2016/09/8 VLDB 2016

Dynamic Influence Analysis in Evolving Networks

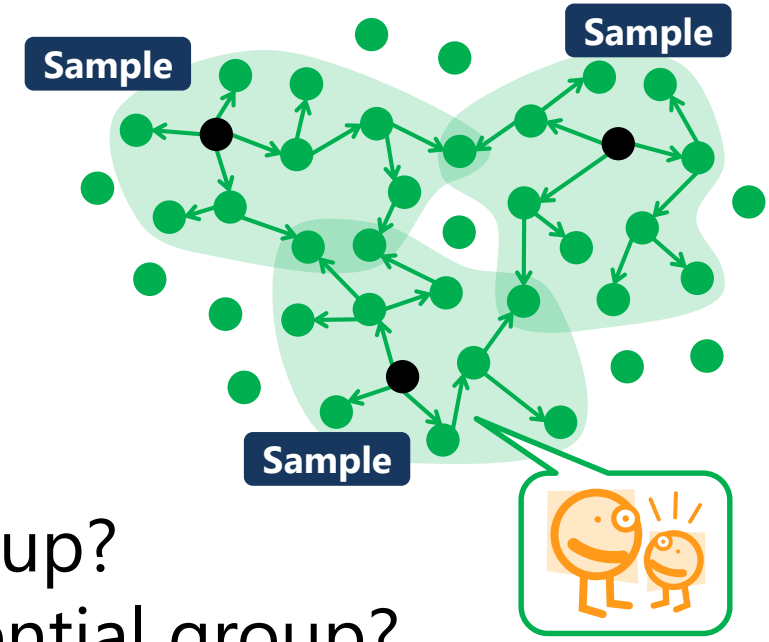
Naoto Ohsaka (UTokyo)
Takuya Akiba (PFN)
Yuichi Yoshida (NII & PFI)
Ken-ichi Kawarabayashi (NII)

Influence analysis in online social networks

Application: viral marketing

[Domingos-Richardson. *KDD'01*]

Product promotion through word-of-mouth effects



Q. How influential is a given group?

Q. How to select the most influential group?

Graph problems

Influence estimation

Influence maximization

[Kempe-Kleinberg-Tardos. *KDD'03*]

Existing algorithms for influence maximization (2003—2015)

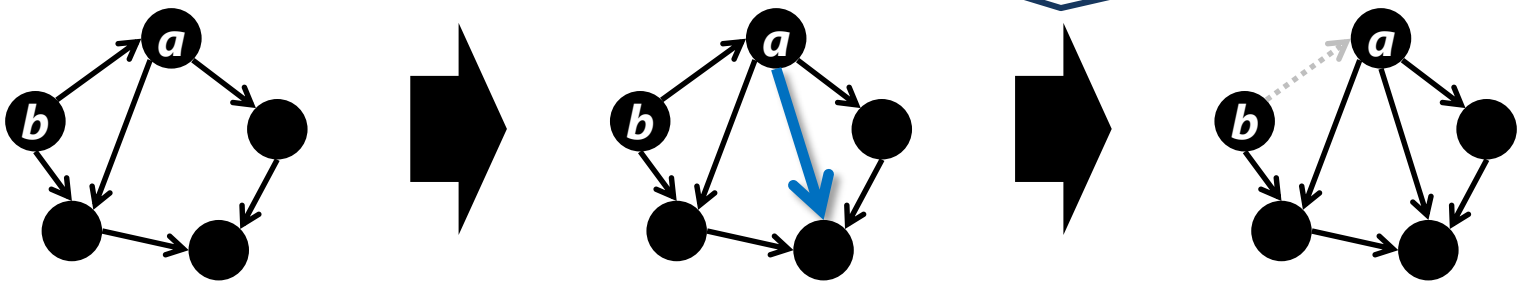
Strategy	Scalability	Accuracy
Simulation [Kempe-Kleinberg-Tardos. *KDD'03*] [Kimura-Saito-Nakano. *AAAI'07*] [Chen-Wang-Yang. *KDD'09*] ↙our previous work [O.-Akiba-Yoshida-Kawarabayashi. *AAAI'14*]	**<100M edges** Quadratic time	**Good** ≈ 63% approx.
Heuristics [Chen-Wang-Yang. *KDD'09*] [Chen-Wang-Wang. *KDD'10*] [Jung-Heo-Chen. *ICDM'12*]	100M – 1B edges	**Bad** No guarant.
Sketching [Borgs-Brautbar-Chayes-Lucier. *SODA'14*] [Tang-Xiao-Shi. *SIGMOD'14*] [Tang-Shi-Xiao. *SIGMOD'15*]	**>1B edges** Near-linear time	**Good** ≈ 63% approx.

These algorithms are **static**

Real-world social networks are **dynamic**

Social networks are **dynamic** and **evolving**

Accounts and friendships appear or disappear



Want to **track** influential vertices

Simply applying static methods ⇝ **>Linear time**

☹ Methods are almost **undeveloped**

[Zhuang-Sun-Tang-Zhang-Sun. *ICDM'13*]
Probing a small number of vertices

[Chen-Song-He-Xie. *SDM'15*] Edge operations only

Our contribution

Fully-dynamic indices

for influence analysis in evolving networks

①Indexing

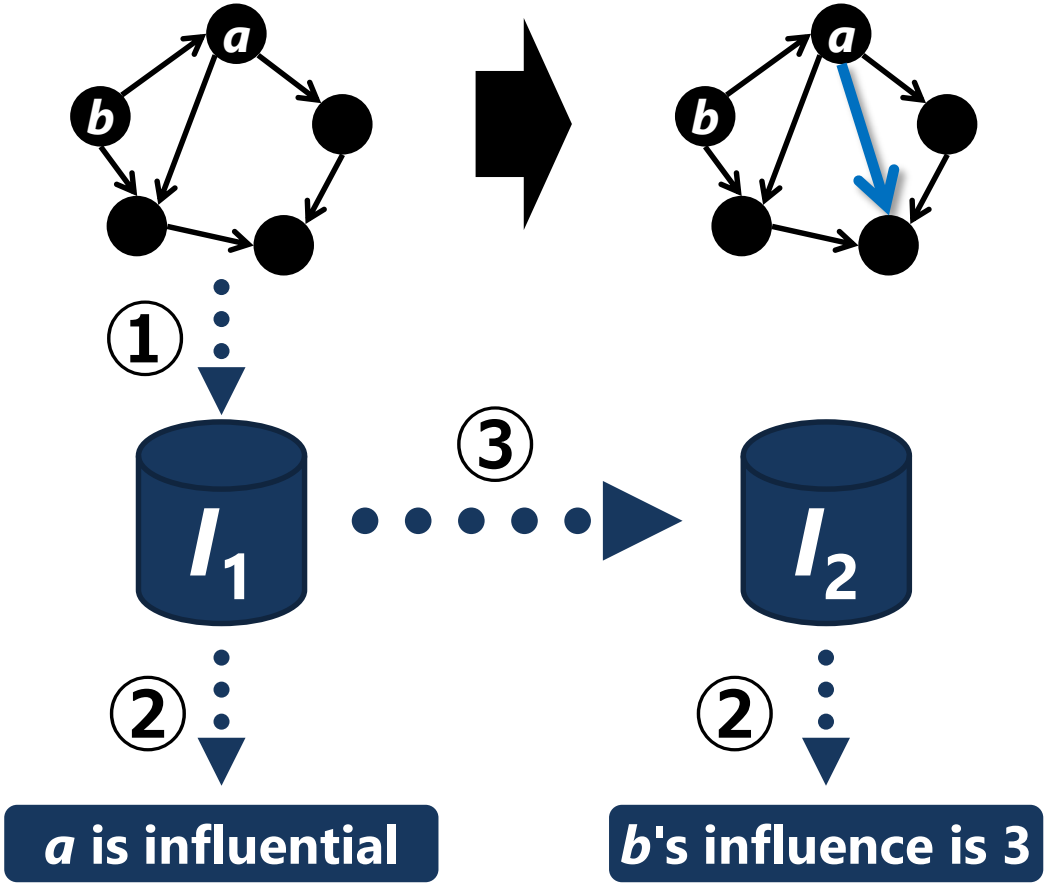
Almost-linear size

②Analysis query

Accuracy guarant.

③Index update

Any change



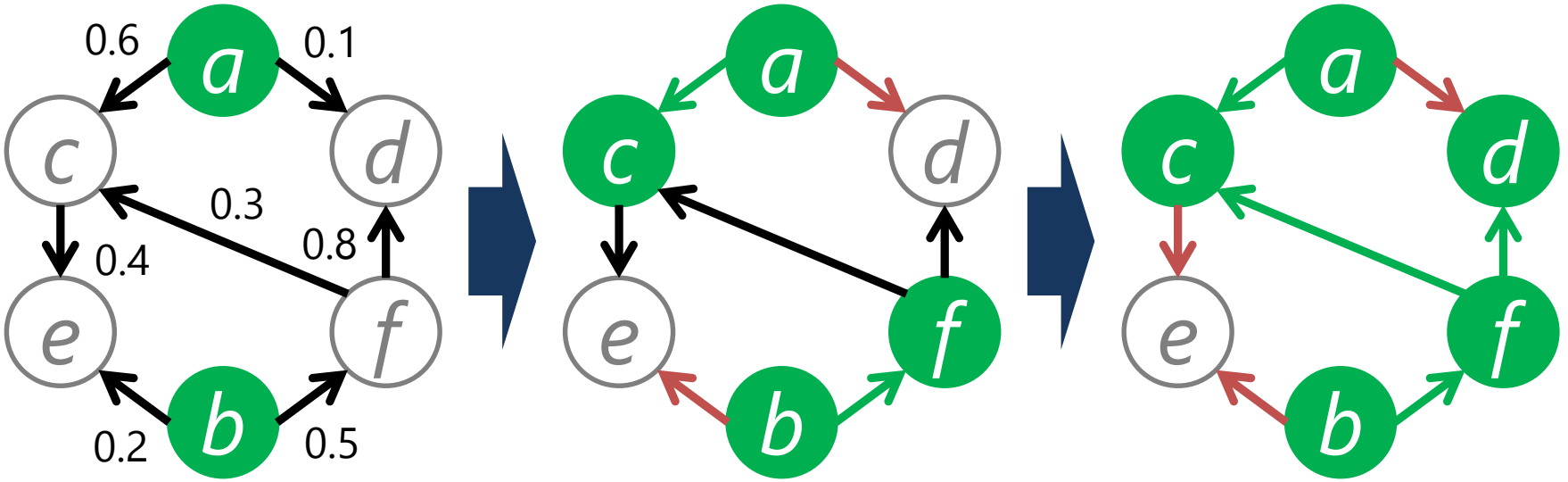
Diffusion model: Independent cascade

[Goldenberg-Libai-Muller. *Market. Lett.'01*]

Graph $G = (V, E, p)$ with edge probabilities
Seed set $S \subseteq V$

Initialize vertex's state
- **Active** if $\in S$
- **Inactive** if $\notin S$

Active u to **Inactive** v
- **Success** *w.p.* p_{uv}
- **Failure** *w.p.* $1 - p_{uv}$



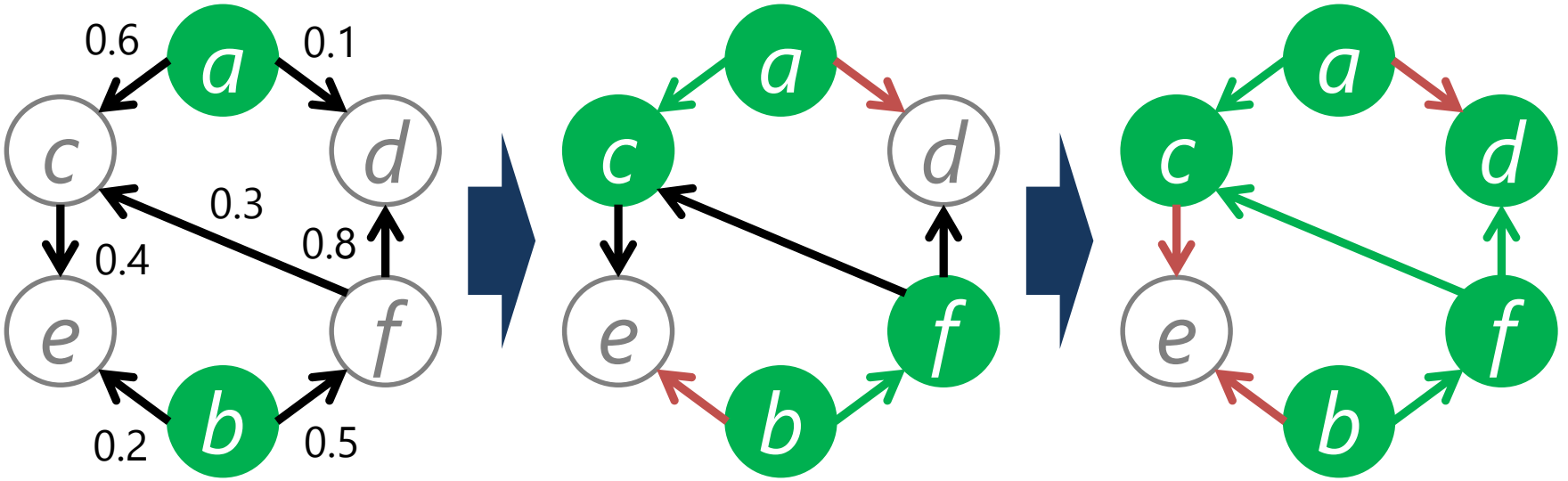
Diffusion model: Independent cascade

[Goldenberg-Libai-Muller. *Market. Lett.'01*]

Graph $G = (V, E, p)$ with edge probabilities
Seed set $S \subseteq V$

Influence spread

$$\sigma(S) := \mathbf{E}[\text{\# active vertices given } S]$$



Problem definition

Influence estimation

Input seed set S

Output $\sigma(S)$

#P-hard

[Chen-Wang-Wang. *KDD'10*]

Monte-Carlo is **good approx.**

Influence maximization

[Kempe-Kleinberg-Tardos. *KDD'03*]

Input integer k

Output $\operatorname{argmax}_{S:|S|=k} \sigma(S)$

NP-hard [Kempe+'03]

Greedy strategy is

$\left(1-\mathrm{e}^{-1}\right) \approx$ **63%-approx.**

[Nemhauser-Wolsey-Fisher. *Math. Program.*'78]

$\sigma(\cdot)$ is **submodular** [Kempe+'03]

$$\forall X \subseteq Y, v \notin Y$$
$$\sigma(X+v)-\sigma(X) \geq \sigma(Y+v)-\sigma(Y)$$

Key: fast & accurate estimation of $\sigma(\cdot)$

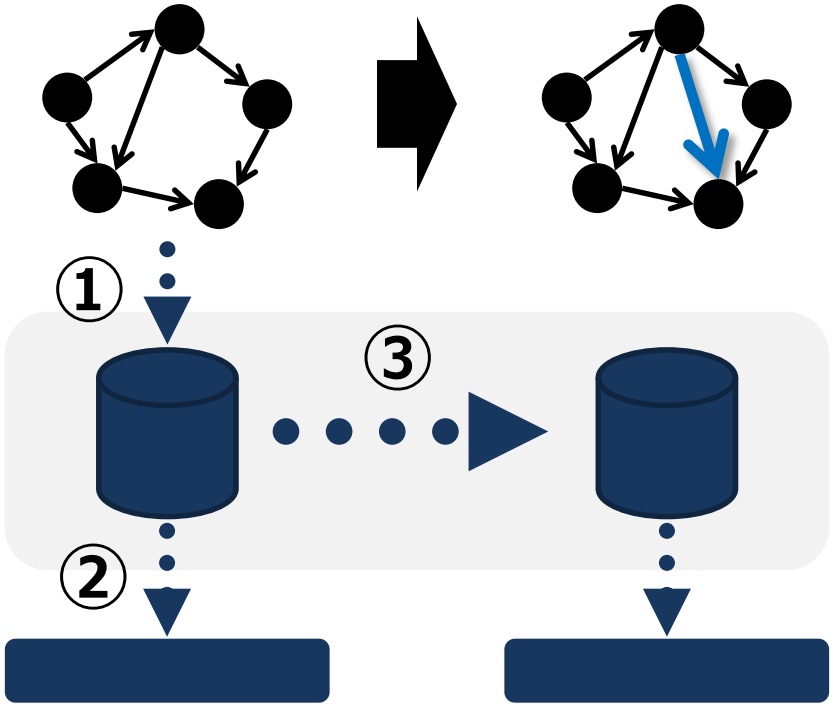
Proposed method

What we need:

① Indexing algorithm

② Influence query algorithms

③ Update algorithms



Index construction

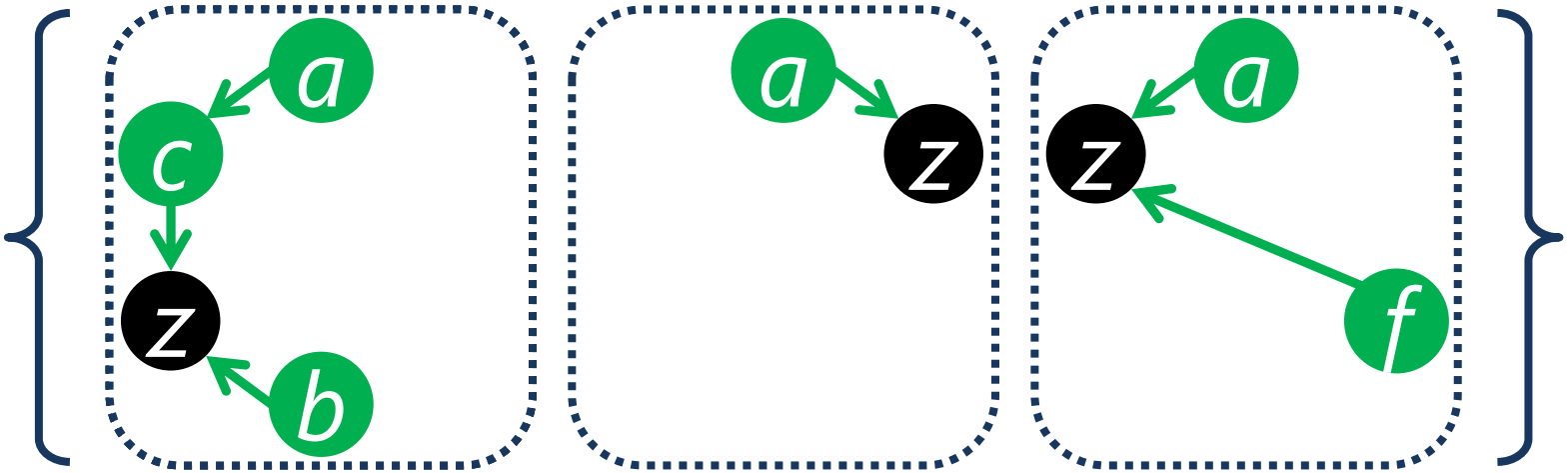
Redesign of *Reverse Influence Sampling*

[Borgs-Brautbar-Chayes-Lucier. *SODA'14*]

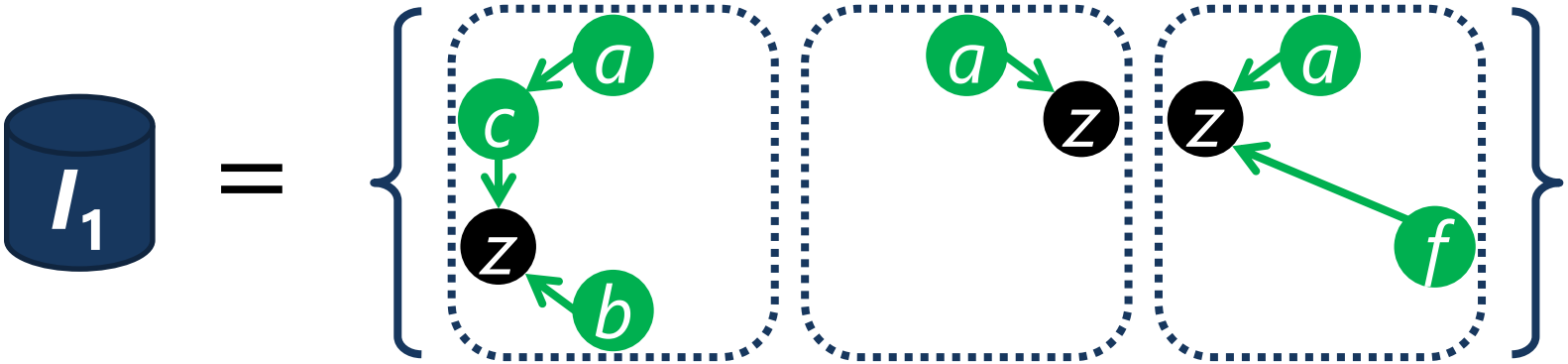
Single sketch construction：

- Randomly select a **target** vertex ●z
- Conduct a **reverse simulation** from ●z
- **Sketch** = (● 's) ∪ (↗ 's)

Index size = #● + ∑●'s in-deg = $\Theta(\epsilon^{-3}(|V| + |E|)\log|V|)$



Property of our index



Vertices frequently appearing in sketches are influential

$$\sigma(S) \propto \mathbf{E}[\#\text{ sketches intersecting } S]$$

[Borgs-Brautbar-Chayes-Lucier. *SODA'14*]

Query algorithms for influence analysis

Based on *Reverse Influence Sampling*
[Borgs-Brautbar-Chayes-Lucier. *SODA'14*]

Influence estimation query (seed set S)

- Computing the size of union on sketches
- $\boldsymbol{\sigma(S) \pm \epsilon|V|}$ *w.h.p.* (Thm. 5.9)

Influence maximization query (solution size k)

- Solving maximum coverage on sketches
- $\boldsymbol{\left(1 - \mathrm{e}^{-1} - \epsilon\right)}$**-approx.** *w.h.p.* (Thm. 5.10)

We further introduce speed-up techniques (see our paper)

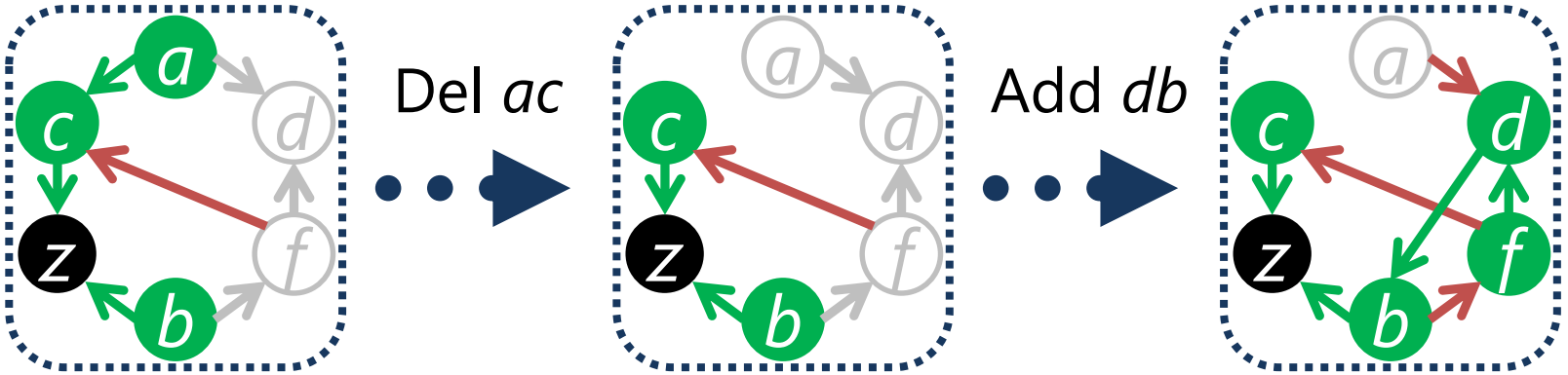
Overview of index update algorithms

3 edge operations
- Edge addition
- Edge deletion
- Probability change

2 vertex operations
- Vertex addition
- Vertex deletion

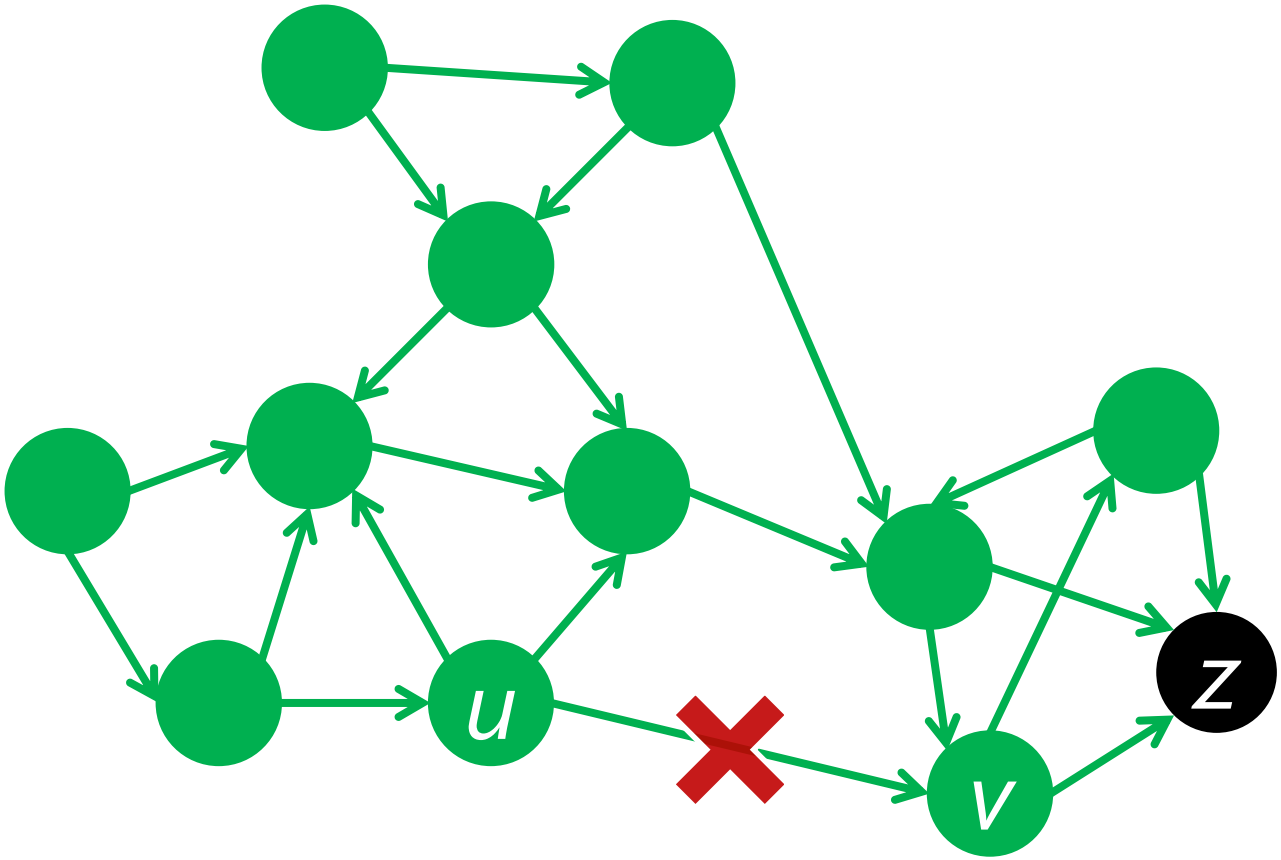
Independently update *each* sketch so that "all ● can reach (z) by passing through ↗"

⇝ Non-degeneracy = **No need reindexing** (Thm. 5.8)



Edge deletion example (1)

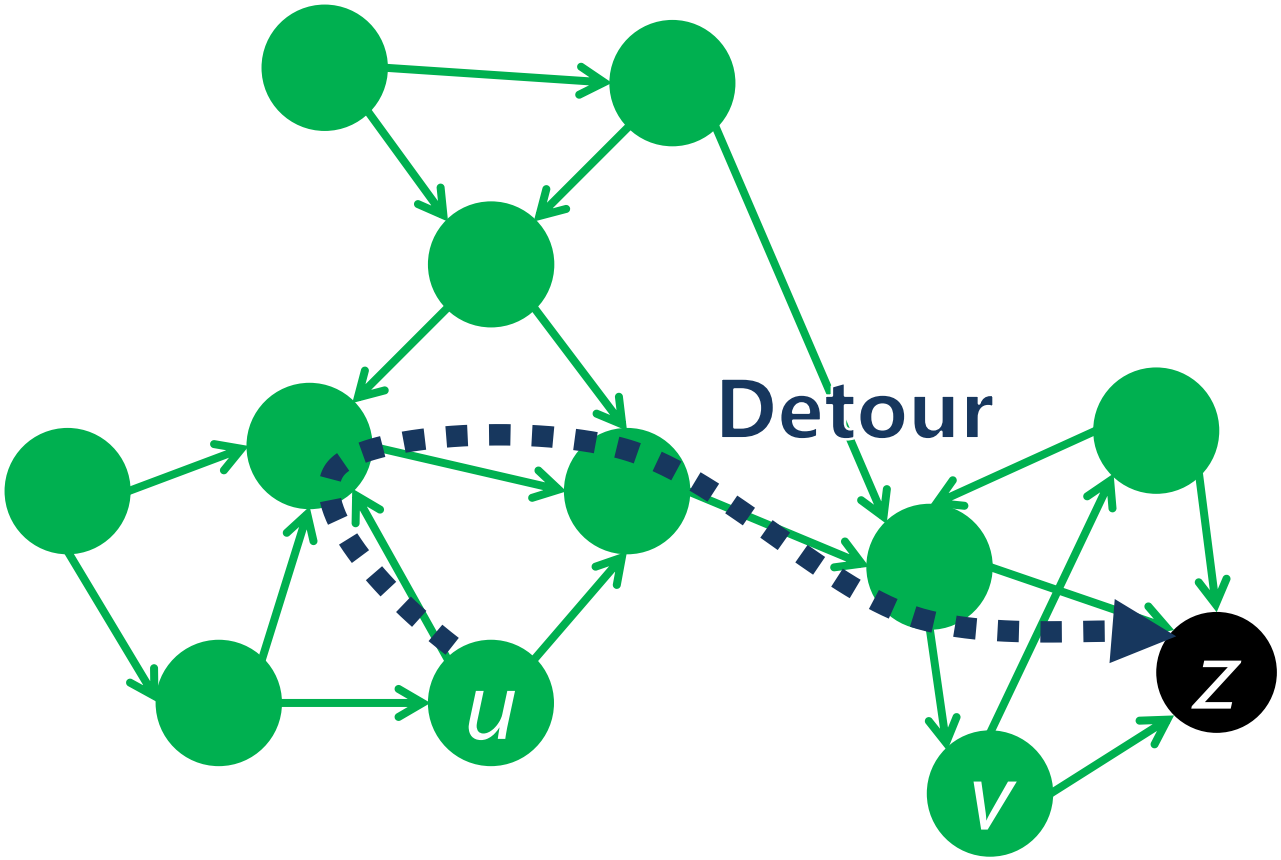
Q. “Vertices that can reach z” **decrease**?



Edge deletion example (1)

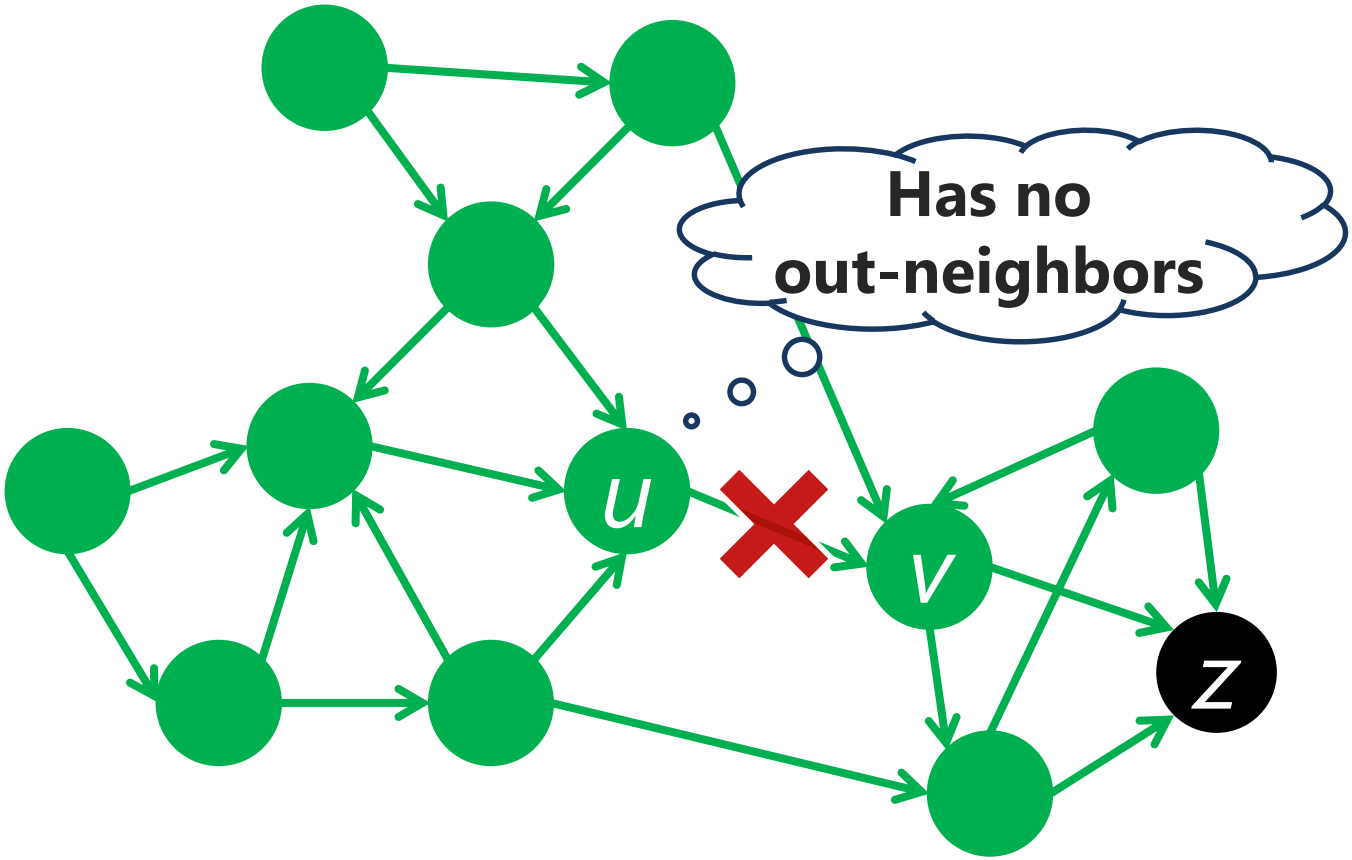
Q. “Vertices that can reach z” **decrease**?

A. NO



Edge deletion example (2)

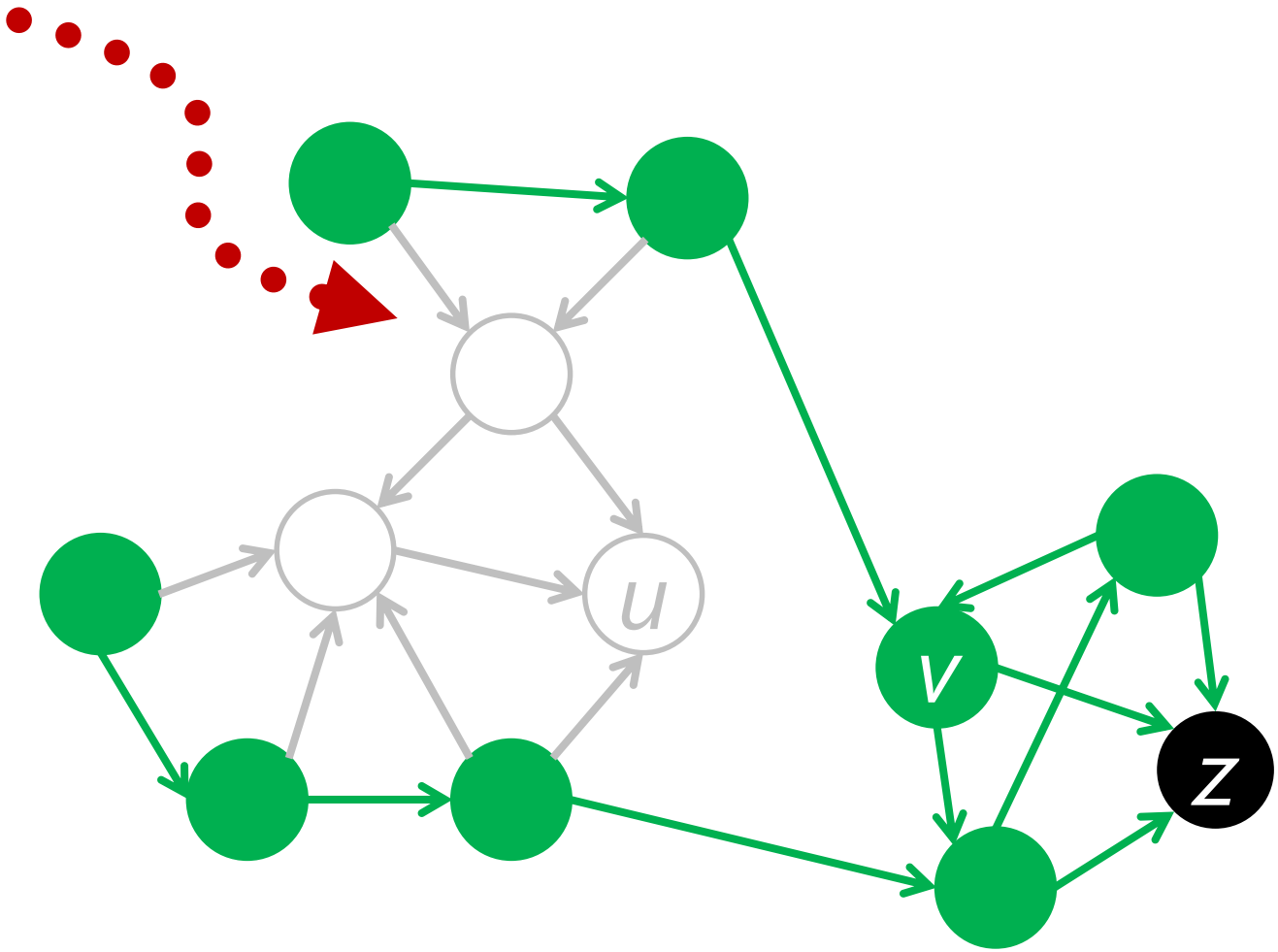
Q. “Vertices that can reach z” **decrease**?



Edge deletion example (2)

Q. “Vertices that can reach **z**” **decrease**?

A. YES



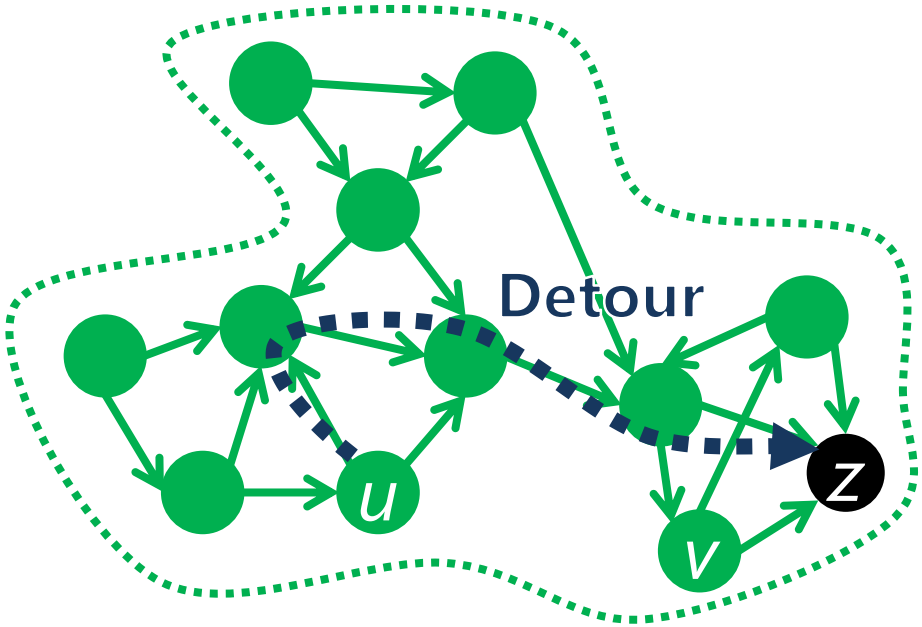
How about naive update for edge deletion?

Perform a reverse BFS from z

To detect ○, need to scan **all** ●

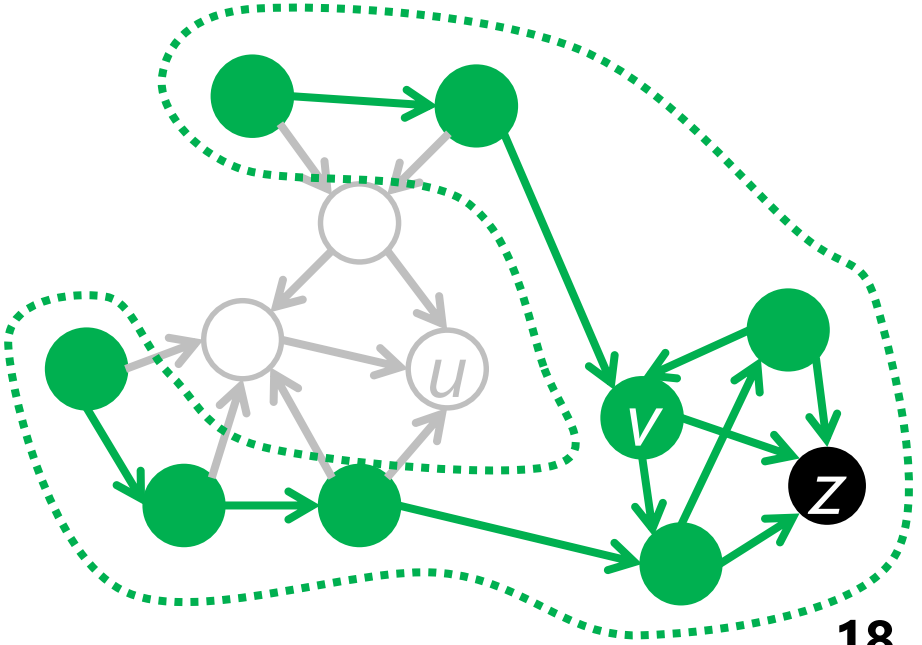
Challenge 1:

Detour detection



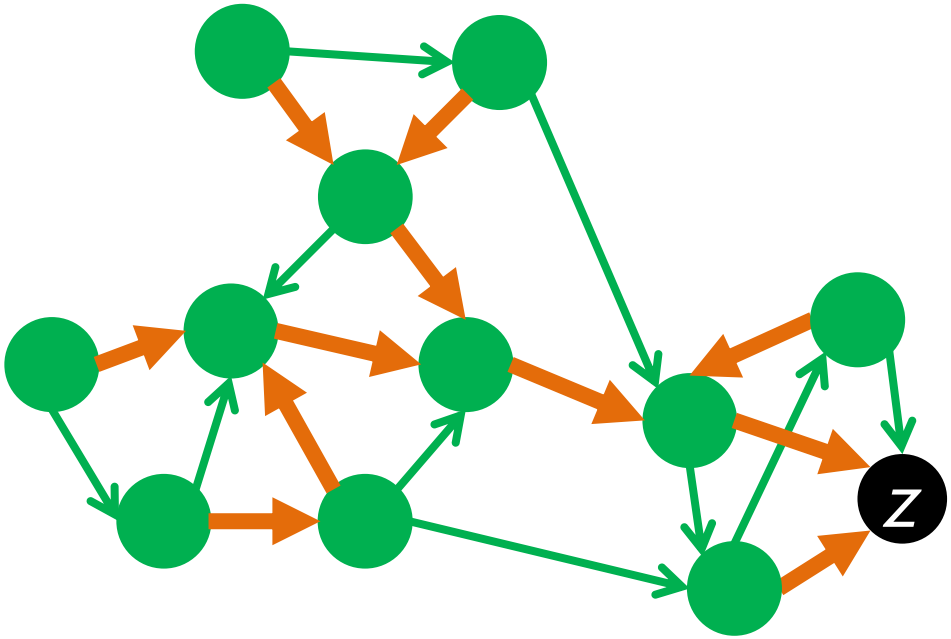
Challenge 2:

Efficient reverse BFSes



Fast edge deletion: Using **reachability tree**

A **directed subtree** of a sketch rooted on z



☺ **Detour existence check**

☺ **Limiting the search range**

* Can be used for vertex deletion

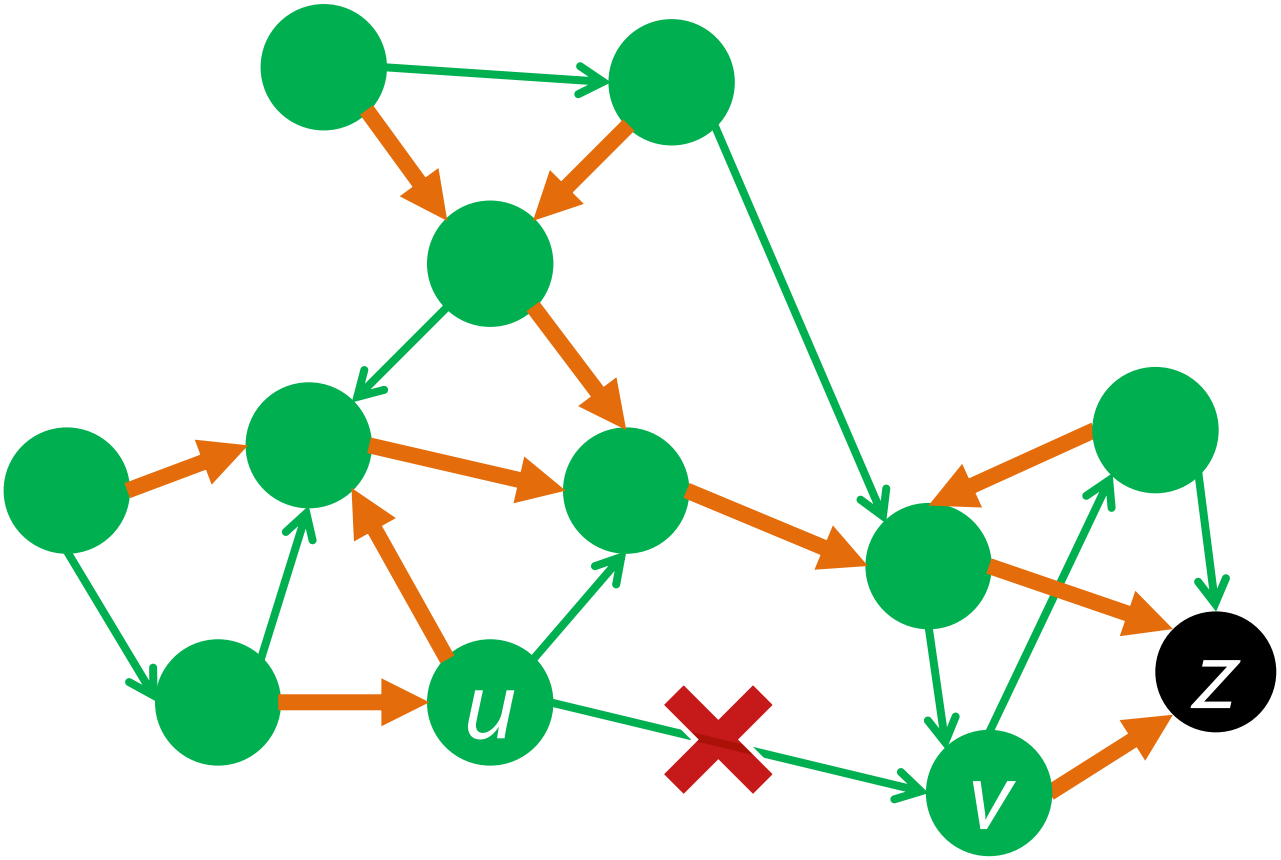
Fast edge deletion:

Detour existence check

uv ∉ **tree** ⇒ ∃ a detour from *u* to *z*

* not vice versa

10% deletions are pruned



Fast edge deletion:

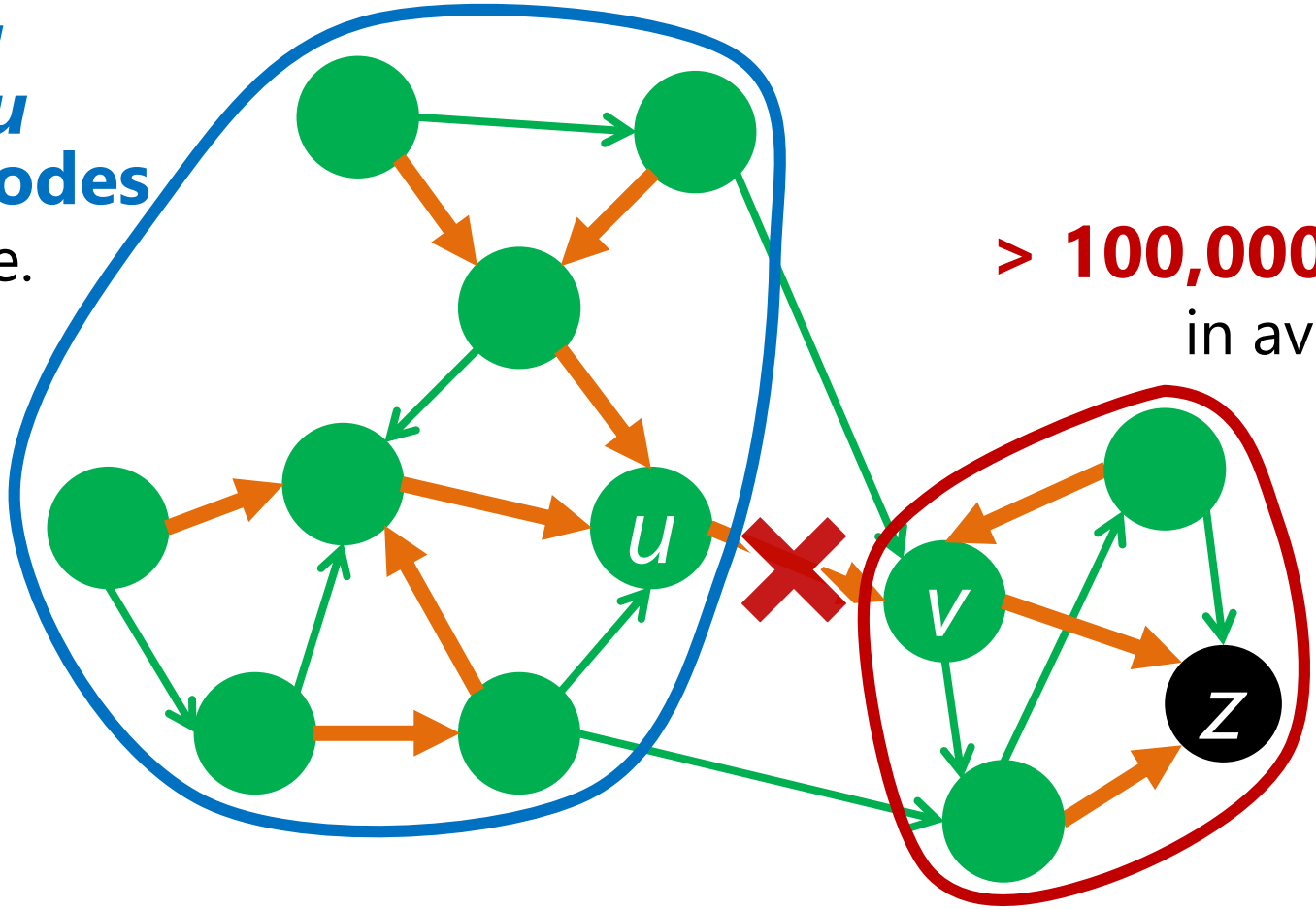
Limiting the search range

Verify a subtree T_u rooted on u & update **tree**

T_u

a few nodes

in ave.



> 100,000 nodes

in ave.

Fast edge deletion:

Limiting the search range

Verify a subtree T_u rooted on u & update **tree**

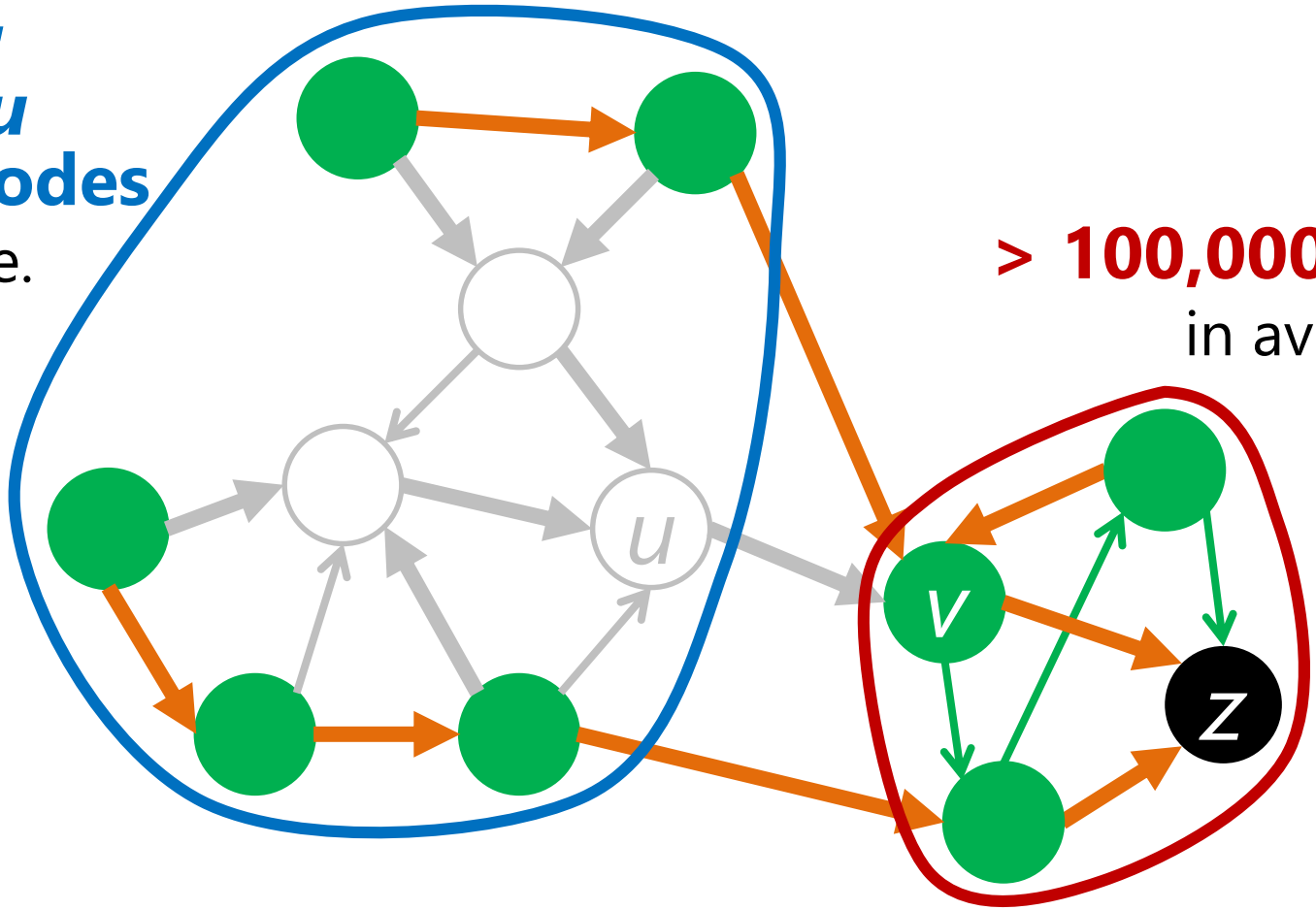
T_u

a few nodes

in ave.

> 100,000 nodes

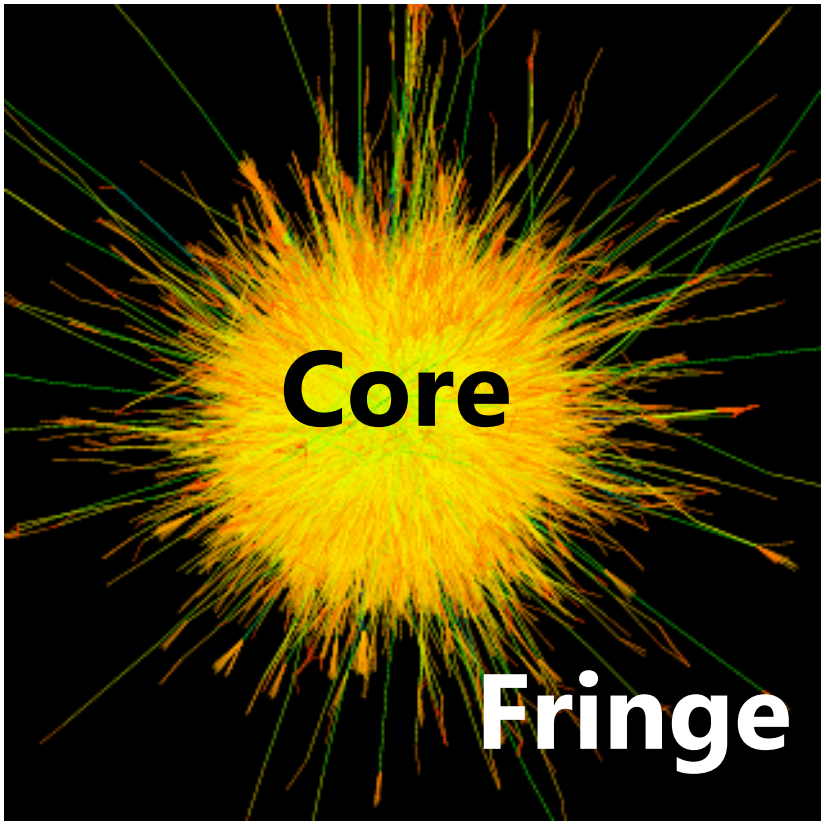
in ave.



Why our techniques are effective?

Core-fringe structure

[Leskovec-Lang-Dasgupta-Mahoney. *WWW'08*]
[Maehara-Akiba-Iwata-Kawarabayashi. *PVLDB'14*]



http://www.cise.ufl.edu/research/sparse/matrices/SNAP/soc-Epinions1.html

Core is **dense**
- Many detours
- 1st tech. works well

Fringe is **tree-like**
- T_u is small
- 2nd tech. works well

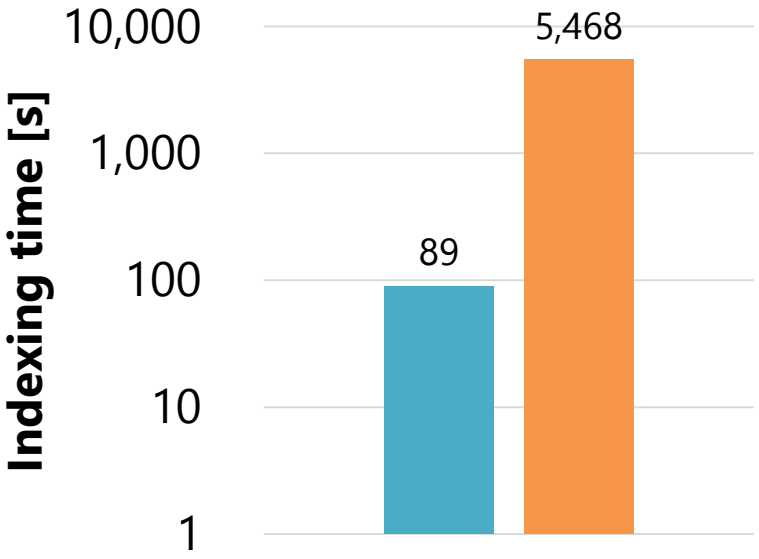
Experiments

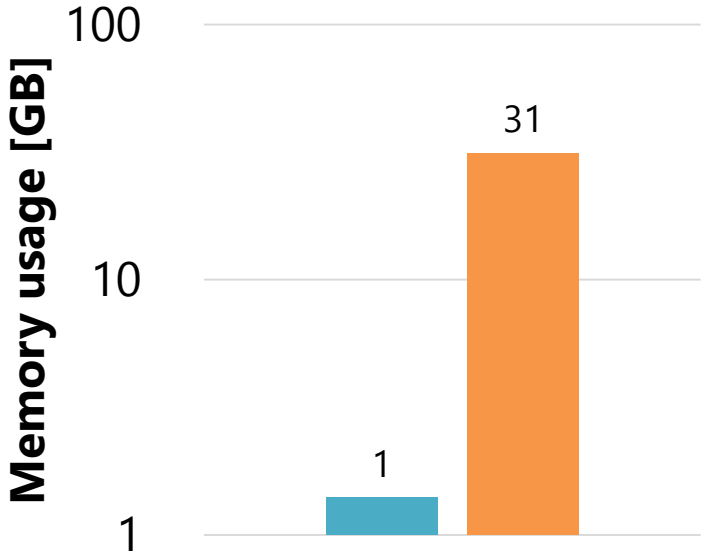
Efficiency of
- Index construction
- Index update
- Influence estimation query
- Influence maximization query

- Dataset：Koblenz Network Collection http://konect.uni-koblenz.de/
  with timestamps at which edge was created
- Machine: Intel Xeon E5-2690 2.90GHz CPU + 256GB RAM
- Compiler: g++v4.6.3 (-O2)
- Index size $= 32(|V| + |E|)\log|V|$
- Edge prob. = randomly chosen from {0.1, 0.01, 0.001}

Index construction

Network	$\|V\|$	$\|E\|$
Epinions	130K	840K
Flickr	2,303K	33,140K



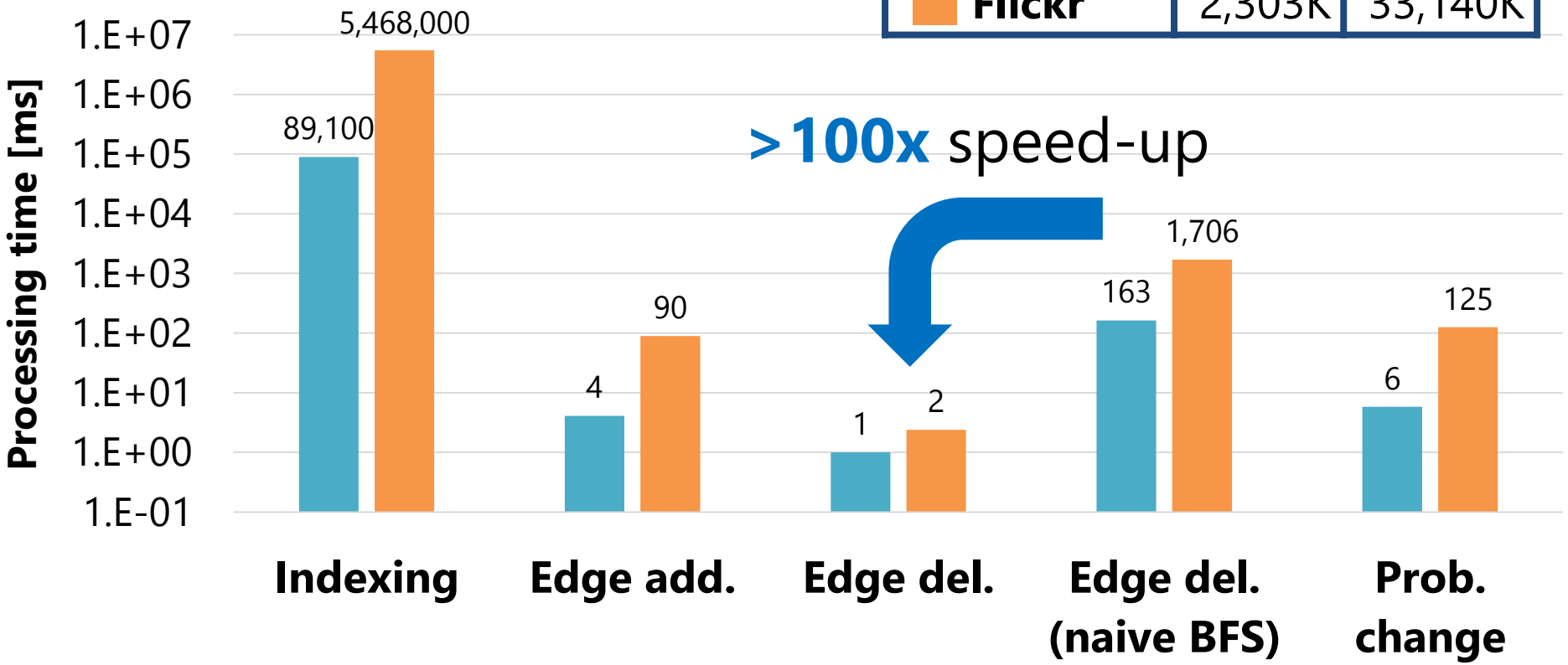


- Can handle graphs with **tens of millions** of edges
  - Indexing is required **just once**

Experiments

Index update time: Edge operations

Network	$\|V\|$	$\|E\|$
Epinions	130K	840K
Flickr	2,303K	33,140K

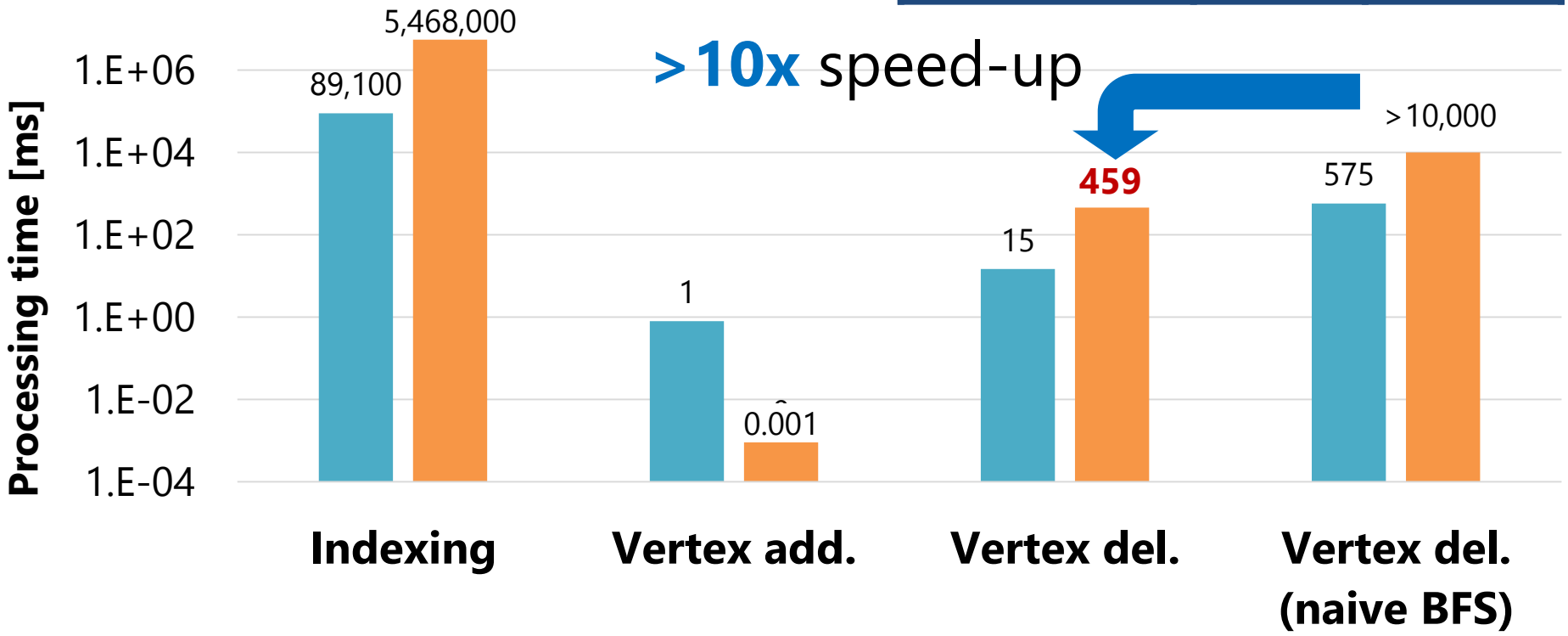


- (Update time) ≪ (Indexing time)

Experiments

Index update time: Vertex operations

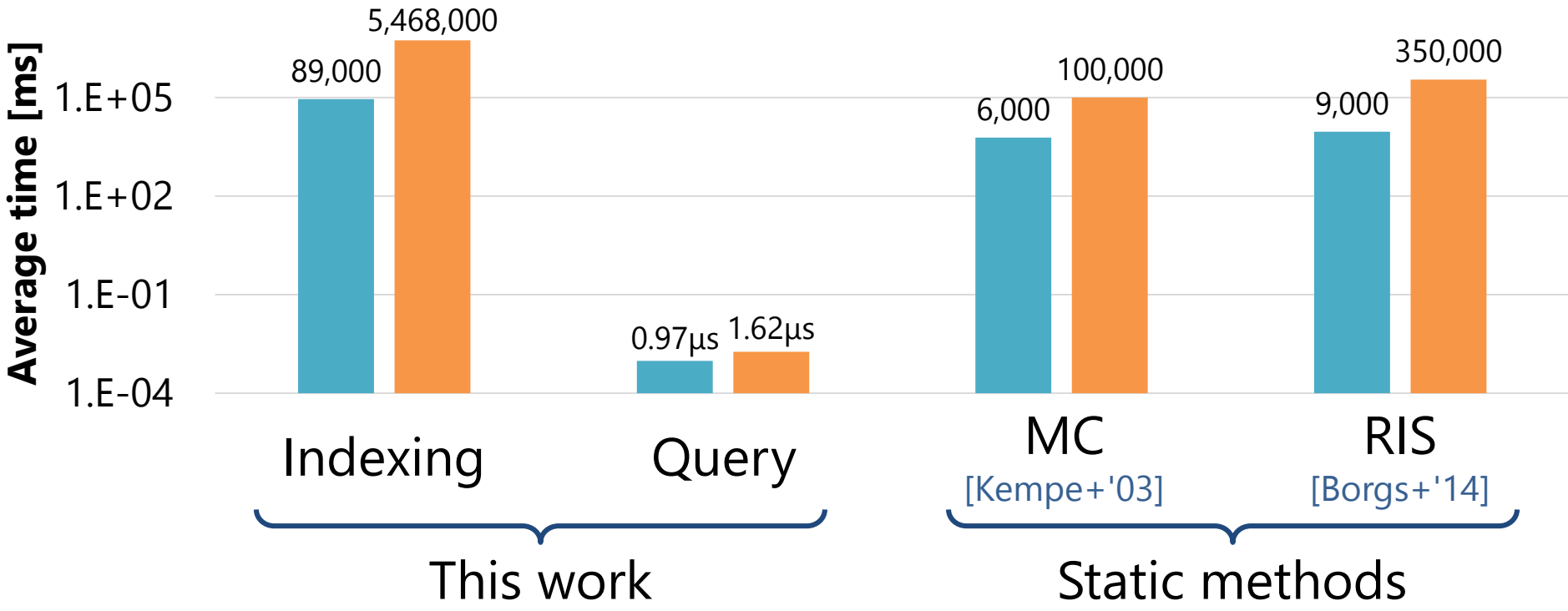
Network	$\|V\|$	$\|E\|$
Epinions	130K	840K
Flickr	2,303K	33,140K



- (Update time) ≪ (Indexing time)
- **Vertex del.** causes **a number of edge dels.**

Experiments

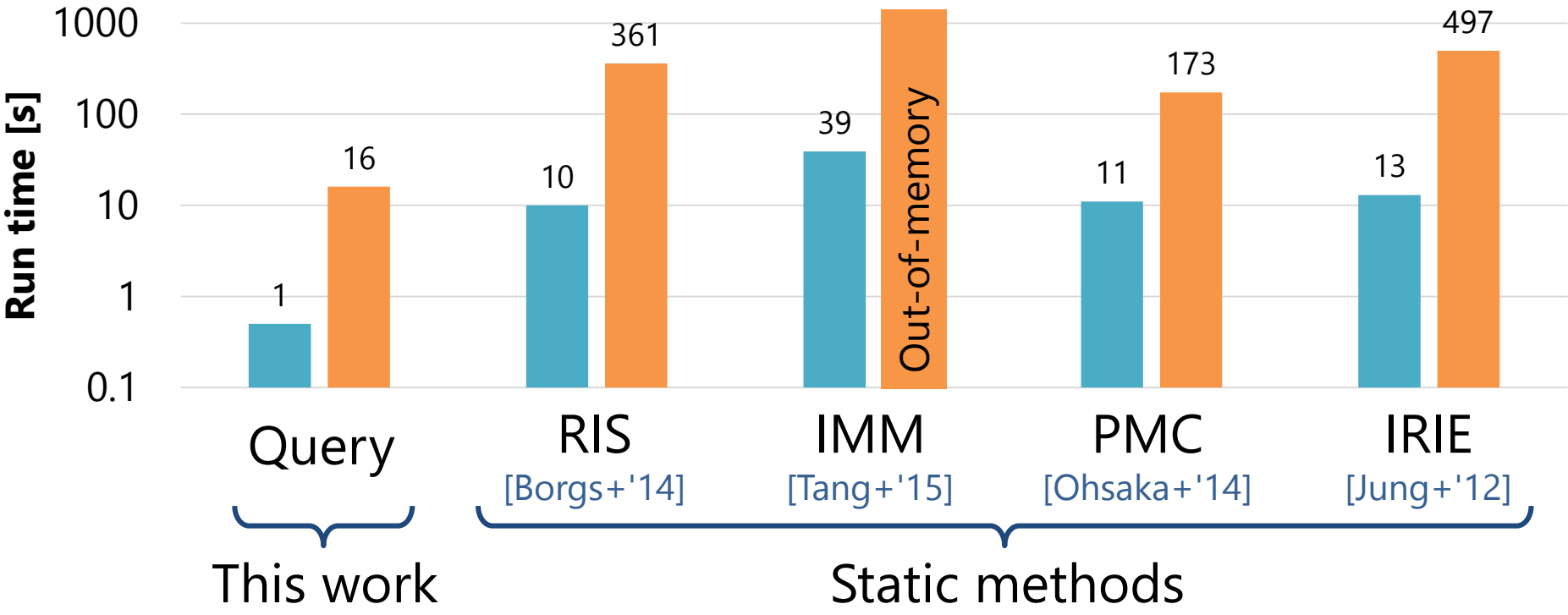
Influence estimation queries:
Time for estimating the influence of a vertex



- Tracking **1M nodes/sec**
  Just perform **table lookup**

Network	$\|V\|$	$\|E\|$
Epinions	130K	840K
Flickr	2,303K	33,140K

Influence maximization queries:
Time for selecting a seed set of size 100



- **>10 times faster**
  Solve max. coverage

Network	$\|V\|$	$\|E\|$
Epinions	130K	840K
Flickr	2,303K	33,140K

Conclusion

Proposed fully-dynamic indices
for influence analysis in evolving networks

① Indexing graphs w/ >10M edges in a few hours
② Reflect any graph change in 1 sec.
③ Fast influence analysis queries

Future directions

- More space saving for billion-scale graphs
- Fast influence maximization query
  Maximum Coverage in online setting