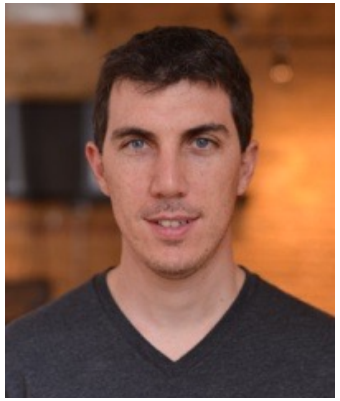
Practical Sublinear Algorithms for Node Sampling in Large Networks

Omri Ben-Eliezer
MIT

Joint with:

Talya Eden
BU/MIT -> Bar Ilan U

Joel Oren
General Motors



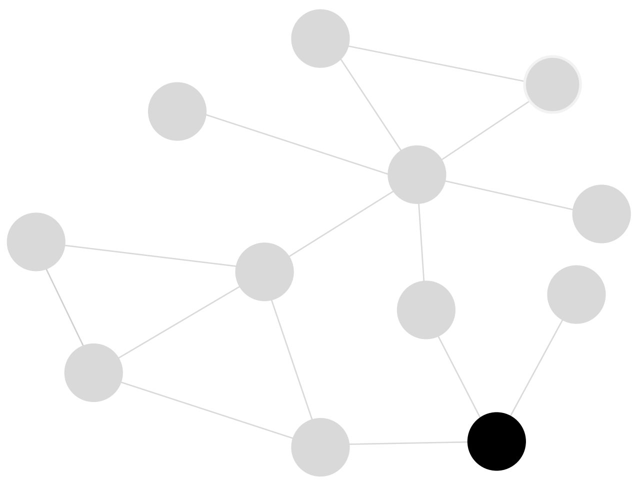
Dimitris Fotakis
Natl Tech U Athens

The problem: Sampling multiple nodes

Start at single random node

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

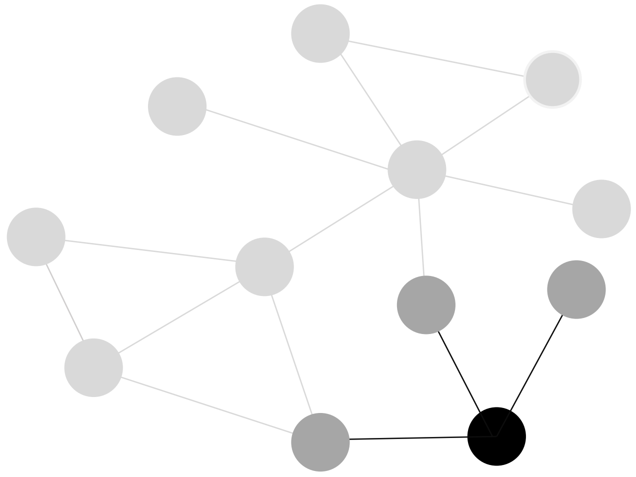
Goal: generate many random nodes with as few queries as possible



The problem: Sampling multiple nodes

Start at single random node

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



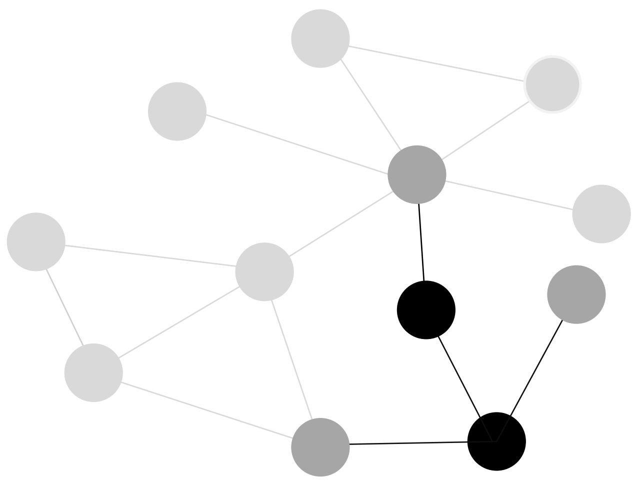
Goal: generate many random nodes with as few queries as possible

The problem: Sampling multiple nodes

Start at single random node

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

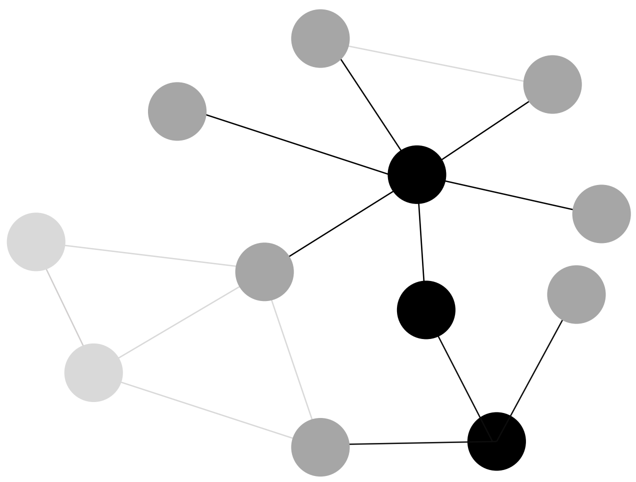
Goal: generate many random nodes
with as few queries as possible



The problem: Sampling multiple nodes

Start at single random node

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

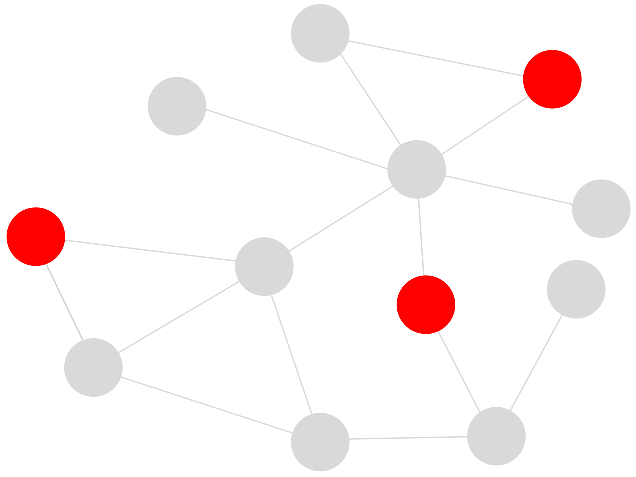


Goal: generate many random nodes with as few queries as possible

The problem: Sampling multiple nodes

Start at single random node

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



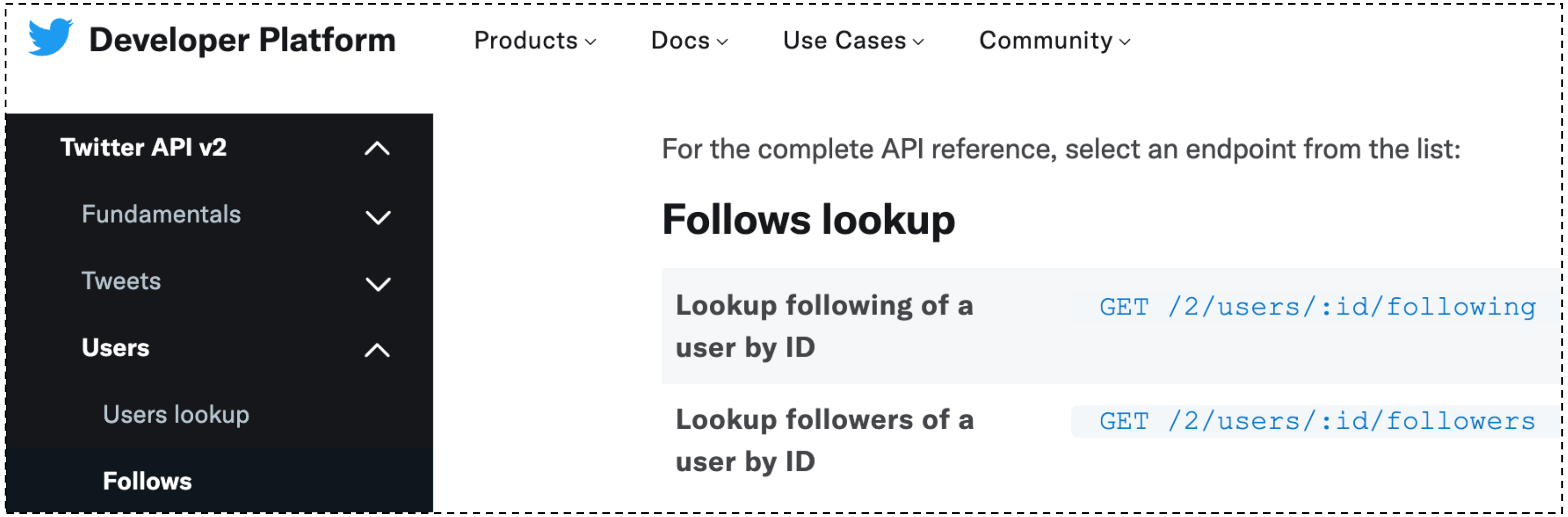
Goal: generate many random nodes
with as few queries as possible:

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 (sublinear-time / property testing, data mining, …) assume access to **random nodes**.
- Exploring many different "parts" of a large network with few queries.
- Queries supported in modern social network APIs.

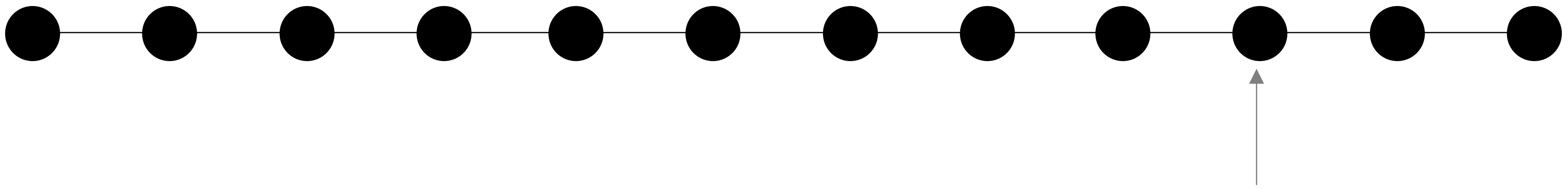


Retrieved from Twitter API (https://developer.twitter.com/en/docs/twitter-api/users/follows/api-reference), July 2022

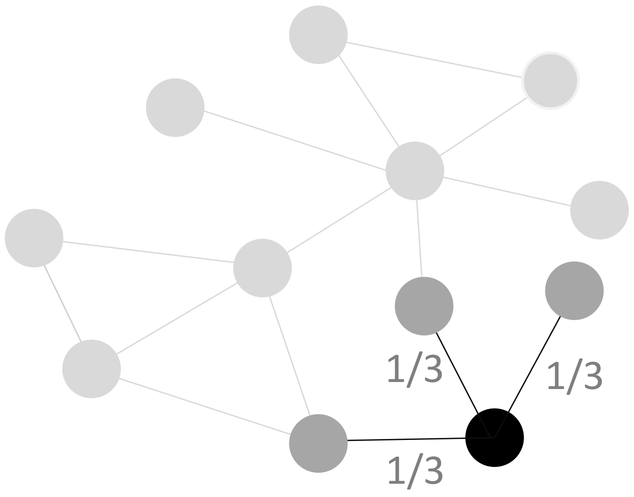
Solution I: BFS

- This talk: **real world** graphs (social networks). But let us start with some theoretical observations.

Trivial solution: Query all nodes, $O(n)$ query complexity. Tight (in worst case) even for sampling a single node!



Solution II: Random walks

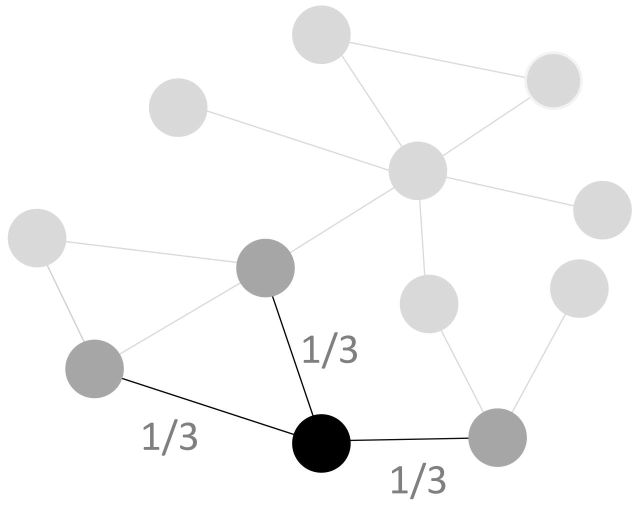


Solution II: Random walks

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

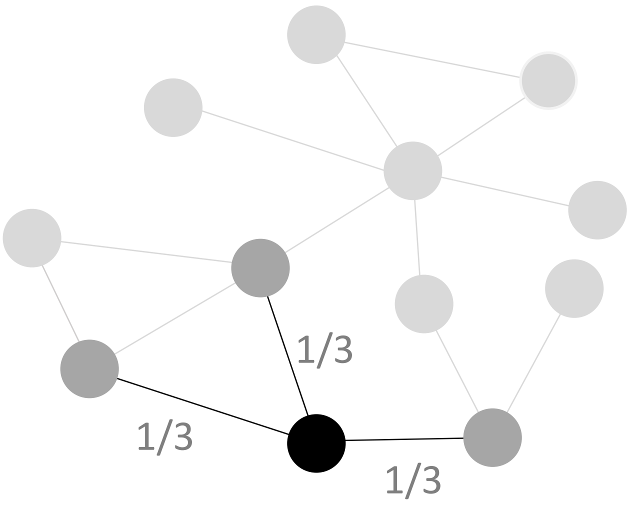
average degree

mixing time of uniform random walk



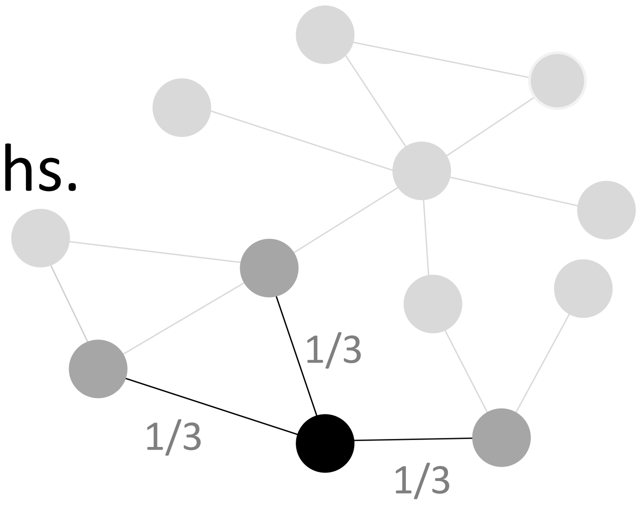
Solution II: Random walks

- Uniform random walk (+ rejection step) 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 some graphs) [Chierichetti, Haddadan '18]



Solution II: Random walks

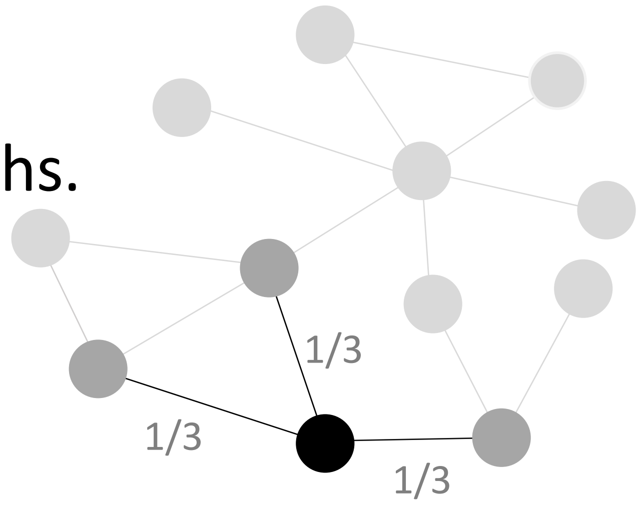
- Uniform random walk (+ rejection step) generates one node ($k = 1$) in $O(d_{avg} t_{mix} \cdot \log 1/\epsilon)$ queries [Chierichetti, Dasgupta, Kumar, Lattanzi, Sarlos ’16]
- Essentially optimal: $\Omega(d_{avg} t_{mix})$ lower bound (for some graphs) [Chierichetti, Haddadan ‘18]
- Not hard to show $\Omega(k \cdot t_{mix})$ lower bounds for sampling k nodes, for wide classes of realistic graphs.



Solution II: Random walks

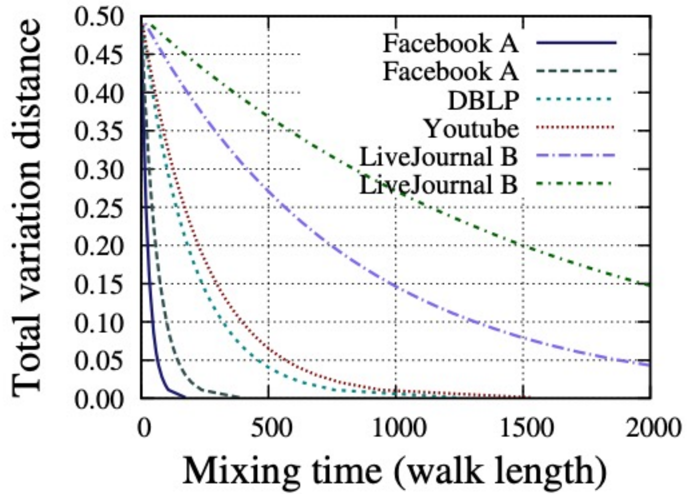
- Uniform random walk (+ rejection step) generates one node ($k = 1$) in $O(d_{avg} t_{mix} \cdot \log 1/\epsilon)$ queries [Chierichetti, Dasgupta, Kumar, Lattanzi, Sarlos '16]
- Essentially optimal: $\Omega(d_{avg} t_{mix})$ lower bound (for some graphs) [Chierichetti, Haddadan '18]
- Not hard to show $\Omega(k \cdot t_{mix})$ lower bounds for sampling k nodes, for wide classes of realistic graphs.

Can we do better than $O(k \cdot t_{mix})$ for large k?



Real-worlds social networks

- t_{mix} can be pretty large: several 100's or more [DR'09,MYK'10,QXZZ'20],
- Some small-world models have $\Theta(\log^2 n)$ mixing time, e.g., Newman-Watts [Dur'10, AL'12, KRS'15].

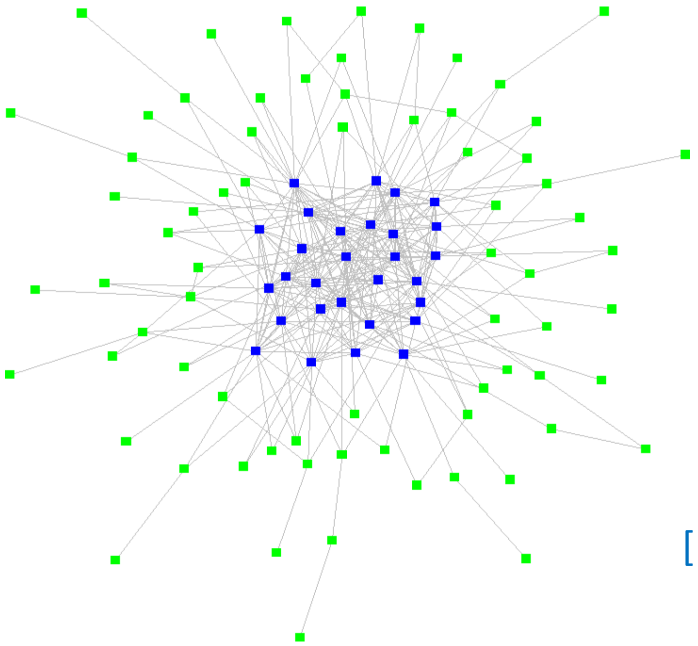


[Mohaisen-Yun-Kim, '10]

⇒ **Issue:** High amortized query complexity for random walk based algorithms!

Real-worlds social networks

- **Power law** degree distribution
- Highly expanding "**core**", isolated "**periphery**" components [BE'99, LLDM '09, RPFM'14, ZMN'15, BK'19, ...]



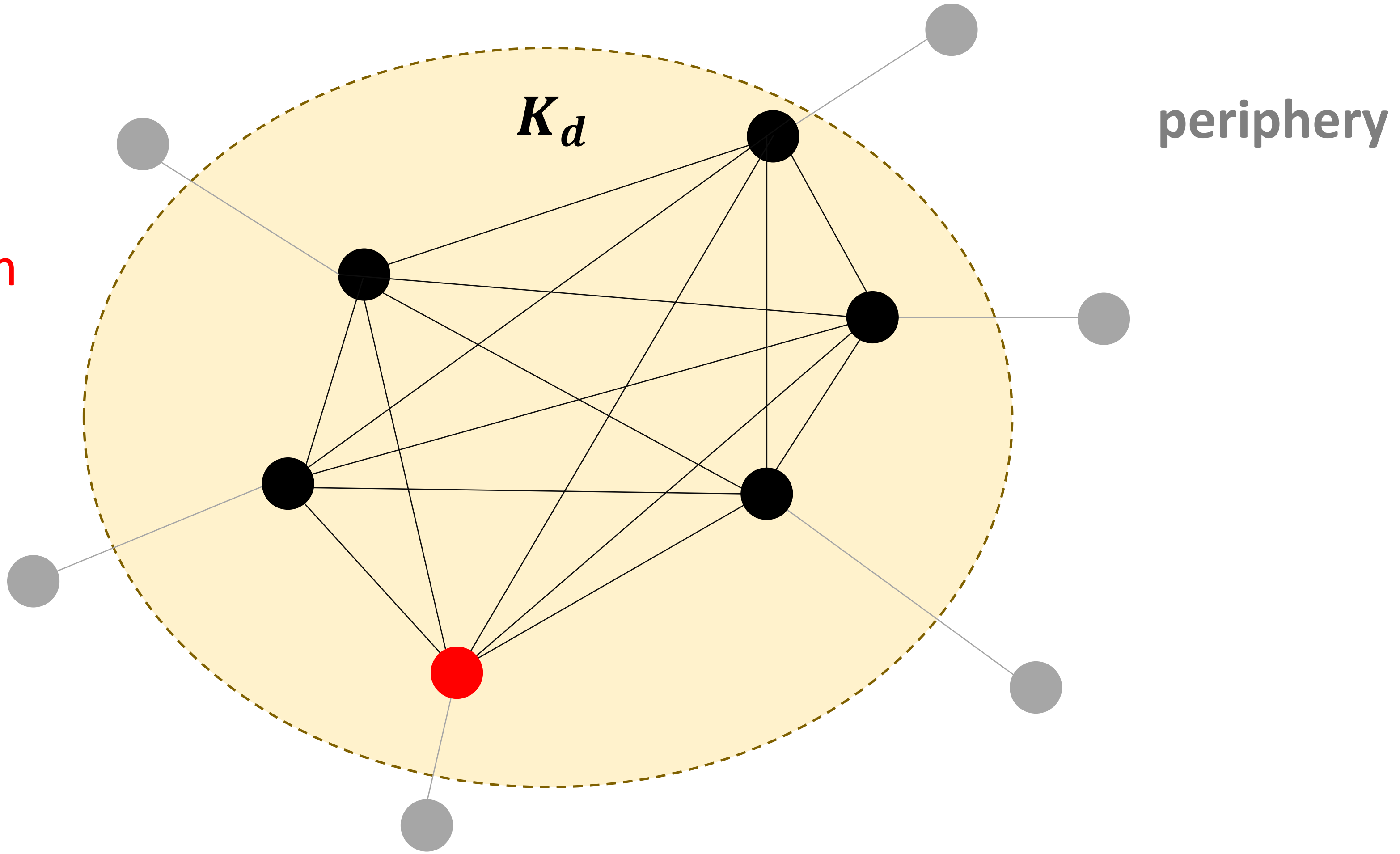
[Krebs-Holley, '06]

PROFILE	FOLLOWERS
Barack Obama	132,382,271
Justin Bieber	114,149,758
KATY PERRY	108,919,460
Rihanna	107,003,013
Cristiano Ronaldo	102,284,851
Elon Musk	102,170,738

[socialtracker.io]

Let’s use core-periphery structure!

Can you reach a random node in less than $t_{mix} = O(d)$ queries?



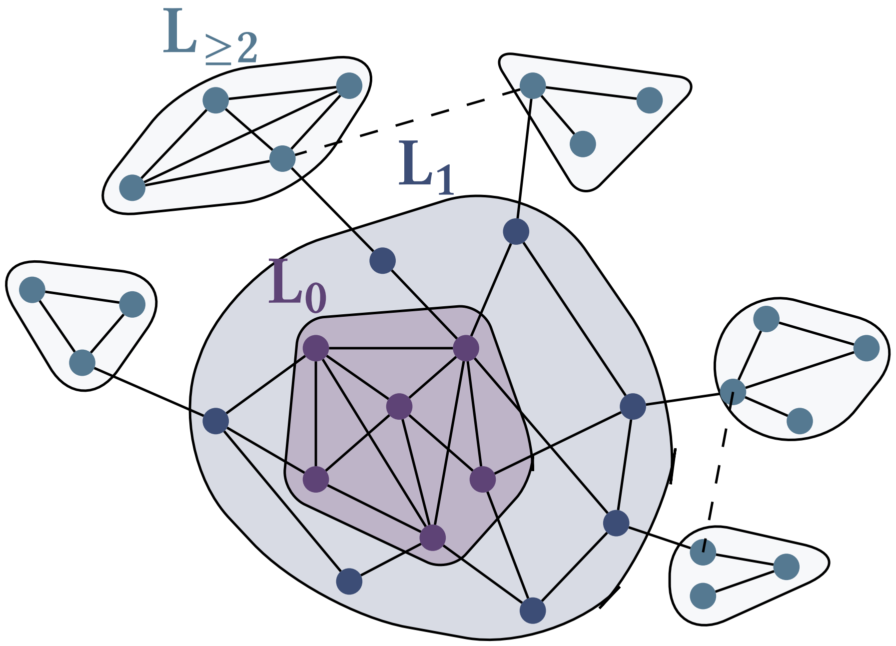

HEURISTICS
AHEAD
imgflip.com

SampLayer [BEFO'22]: New node sampling algorithm

- **Preprocessing**: Greedily search for "most influential" nodes in network, L_0.

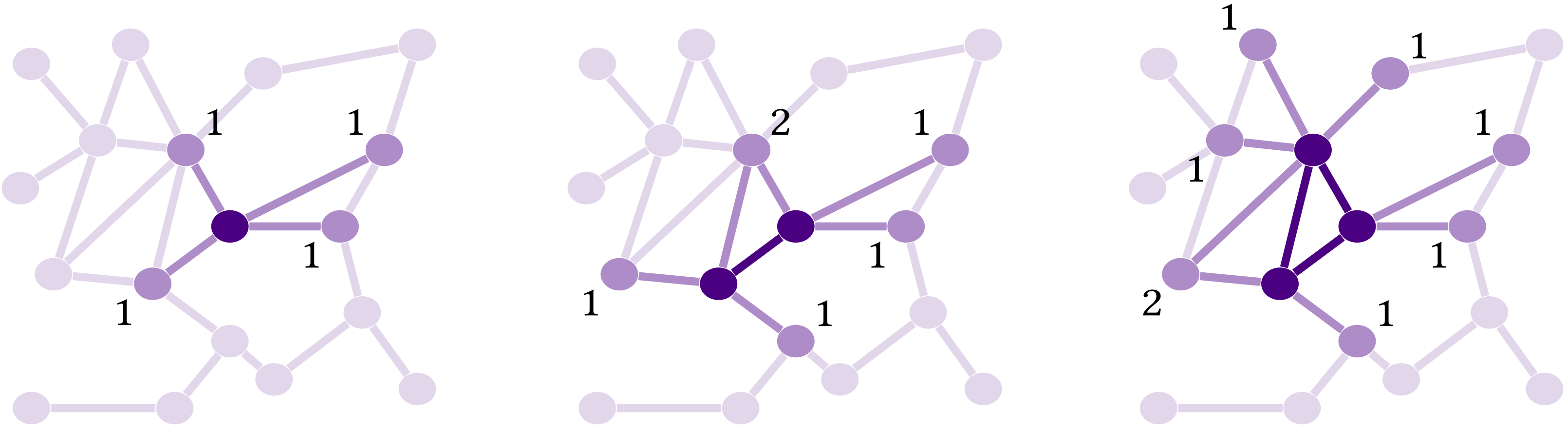
- **Layering & Calibrating**: implicitly partition network into three layers: L_0, L_1, and the periphery $L_{\geq 2}$.

- **Sampling** by length 2 walks from L_0 to $L_{\geq 2}$ + local BFS in $L_{\geq 2}$ + rejection.



Phase 1: Greedy core construction

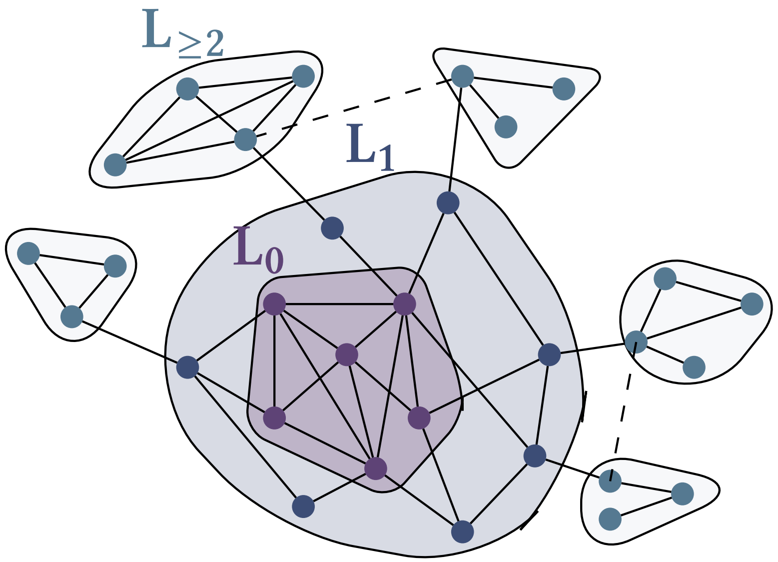
Starting from single node, construct L_0 by repeatedly adding node v with highest "perceived degree" and querying v.



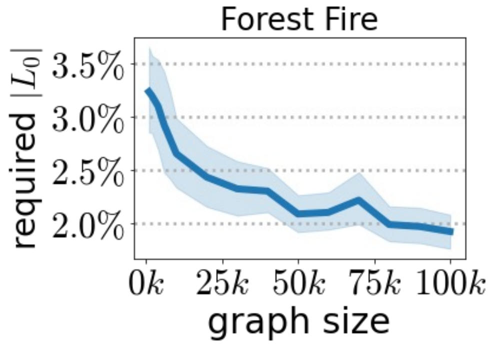
Phase 2: Structural layering

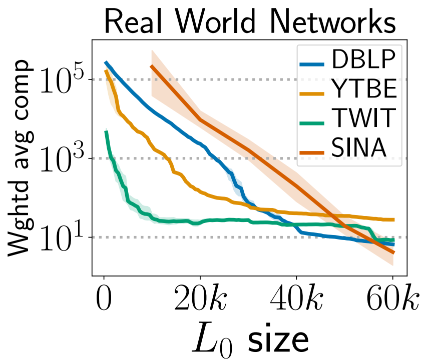
L_1 : all neighbors of L_0,

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



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

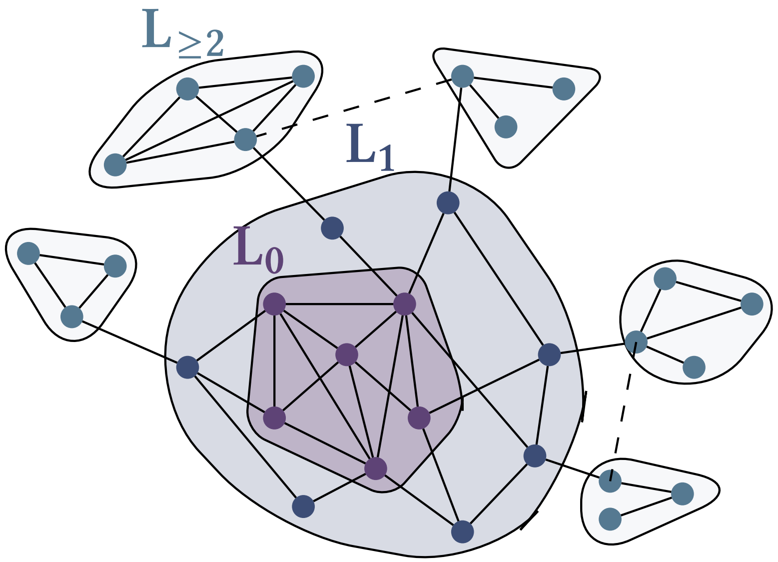




Phase 2: Structural layering

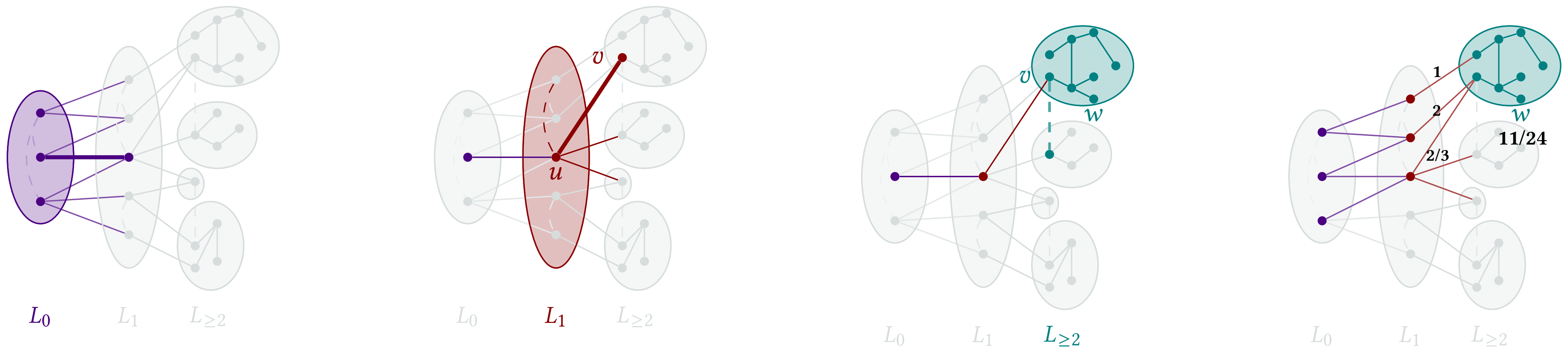
“Preparations” for sampling:

- Estimate $L_{\geq 2}$ size ($|L_0|$, $|L_1|$ known).
- Find a “reachability baseline” for $L_{\geq 2}$
    - Generated distribution will be uniform except for “low reachability” nodes.



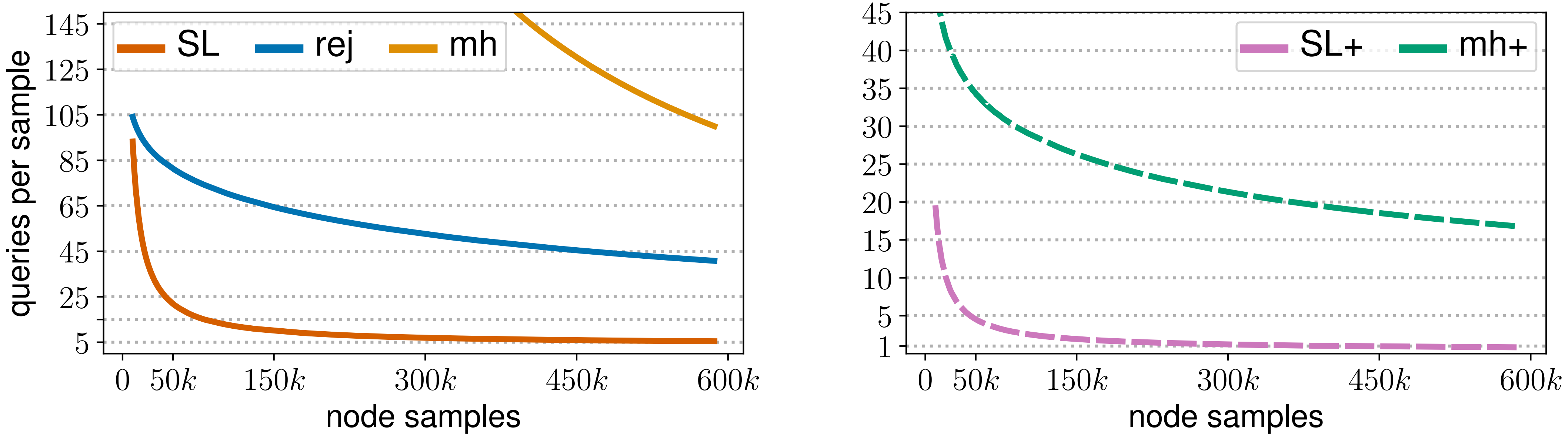
Phase 3: Sampling

- Sampling from $L_0 \cup L_1$ straightforward.
- Sampling from $L_{\geq 2}$ by **length-2 walk** between L_0 and $L_{\geq 2}$, then **BFS** in reached $L_{\geq 2}$ component.
  Finally, **rejection step** to ensure uniform probabilities.



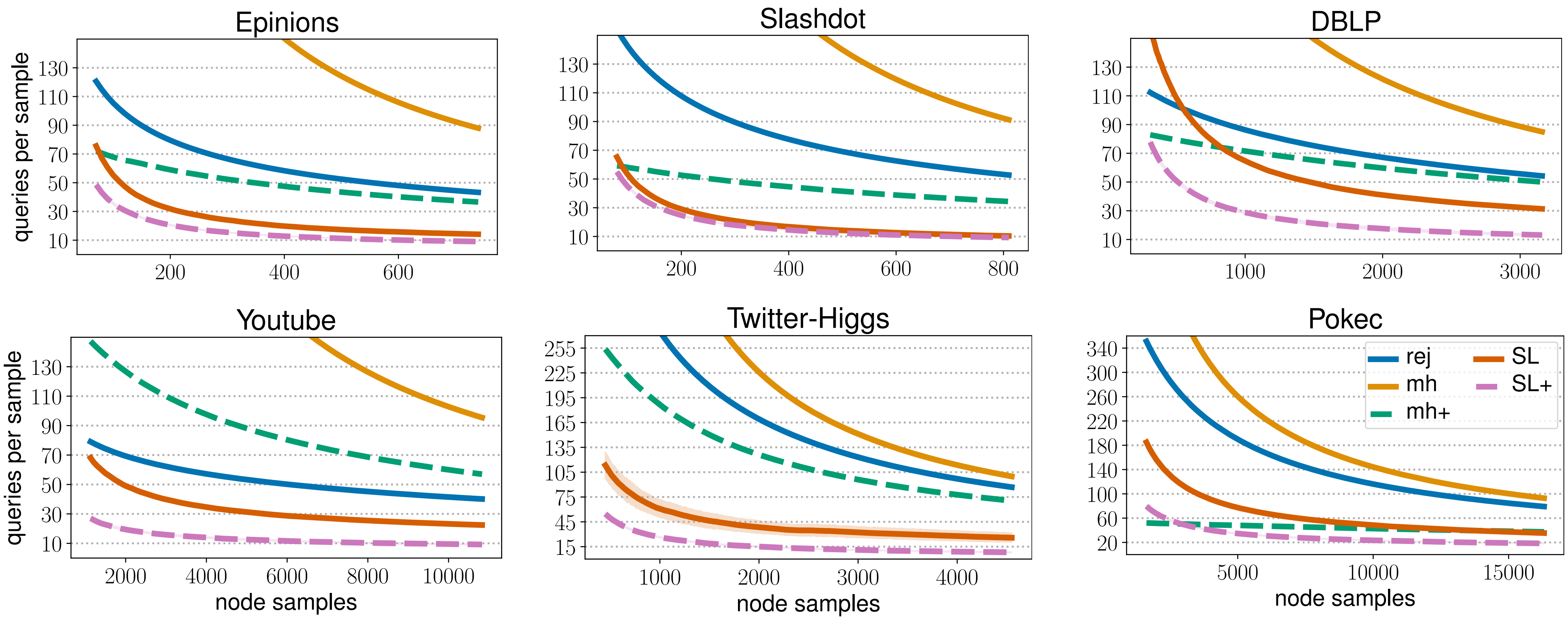
Empirical results: SampLayer vs random walks

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



Empirical results: SampLayer vs random walks

- Other social & information networks



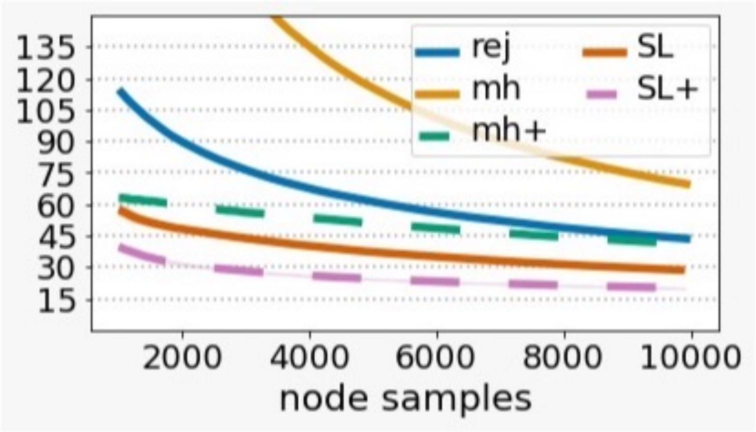
Empirical results: SampLayer vs random walks

Dataset	n	m	d_{avg}	L_0 size	
				SL	SL+
Epinions [47]	76K	509K	13.4	3K	1K
Slashdot [38]	82K	948K	23.1	3K	2K
DBLP [56]	317K	1.05M	6.62	30K	20K
Twitter-Higgs [12]	457K	14.9M	65.1	25K	10K
Forest Fire [36, 37]	1M	6.75M	13.5	10K	10K
Youtube [56]	1.1M	2.99M	5.27	30K	10K
Pokec [54]	1.6M	30.6M	37.5	200K	100K
SinaWeibo [58]	58.7M	261M	8.91	500K	100K

Table 1: The list of networks we considered with numbers of nodes (n), edges (m), their average degrees (d_{avg}), and L_0 sizes we selected for SampLayer and SampLayer+.

Empirical results: SampLayer vs random walks

- Forest Fire network model [LKF'05] with $p_f = 0.37, p_b = 0.3$



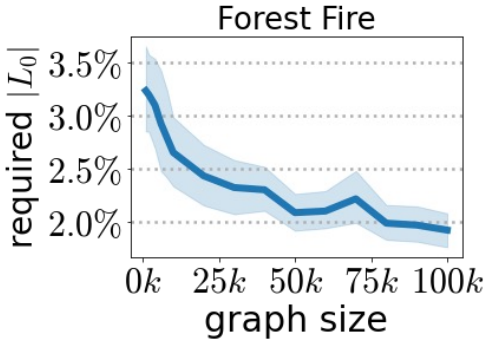
Why does it work?

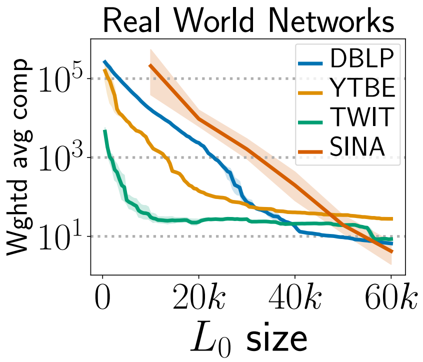
- Algorithm provably converges to uniformity:

THEOREM 3.1. *If our size estimation for $L_{\geq 2}$ is in $(1 \pm o(1))|L_{\geq 2}|$, and if the baseline reachability rs_0 used in our algorithm is the $o(1)$-percentile in the reachability distribution, then the output node distribution of* SAMPLE *is $o(1)$-close to uniform in total variation distance. Furthermore, the sampling probability of any node is at most $\frac{1+o(1)}{n}$.*

Why does it work?

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

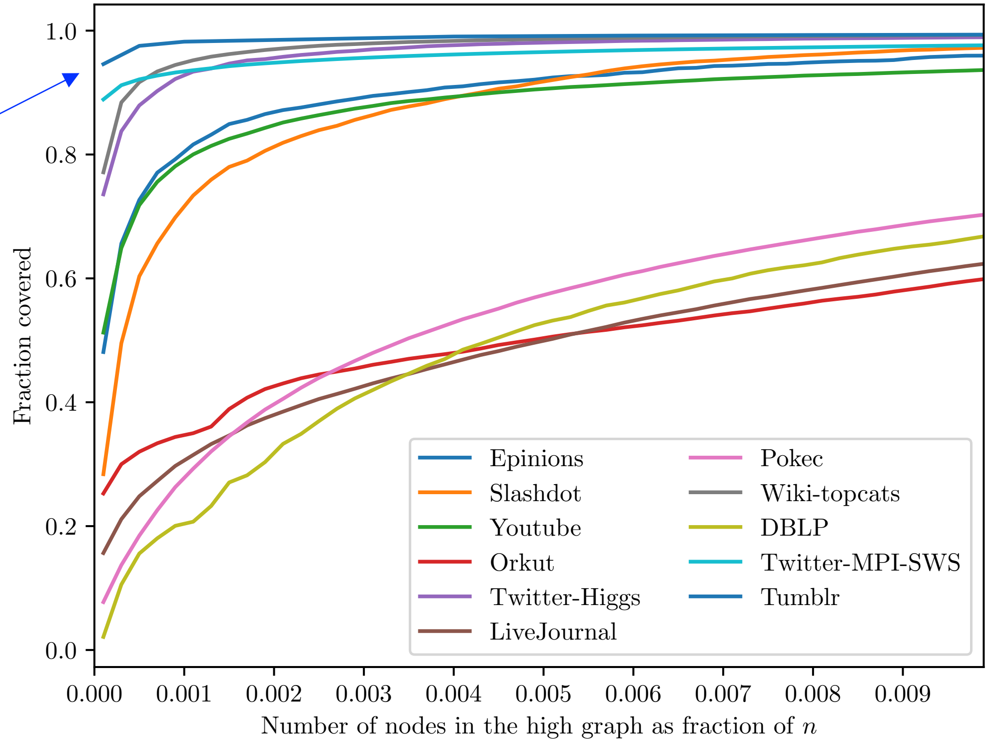




Why does it work?

- **Sublinear "almost domination"**: Most nodes with, say, (out-)degree ≥ 10 have a neighbor in top 0.1%-1% highest degrees.

Tumblr



Dataset	No. Nodes	No. Edges
Epinions [39]	76K	509K
Slashdot [28]	82K	948K
DBLP [48]	317K	1.05M
Twitter-Higgs [18]	457K	14.9M
Youtube [48]	1.1M	2.99M
Pokec [45]	1.6M	30.6M
Wiki-topcats [49]	1.8M	28.5M
Orkut [48]	3.1M	117M
LiveJournal [48]	4.8M	69M
Twitter-MPI-SWS [9]	**53M**	**2.0B**
Tumblr	**247M**	**14.5B**

Why does it work?

- (Weak) theoretical bounds on query complexity:

THEOREM 3.2. *The expected query complexity of sampling a single node using* SAMPLAYER *is* $O\left(c \cdot \left(\frac{1}{\alpha} + wd\right)\right)$.

Open Questions

- More explanations and applications for “sublinear almost domination”? [BLMPP’15, MSSK’13, NA’12]
- Efficient node sampling in the random walk query model? (e.g., [PS’21])
- Other practical algorithms based on core-periphery? [ASK’12, AIY’13, BK’19]
    - Also, better theoretical guarantees for our algorithm?
- Learning-augmented models for algorithms on large networks? (e.g., [CEILNRSWZ’22])

Thank you!