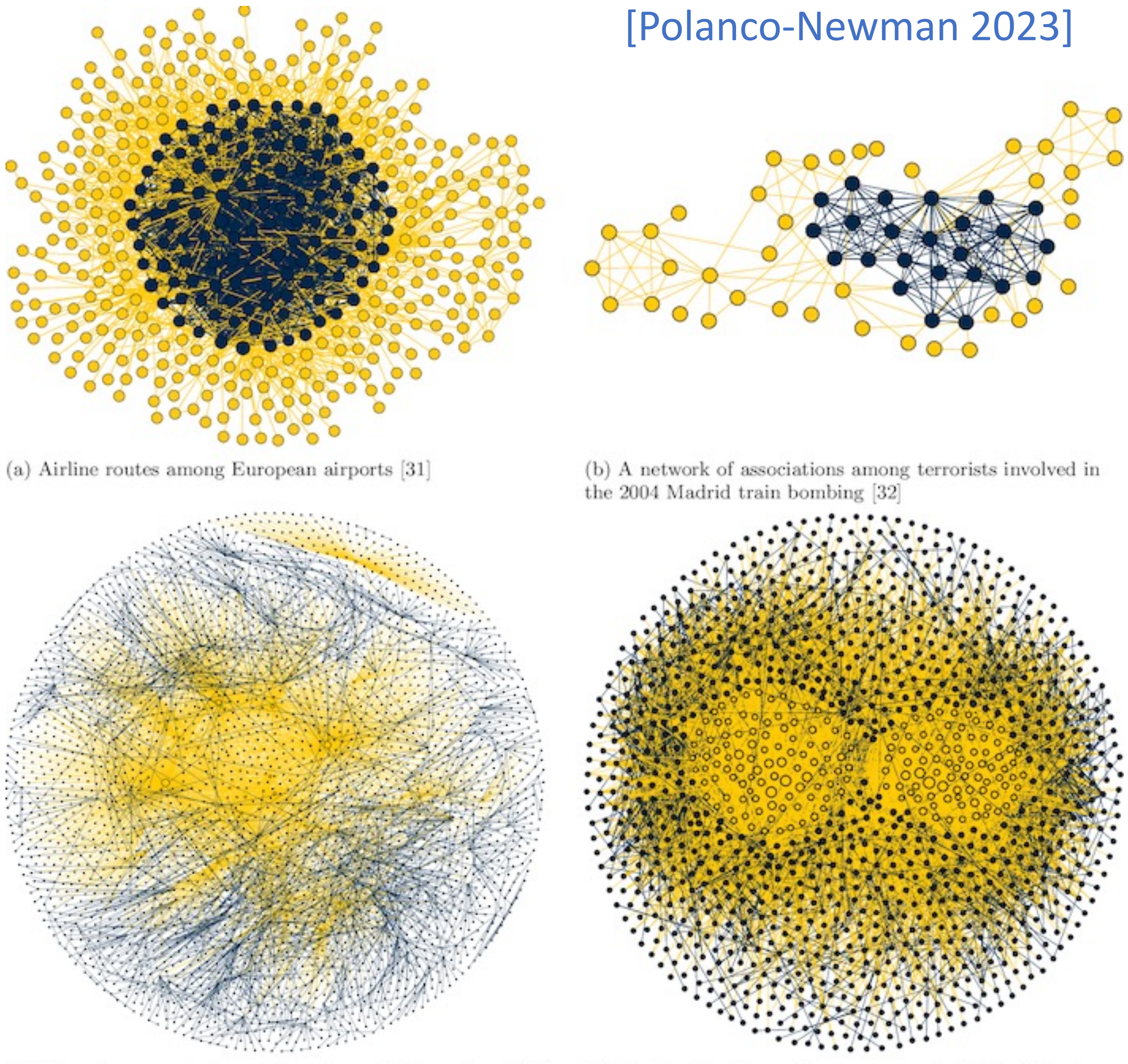
Username	Tweets	Followers	Following
elonmusk	45,834	187,591,264	633
BarackObama	17,038	131,747,695	541,856
Cristiano	4,103	111,649,069	70
justinbieber	31,148	110,657,901	273,591
rihanna	10,744	107,966,010	969
katyperry	12,020	106,335,380	243
narendramodi	42,888	98,921,919	2,673
taylorswift13	855	95,242,147	–
realDonaldTrump	59,120	87,180,802	51
ArianaGrande	47,154	85,268,269	55,494
ladygaga	9,965	83,203,487	112,207
NASA	71,976	80,963,365	175
YouTube	58,850	80,326,803	1,171
KimKardashian	36,740	75,157,368	129
EllenDeGeneres	24,641	74,163,372	25,002
X	15,331	67,769,279	–

Most followed accounts, retrieved from socialblade.com, 2024

Examples from
[Polanco-Newman 2023]

Social & information networks: **properties**

- core-periphery structure
  [BE'99, LLDM '09, RPFM'14, ZMN'15, BK'19, …]



(a) Airline routes among European airports [31]

(b) A network of associations among terrorists involved in the 2004 Madrid train bombing [32]

(c) Network representation of the Internet in November 1997 at the autonomous system level [33]

(d) A network of hyperlinks among a set of US political blogs [34]

Social & information networks: **properties**

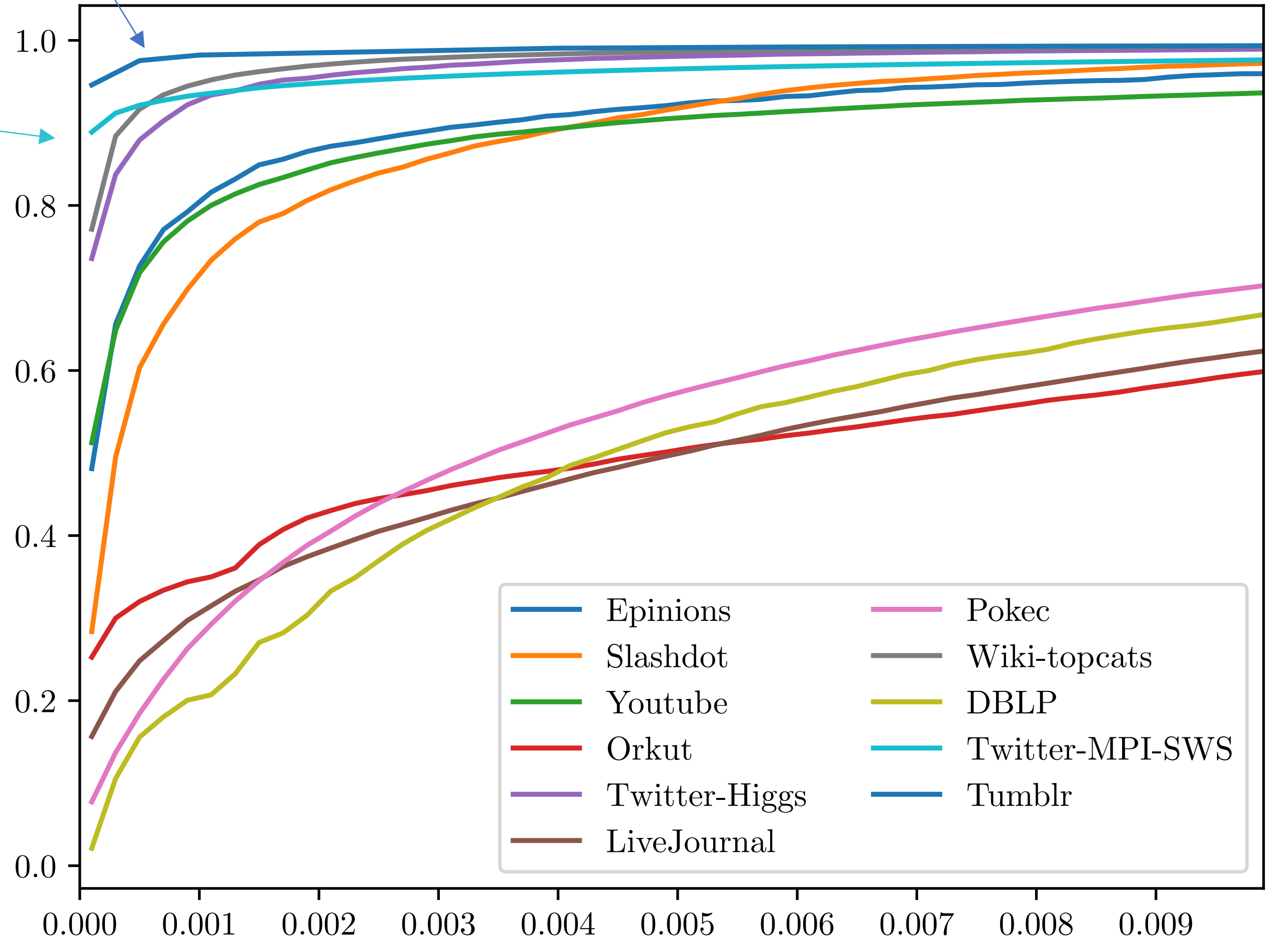
- "sublinear almost domination"
  [B., Eden, Oran, Fotakis '22]

Most nodes with out-deg ≥ 10 have neighbor in top 0.1% highest degrees.

Tumblr
247M nodes
14.5B edges

Twitter-MPI
53M node
2.0B edge

fraction "engaged" covered

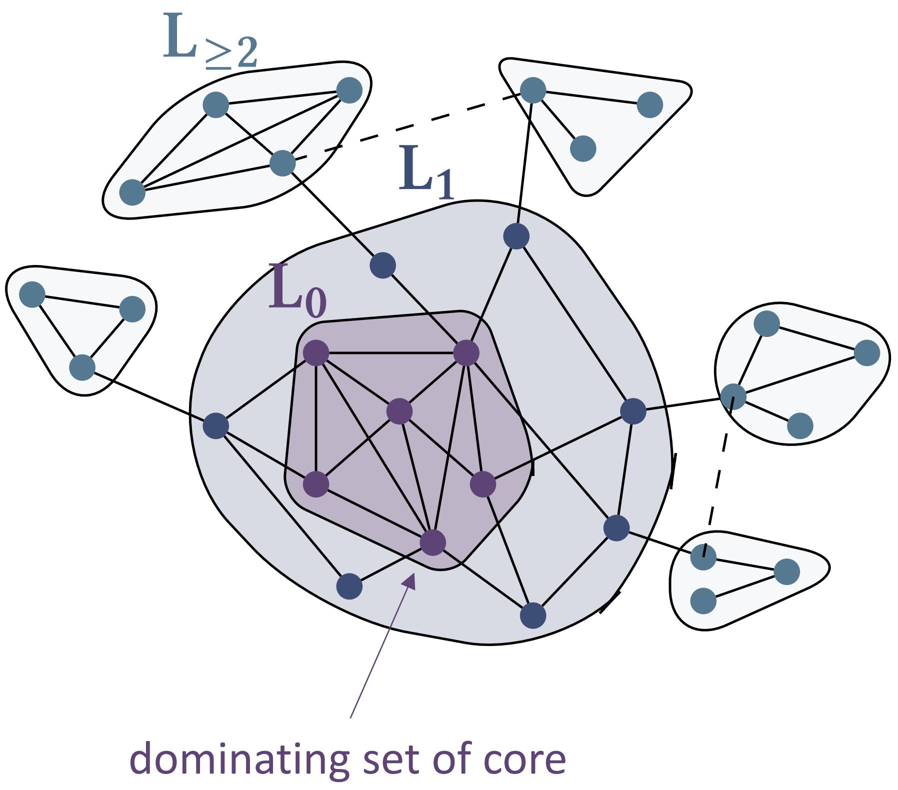


fraction of nodes in dominating set

Social & information networks: **properties**

- Power-law degree distribution
- Core-periphery structure
- "Sublinear almost domination" of core

<u>This talk</u>: core-periphery decomposition
$\Rightarrow$ sublinear algorithms



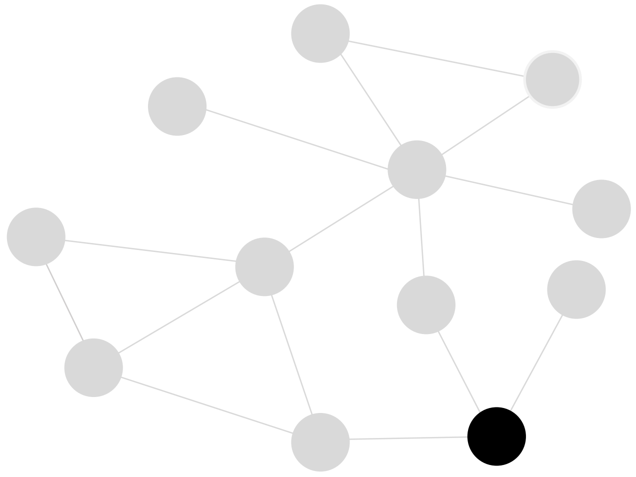
Part I: Sampling nodes

The problem: Sampling multiple nodes

Start at single random node

Explore graph through **query access**: querying node reveals its **neighbors**

Goal: generate many random nodes with as few queries as possible

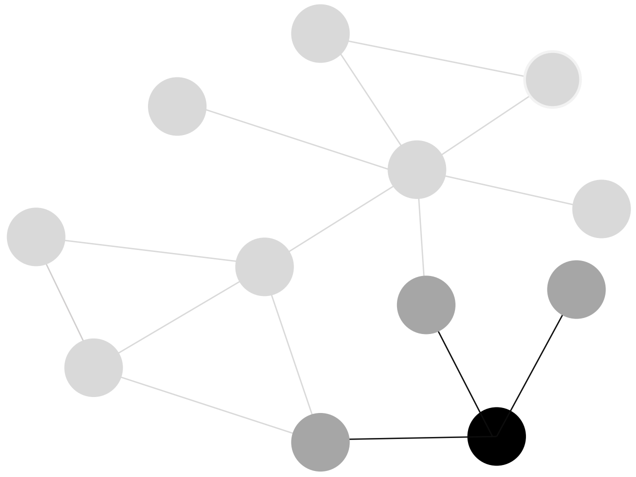


The problem: Sampling multiple nodes

Start at single random node

Explore graph through **query access**: querying node reveals its **neighbors**

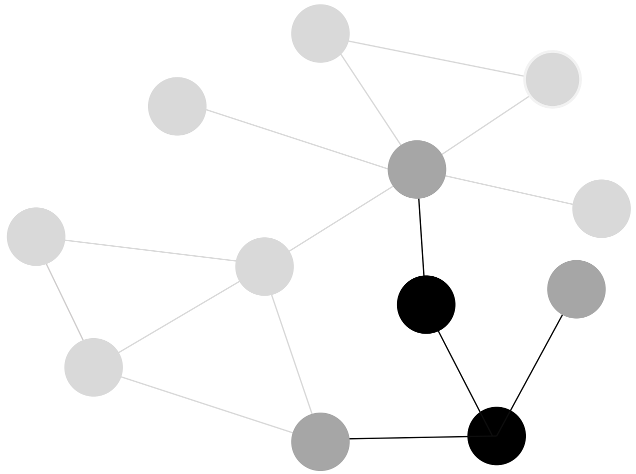
Goal: generate many random nodes with as few queries as possible



The problem: Sampling multiple nodes

Start at single random node

Explore graph through **query access**: querying node reveals its **neighbors**

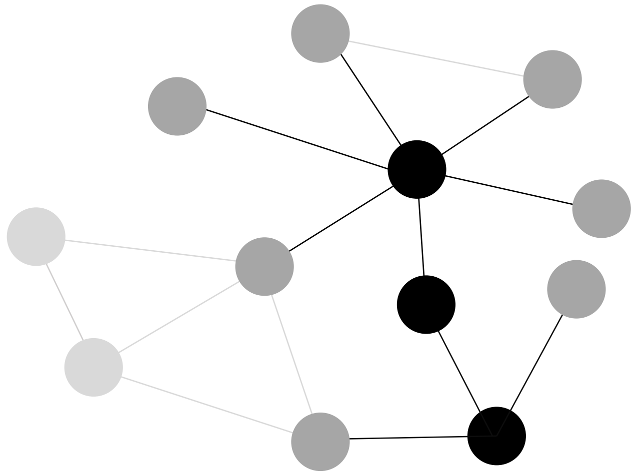


Goal: generate many random nodes with as few queries as possible

The problem: Sampling multiple nodes

Start at single random node

Explore graph through **query access**: querying node reveals its **neighbors**

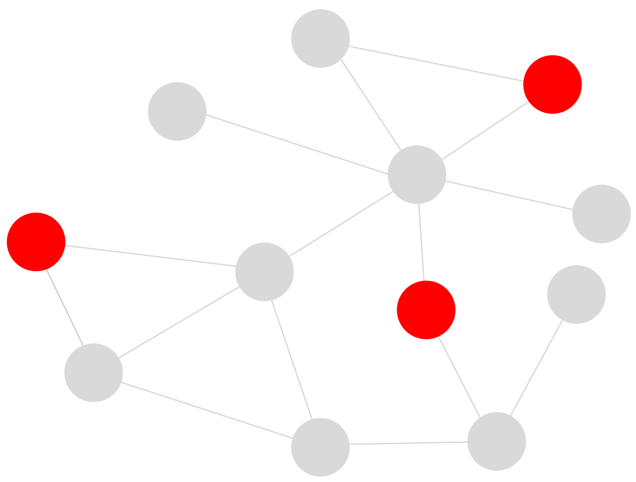


Goal: generate many random nodes with as few queries as possible

The problem: Sampling multiple nodes

Start at single random node

Explore graph through **query access**: querying node reveals its **neighbors**



Goal: generate many random nodes with as few queries as possible:

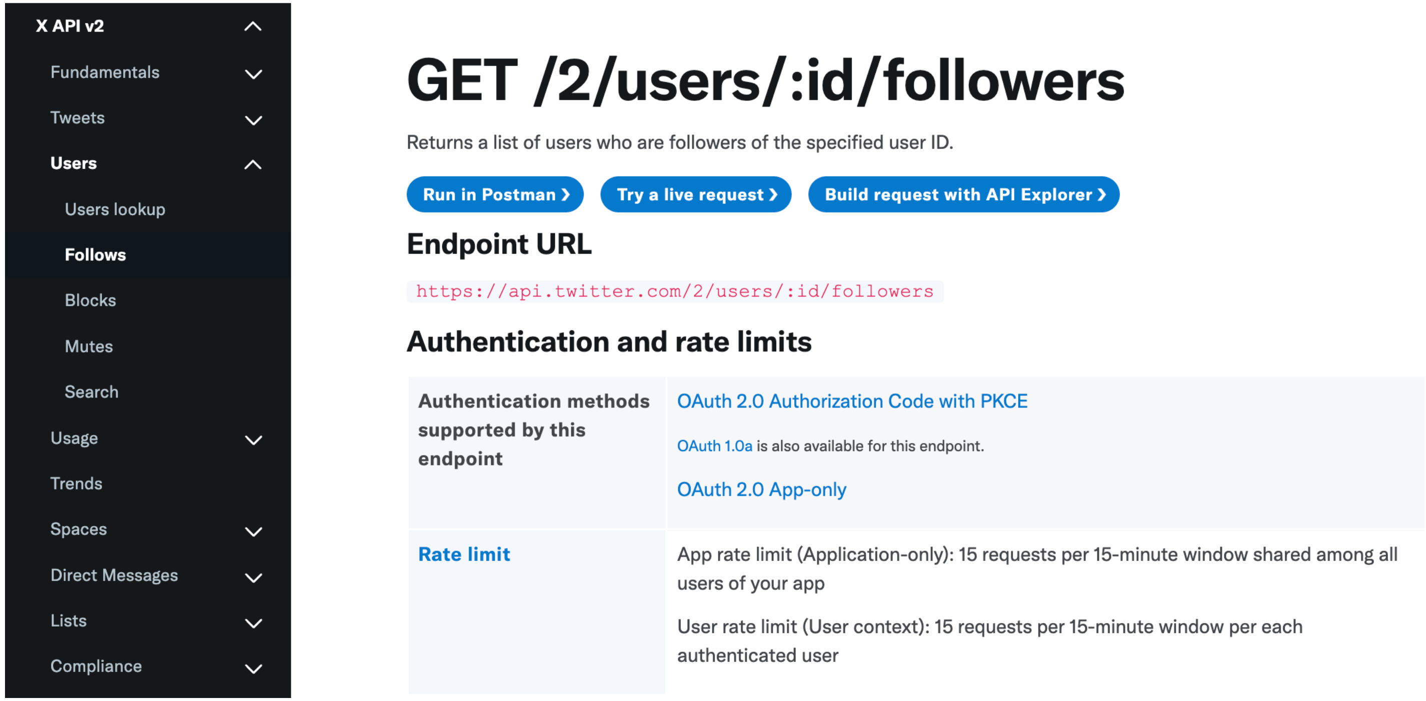
For $\epsilon > 0$ and $k \ll n$, return random S $\in \binom{V}{k}$

where $\Pr(S) \leq \frac{1+\epsilon}{\binom{n}{k}}$ for all $S \in \binom{V}{k}$

$|V| = n$

Motivation

- Many algorithms (e.g., page rank) assume access to **random nodes**.
- Exploring many different “parts” of a large network with few queries.
- Queries supported in modern social network APIs.



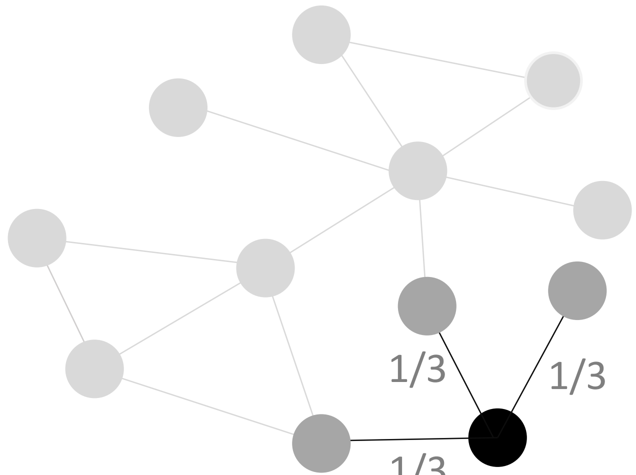
Retrieved from Twitter API (https://developer.x.com/en/docs/twitter-api/users/follows/api-reference), June 2024

Existing solution: Random walks

- Uniform random walk + rejection sampling generates one node ($k = 1$) in $O(d_{avg} t_{mix} \cdot \log 1/\epsilon)$ queries
  [Chierichetti, Dasgupta, Kumar, Lattanzi, Sarlos '16]

average degree

mixing time of lazy random walk

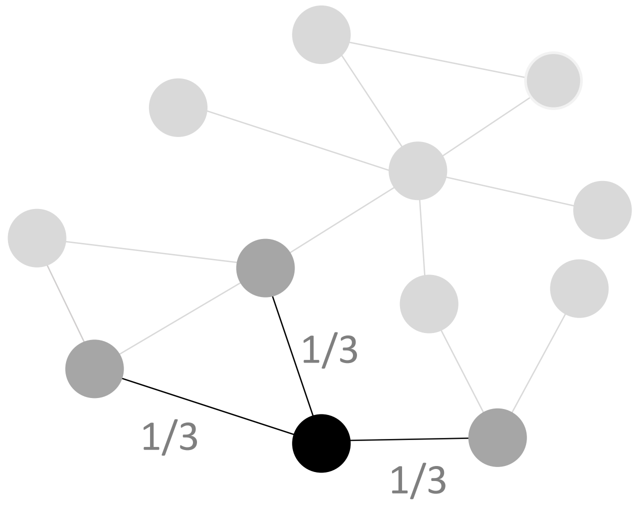


Existing solution: Random walks

- Uniform random walk + rejection sampling generates one node ($k = 1$) in $O(d_{avg} t_{mix} \cdot \log 1/\epsilon)$ queries
  [Chierichetti, Dasgupta, Kumar, Lattanzi, Sarlos '16]

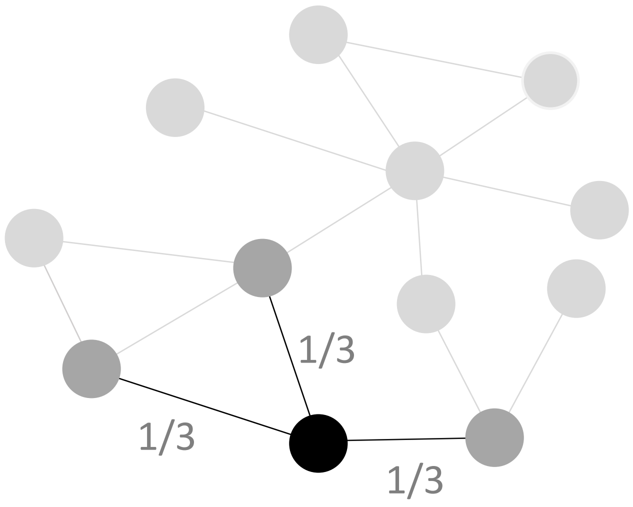
average degree

mixing time of lazy random walk



Existing solution: Random walks

- Uniform random walk + rejection sampling generates one node ($k = 1$) in $O(d_{avg} t_{mix} \cdot \log 1/\epsilon)$ queries [Chierichetti, Dasgupta, Kumar, Lattanzi, Sarlos ’16]
- Essentially optimal: $\Omega(d_{avg} t_{mix})$ lower bound (for specific graph) [Chierichetti, Haddadan ‘18]



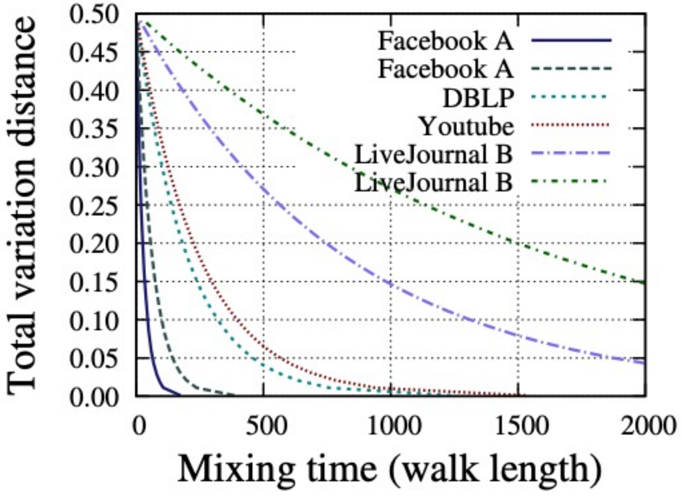
Existing solution: Random walks

- Uniform random walk + rejection sampling generates one node ($k = 1$) in $O(d_{avg} t_{mix} \cdot \log 1/\epsilon)$ queries [Chierichetti, Dasgupta, Kumar, Lattanzi, Sarlos '16]
- Essentially optimal: $\Omega(d_{avg} t_{mix})$ lower bound (for specific graph) [Chierichetti, Haddadan '18]

Can we do better (amortized) as k grows?

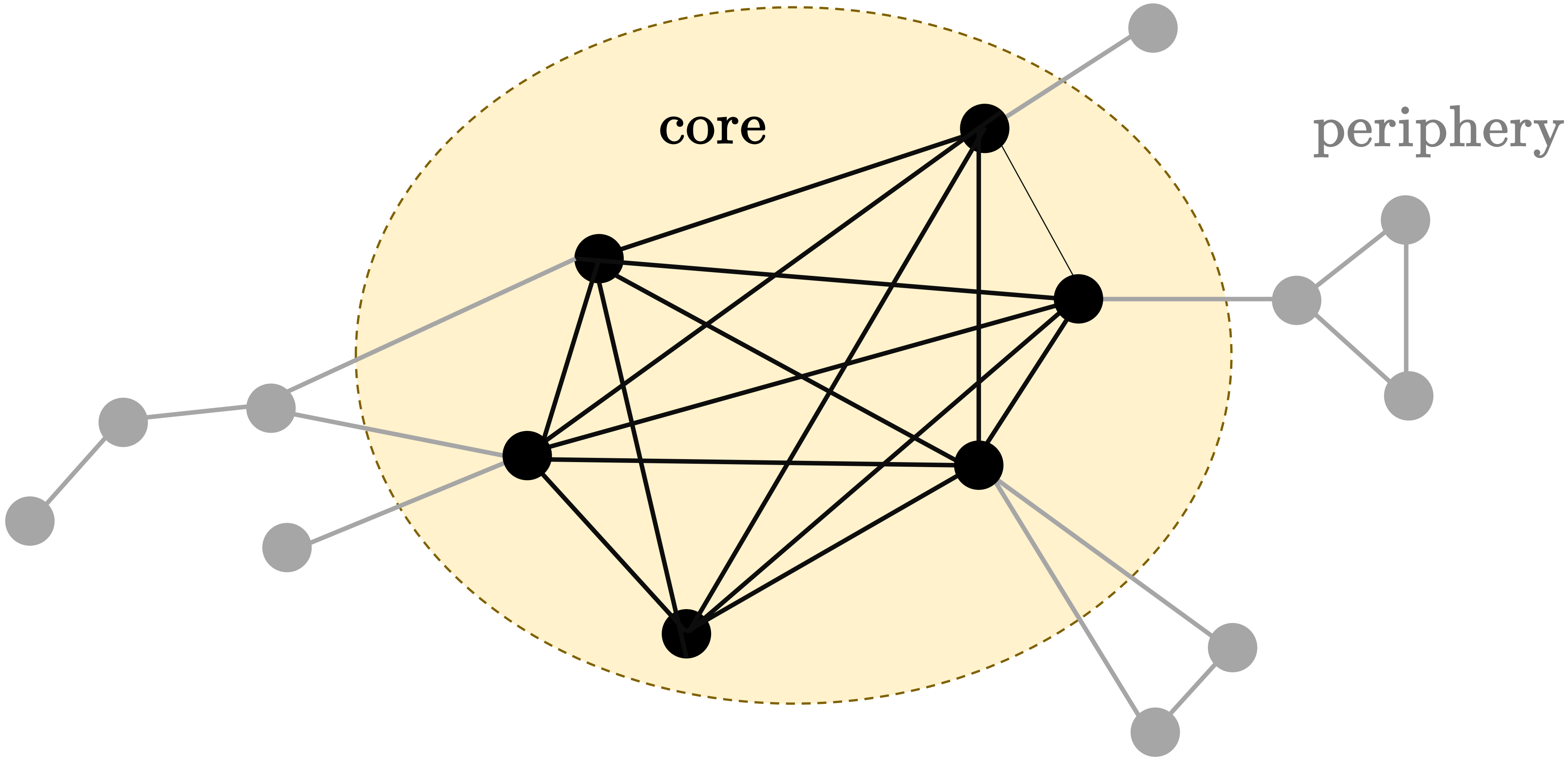
Mixing time can be several 100's or more [DR'09,MYK'10,QXZZ'20]

Some theoretical models have $\Theta(\log^2 n)$ mixing time, e.g., Newman-Watts [Dur'10, AL'12, KRS'15]



[Mohaisen-Yun-Kim, '10]

Let's use core-periphery structure!

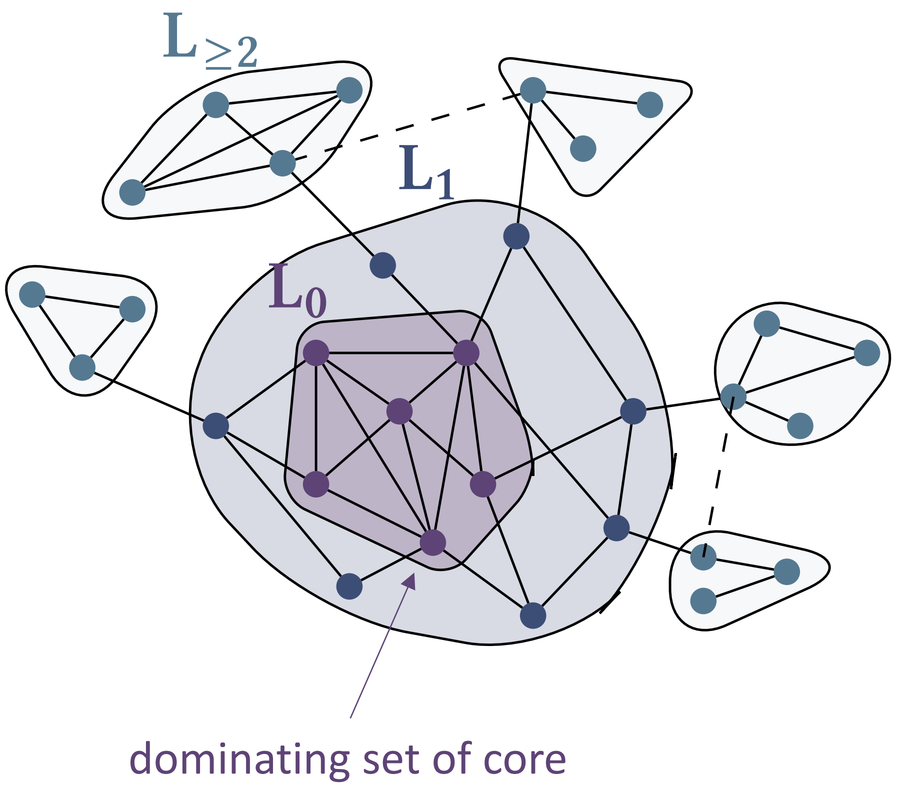



HEURISTICS
AHEAD

SampLayer: New node sampling algorithm

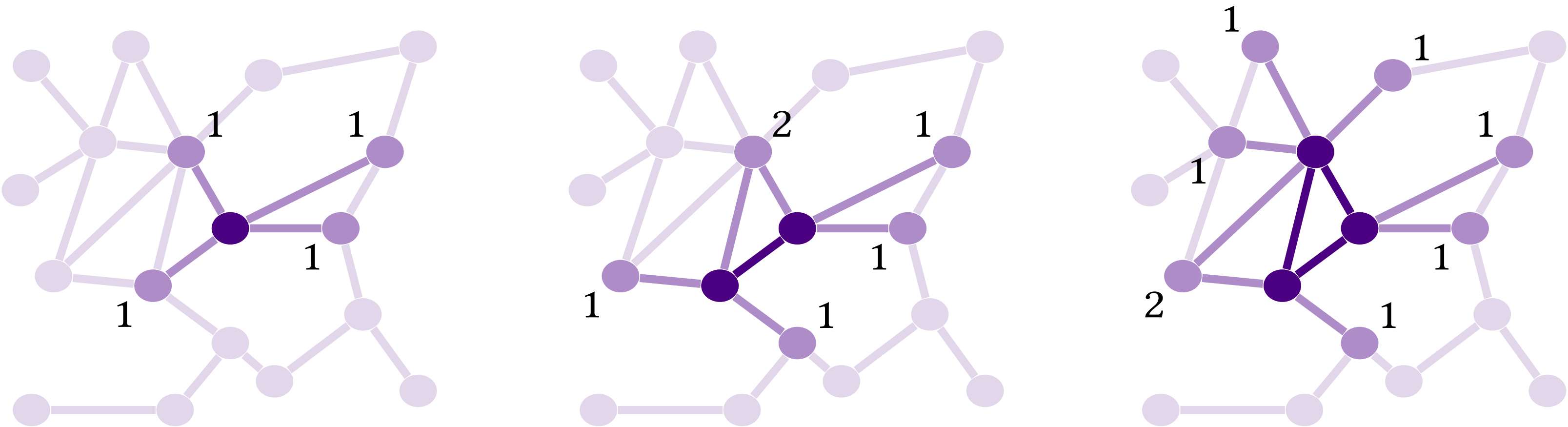
[B., Eden, Oren, Fotakis, WSDM'22]

- **Preprocessing**: Greedily search for "most influential" nodes in network, L_0.
- **Layering & Calibrating**: implicitly partition network into three layers: L_0, L_1, and the periphery $L_{\geq 2}$.
- **Sampling** by length 2 walks from L_0 to $L_{\geq 2}$ + local BFS in $L_{\geq 2}$ + rejection.



Phase 1: Greedy core construction

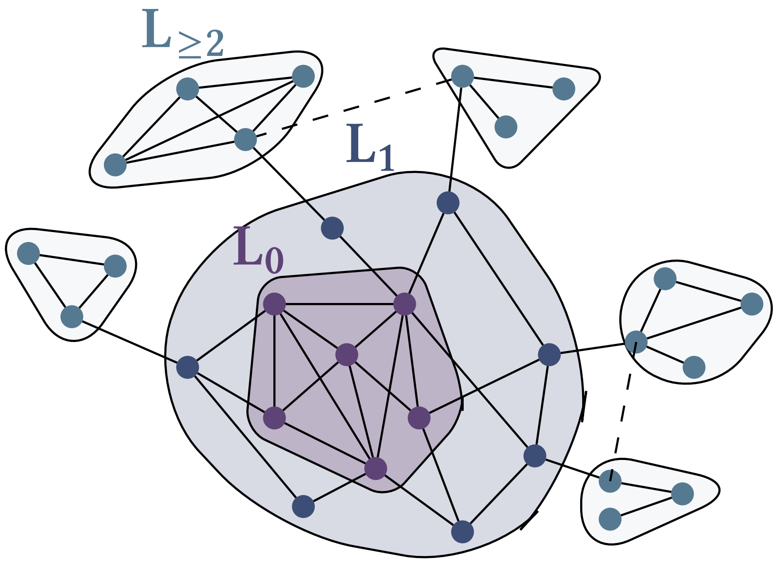
Starting from single node, construct L_0 by repeatedly adding node v with highest "perceived degree" and querying v.



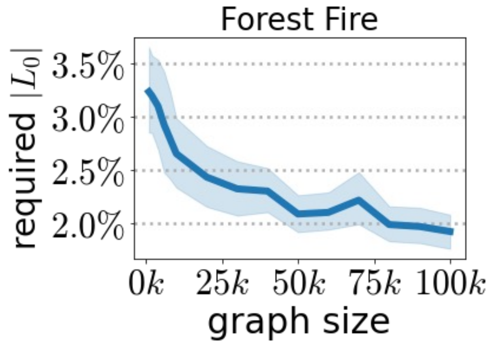
Phase 2: Structural layering

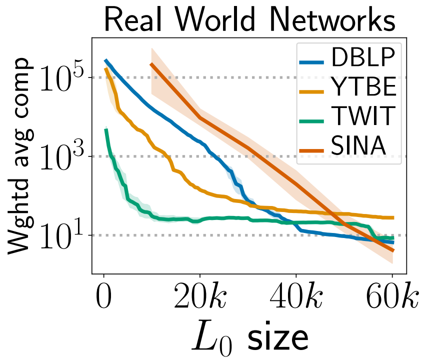
L_1 : all neighbors of L_0

$L_{\geq 2}$: all other nodes in network



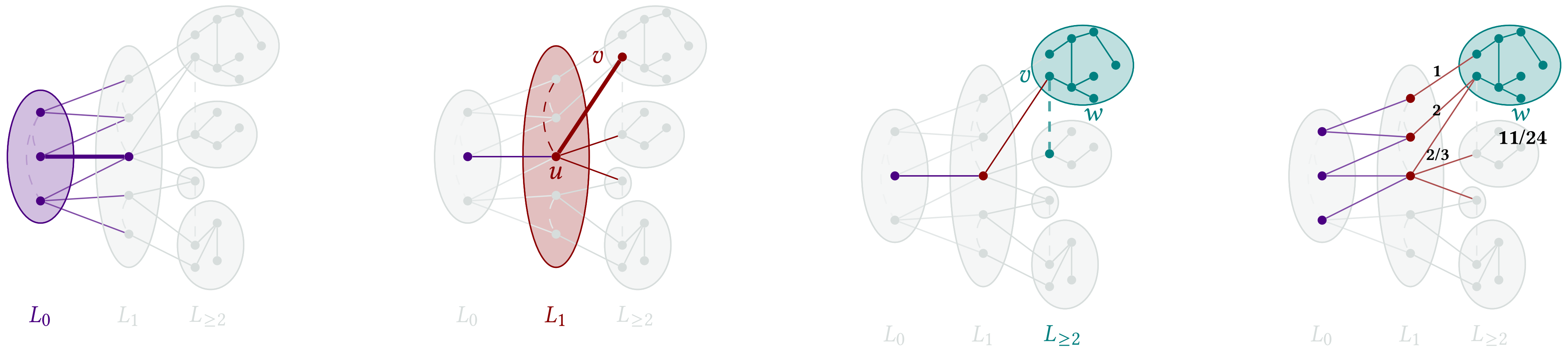
Key observation: sublinear-sized L_0 can decompose $L_{\geq 2}$ into **tiny components**!





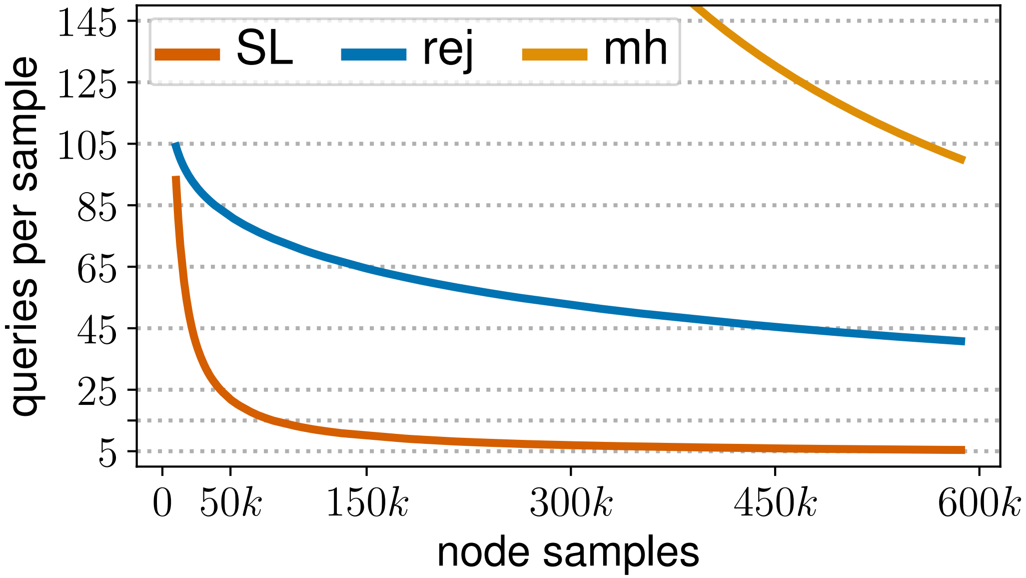
Phase 3: Sampling

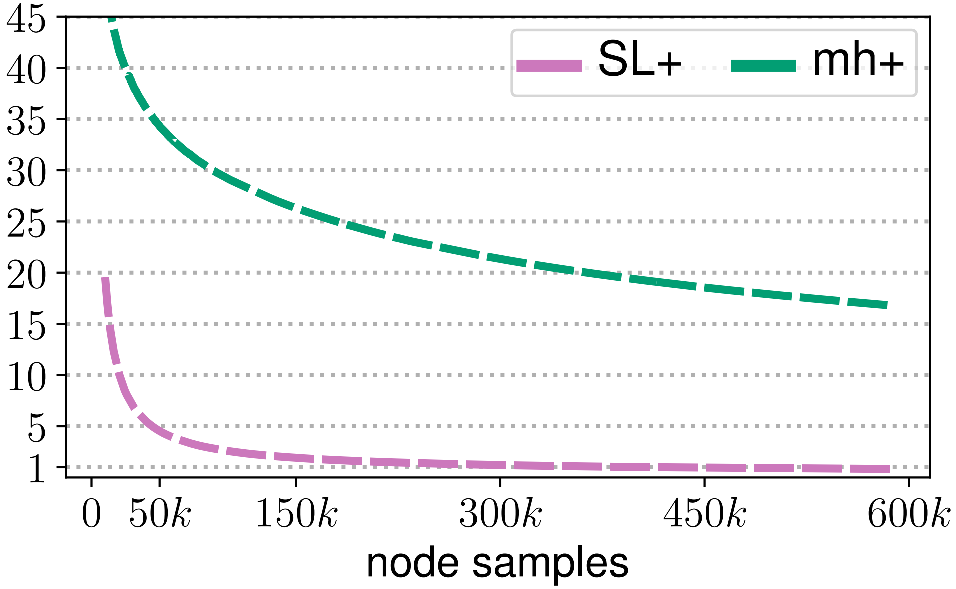
- Sampling from $L_0 \cup L_1$ straightforward.
- Sampling from $L_{\geq 2}$ by **length-2 walk** + **BFS** + **rejection**



Empirical results: SampLayer vs. random walks

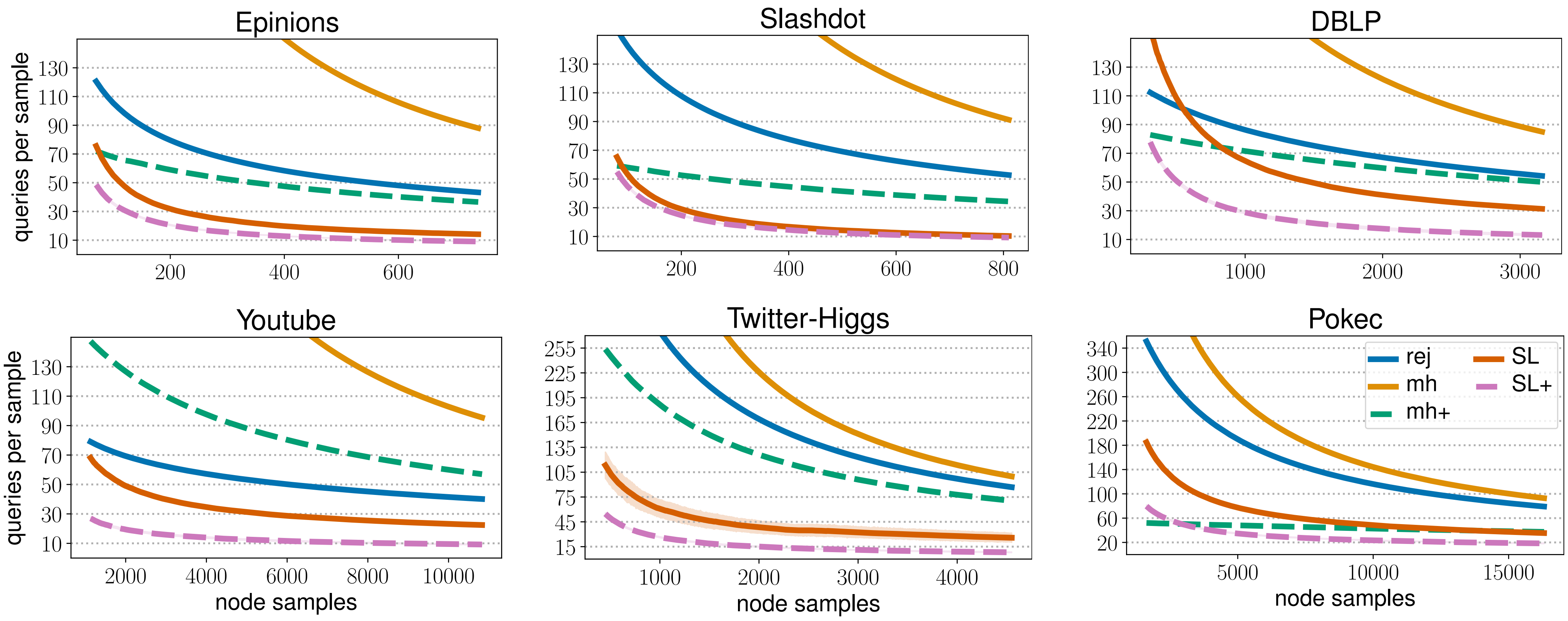
- Sina Weibo [ZYLX'14], social network with ≈ **60M nodes, 260M edges**





Empirical results: SampLayer vs. random walks

- Other social & information networks



Part II: Computing paths

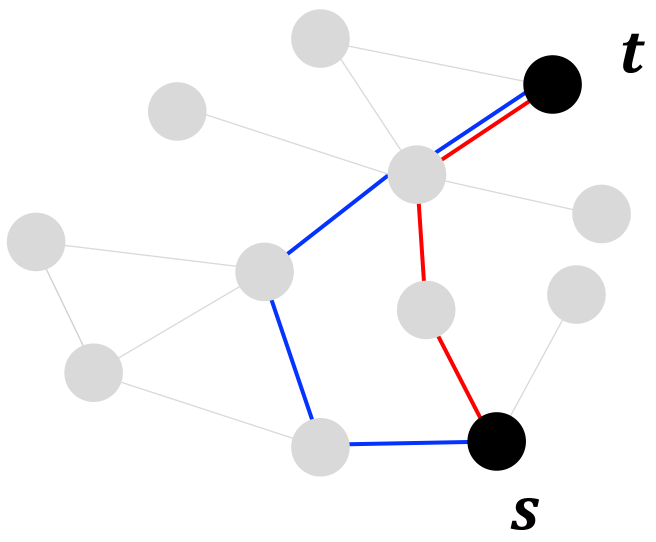
The problem

Basic task: Given undirected G, build data structure to support shortest* path queries $SP(s,t)$**

Motivation: centrality estimation, graph learning, …

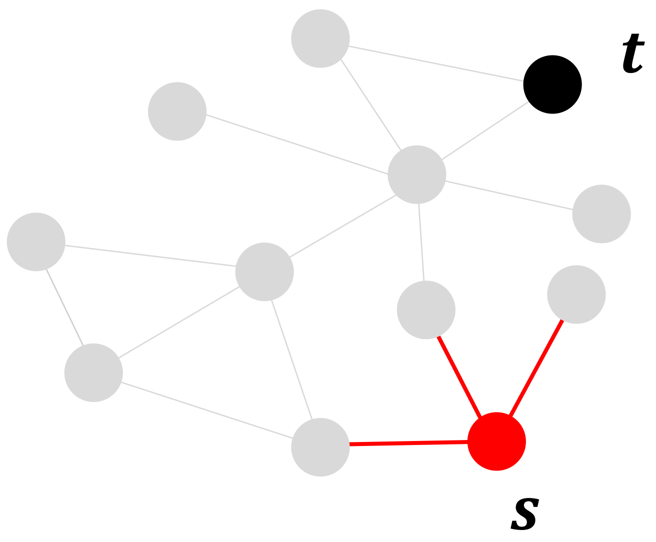
* allow for small additive error

** (s,t) pairs unknown during construction



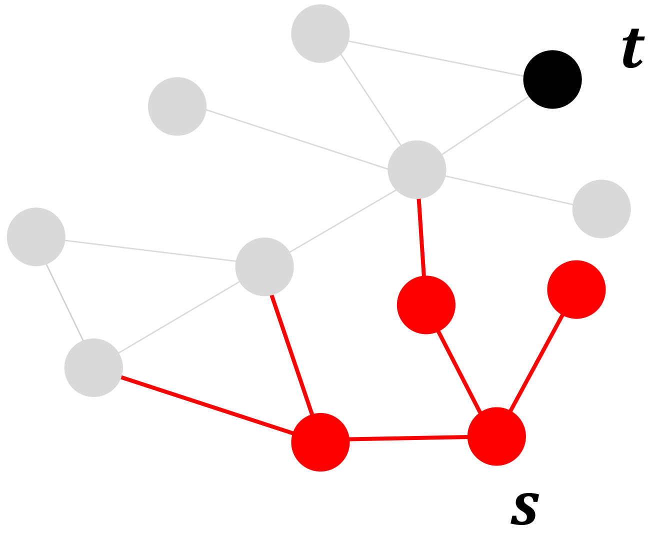
Case I: Single query

Solution 1: BFS from s
Running time: $O(m)$



Case I: Single query

Solution 1: BFS from s
Running time: $O(m)$



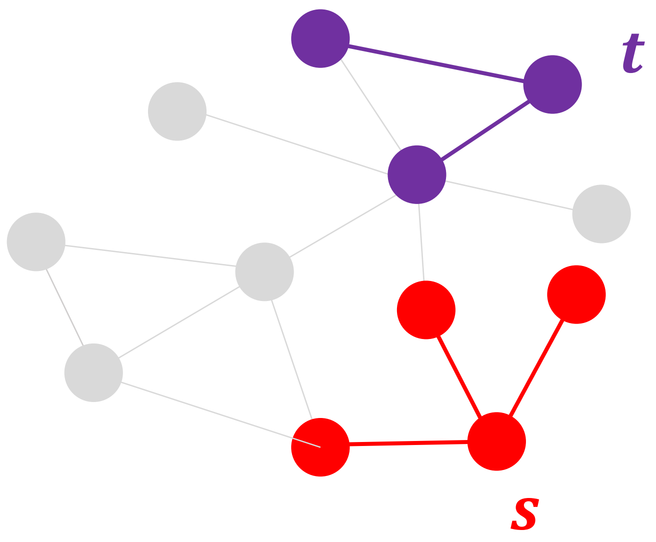
Case I: Single query

Solution 1: BFS from s
Running time: $O(m)$

Solution 2: **Bidirectional BFS** from $\boldsymbol{s}$ and from $\boldsymbol{t}$

Running time: **sublinear** under mild assumptions!

[BN'19, BFFKMT'22, AGJL'23]



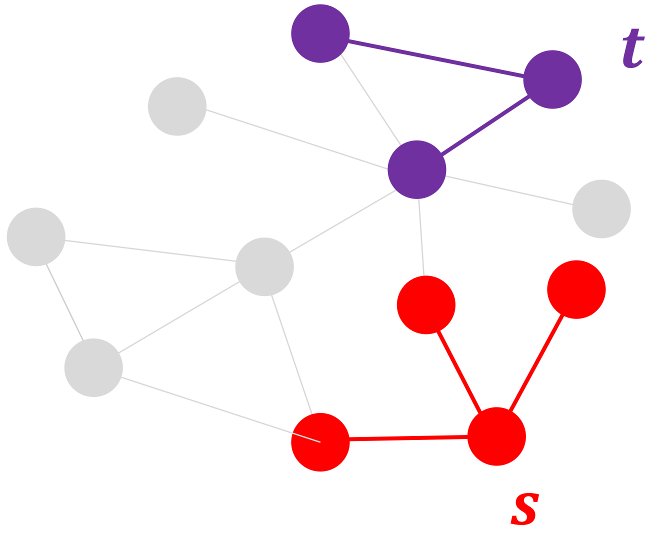
Case I: Single query

Solution 1: BFS from s
Running time: $O(m)$

Solution 2: **Bidirectional BFS** from $\boldsymbol{s}$ and from $\boldsymbol{t}$

Running time: **sublinear** under mild assumptions!

[BN'19, BFFKMT'22, AGJL'23]



LB: $n^{\Omega(1)}$ time per query on random models **without preprocessing**

[Alon, Grønlund, Jørgensen, Larsen '23] + [Basu, Kōshima, Eden, B., Seshadhri '24]

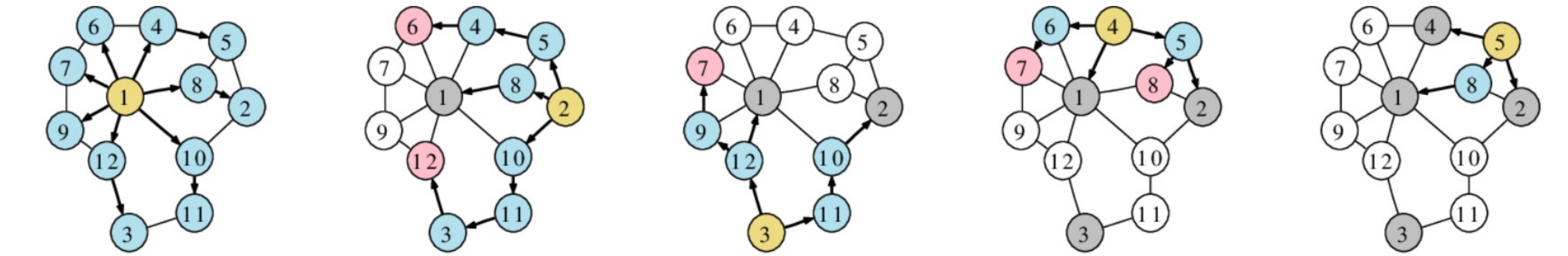
In practice, doesn't scale well with # queries

Case II: "∞" queries

Landmark-based solutions: pre-compute distances* from landmarks $\mathcal{L}$ to accelerate real-time queries via 2-hop labeling

[Akiba, Iwata, Yoshida '13], [Zhang, Yu, Goel '19], [Li, Qiao, Qin, Zhang, Chang, Lin '20], [Zhang, Li, Yuan, Qin, Zhang, Chang' 22], [Wang, Wang, Koehler, Lin '22], …

*paths can be reconstructed with extra effort



(a) First BFS from vertex 1. We visited all the vertices.
(b) Second BFS from vertex 2. We did not add labels to five vertices.
(c) Third BFS from vertex 3. We only visited the lower half of the vertices.
(d) Fourth BFS from vertex 4. This time we only visited the higher half.
(e) Fifth BFS from vertex 5. The search space was even smaller.

Example from PLL paper [Akiba, Iwata, Yoshida '13]

Case II: "∞" queries

Landmark-based solutions: pre-compute distances* from landmarks $\mathcal{L}$ to accelerate real-time queries via 2-hop labeling

[Akiba, Iwata, Yoshida '13], [Zhang, Yu, Goel '19], [Li, Qiao, Qin, Zhang, Chang, Lin '20], [Zhang, Li, Yuan, Qin, Zhang, Chang' 22], [Wang, Wang, Koehler, Lin '22], …

*paths can be reconstructed with extra effort

+ Very fast query time ($< 10^{-3}$s for large graphs)
− **Costly preprocessing:** space complexity up to $O(n^2)$, difficult to scale in practice

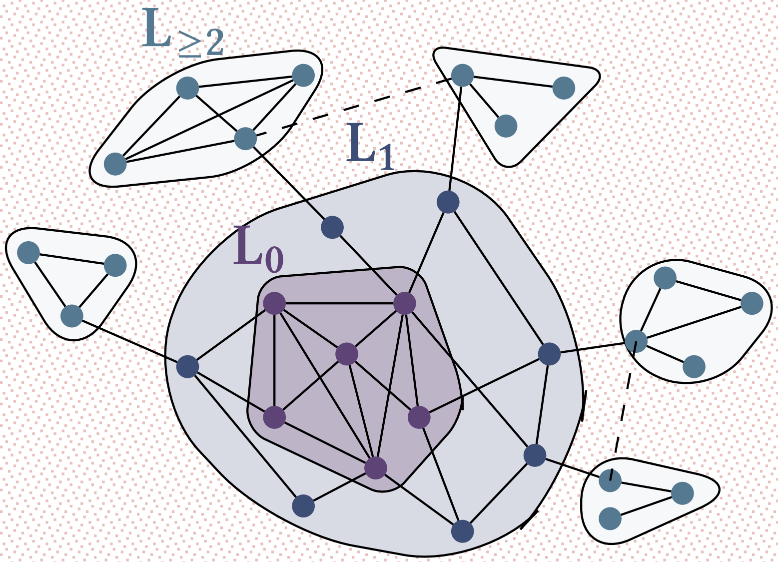
Open: Good query time **and** scalable preprocessing?

Case II: “∞” queries

Lan ks $\mathcal{L}$
to

[Ak Li,
Yua

*pa



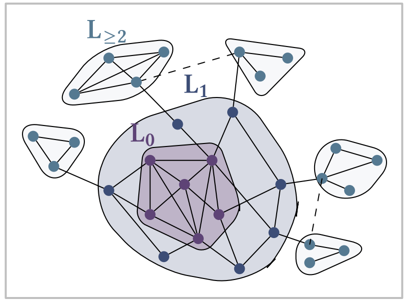
to the rescue!

✚ Ve

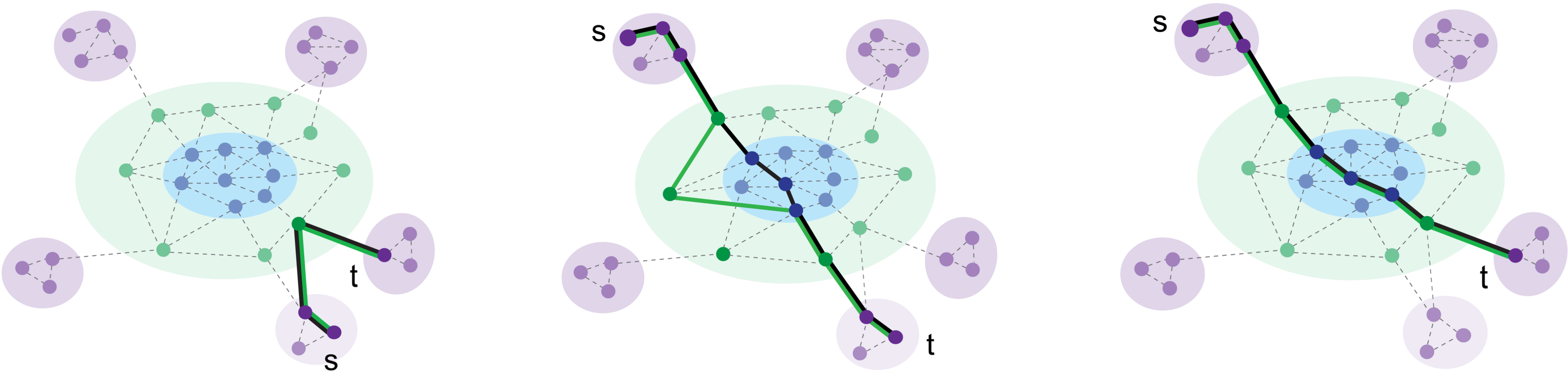
▬ **Co** le in
pra

Open: Good query time **and** scalable preprocessing?

Our algorithm: WormHole [Basu, Kōshima, Eden, B., Seshadhri '24]

- **Preprocessing**: construct  with greedy L_0

- **Per query:** run BiBFS from s and t until BFS trees reach L_0 or collide
  - If reached L_0, find shortest path inside L_0

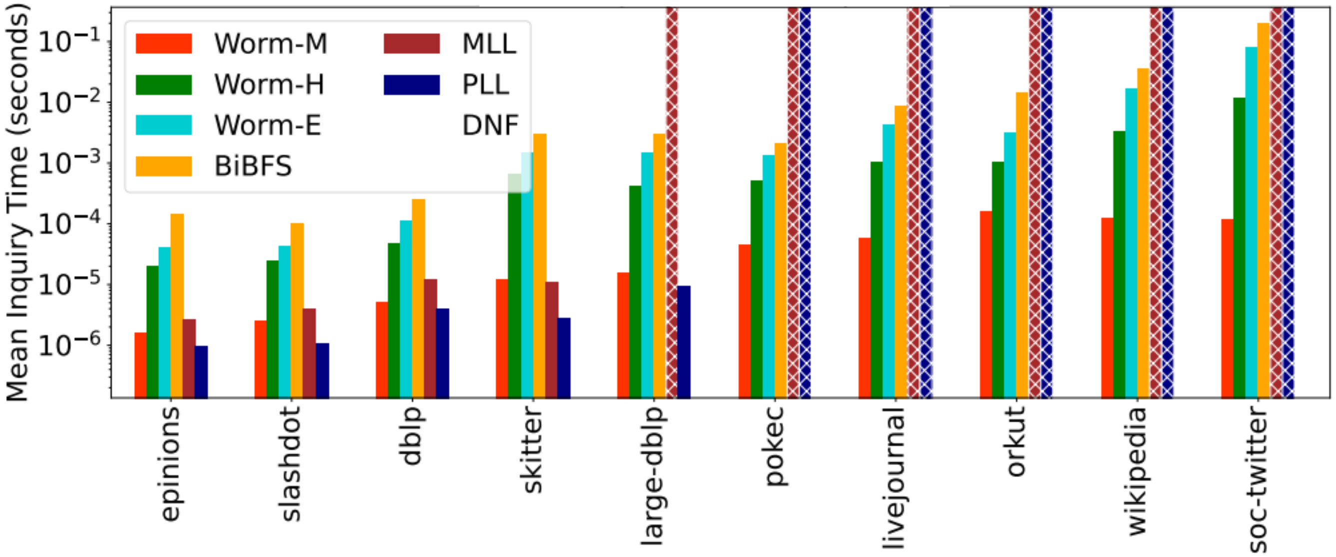


Some empirical results

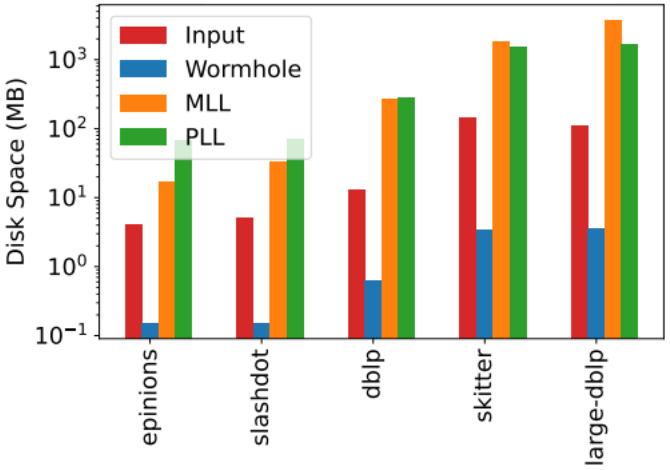
graphs

Network	$\lvert V \rvert$	$\lvert E \rvert$
epinions	$7.6\cdot10^4$	$5.1\cdot10^5$
slashdot	$7.9\cdot10^4$	$5.2\cdot10^5$
dblp	$3.2\cdot10^5$	$1.0\cdot10^6$
skitter	$1.7\cdot10^6$	$1.1\cdot10^7$
large-dblp	$1.8\cdot10^6$	$2.9\cdot10^7$
soc-pokec	$1.6\cdot10^6$	$3.1\cdot10^7$
soc-live	$4.8\cdot10^6$	$6.8\cdot10^7$
soc-orkut	$3.1\cdot10^6$	$1.2\cdot10^8$
wikipedia	$1.4\cdot10^7$	$4.4\cdot10^8$
soc-twitter	$4.2\cdot10^7$	$1.5\cdot10^9$

query time



space cost



additive error (stretch)

	BiBFS	WormHole$_E$					WormHole$_H$				
Network	MIT	MIT	SU/I	+0(%)	≤+1 (%)	≤+2 (%)	MIT	SU/I	+0(%)	≤+1 (%)	≤+2 (%)
epinions	144	41	4.5	98.06	99.99	100.00	20	24	66.97	99.54	100.00
slashdot	99	46	2.8	73.43	95.37	99.28	24	14	63.09	98.78	99.98
dblp	247	110	2.4	97.02	99.96	100.00	48	11	44.72	82.42	96.53
skitter	3004	1439	2.3	94.71	99.89	100.00	660	24	58.99	96.78	99.98
large-dblp	3041	1447	2.3	85.37	99.10	99.95	417	21	47.61	89.74	99.04
pokec	2142	1317	1.8	51.37	92.15	99.63	506	11	14.52	59.51	90.71
livejournal	8565	4318	2.1	71.98	97.95	99.86	1054	29	28.86	77.93	97.83
orkut	14k	3213	4.4	58.50	94.56	99.64	1030	35	20.66	68.11	93.93
wikipedia	35k	17k	2.4	94.94	99.92	100.00	3394	36	44.65	98.74	100.00
soc-twitter	204k	81k	3.4	93.30	99.98	100.00	12k	181	35.13	99.30	99.99

Main theoretical result

Theorem [Basu, Kōshima, Eden, B., Seshadhri '24]**:**
On Random Chung-Lu graph with power law parameter $2 < \beta < 3$,
WormHole achieves whp:

- Preprocessing query complexity of $o(n)$
- Query complexity and time $n^{o(1)}$ for a single pair query (s, t)
- Additive error of $O(\log \log n)$ in worst case

Main theoretical result

Theorem [Basu, Kōshima, Eden, B., Seshadhri '24]**:**
On Random Chung-Lu graph with power law parameter $2 < \beta < 3$,
WormHole achieves whp:

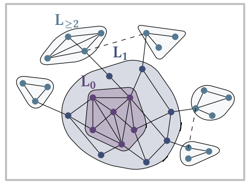
- Preprocessing query complexity of $o(n)$
- Query complexity and time $n^{o(1)}$ for a single pair query (s, t)
- Additive error of $O(\log\log n)$ in worst case

$n^{\Omega(1)}$ for BiBFS

$\Omega(n)$ for landmark-based solutions

not clear if tight

Final remarks

- Strength of 

as explicit sublinear decomposition.
More applications to come!

- Stronger theoretical foundations? Connections to highway dimension?
[Abraham, Fiat, Goldberg, Werneck '10]

- Applications in distributed settings?

Thank you!

https://arxiv.org/abs/2110.13324
https://arxiv.org/abs/2406.08624