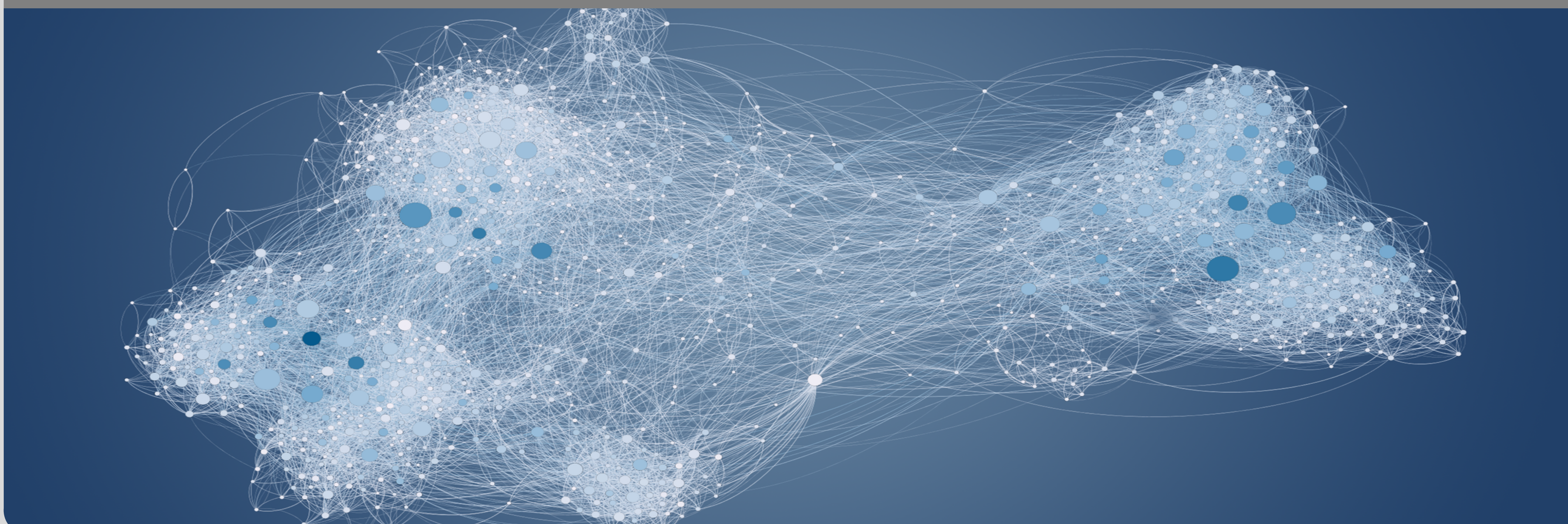
Karlsruhe Institute of Technology

On Betweenness Centrality Problems in Dynamic Graphs

Elisabetta Bergamini, Henning Meyerhenke

SIAM Conference on Computational Science and Engineering · February 27 - March 3, 2017



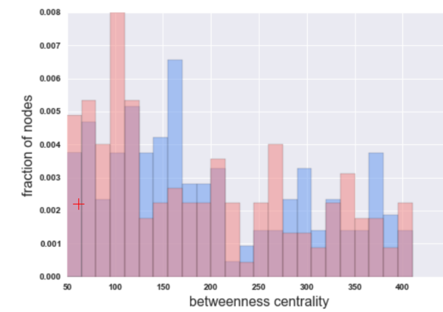
Network Analysis

Graphs/networks can be used to model **relations** or **interactions**:

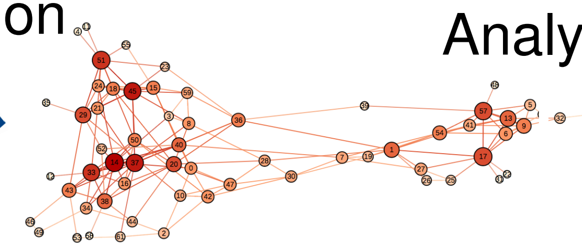
Examples:

- Social networks
- Protein interactions
- Transportation networks
- Climate correlations
- ...

Phenomenon



Analysis



Network representation

Goal:

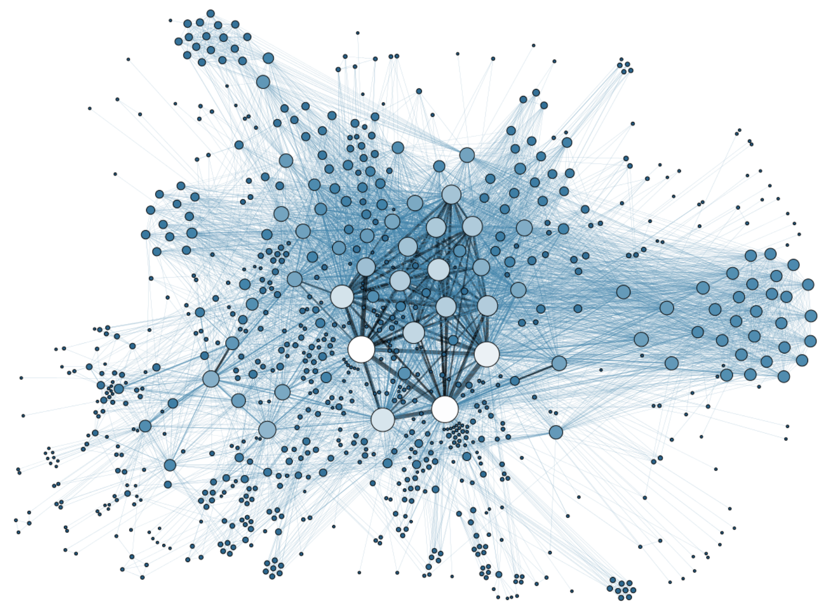
Discover **useful information** by analyzing the **network structure**

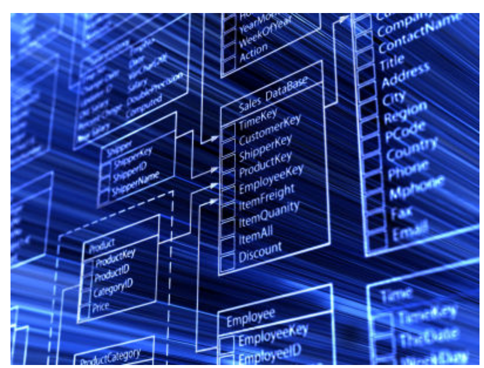
Features:

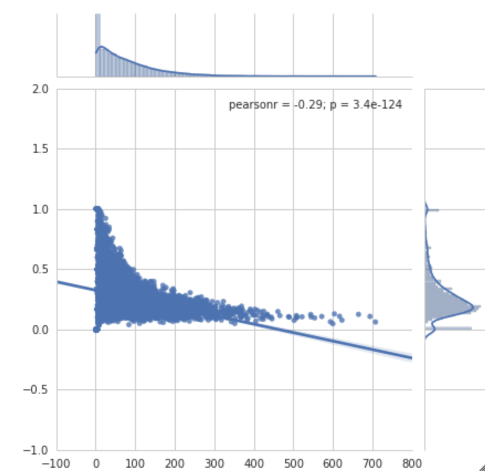
- Small diameter
- Skewed degree distribution

Targeting:

- Large networks
- Dynamic networks







Introduction | Betweenness centrality

- BC: participation of nodes in the **shortest paths** of the network
- Nodes with **high betweenness** $\rightarrow$ lie in **many shortest paths** between pairs of nodes
- Given $G = (V, E)$ and $v \in V$:

$$b_C(v) = \sum_{\substack{s,t \in V \\ s \neq v \neq t}} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

[geoidin.wordpress.com]

where:

- σ_{st} = number of s.p. between s and t
- $\sigma_{st}(v)$ = number of s.p. between s and t that go through v

- **Betweenness Approximation in Dynamic Networks**

  [Bergamini, Meyerhenke and Staudt, ALENEX 2015]

  [Bergamini and Meyerhenke, ESA 2015]

  [Bergamini and Meyerhenke, Internet Mathematics]

- **Single-node Betweenness Update**

  [Bergamini, Crescenzi, D'Angelo, Meyerhenke, Severini, Velaj. Under review.]

Karlsruhe Institute of Technology

Exact solution

- Brandes's algorithm: $\Theta(|V||E| + |V|^2 \log |V|)$ [Brandes, JMSo 2001]

Approximation algorithms

- Extrapolate betweenness from randomly sampled shortest paths

  [Geisberger et al., ALENEX 2008], [Bader et al., WAW 2007],

  [Riondato and Kornaropoulos, DAMI], [Riondato and Upfal, KDD 2016]...

Exact dynamic algorithms

- Several approaches [Lee et al., WWW 2012], [Green et al., SocialCom 2012] ...
- None of them is asymptotically faster than recomputation

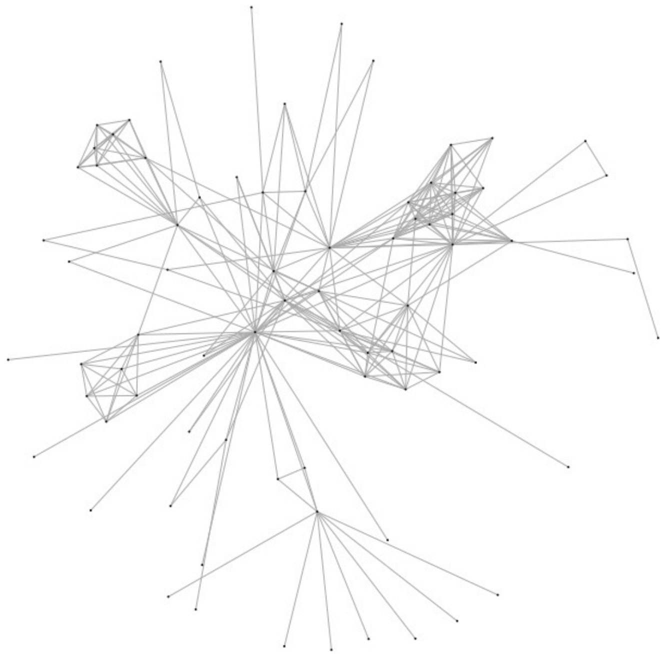
Our Contribution

- **Dynamic approximation** algorithms

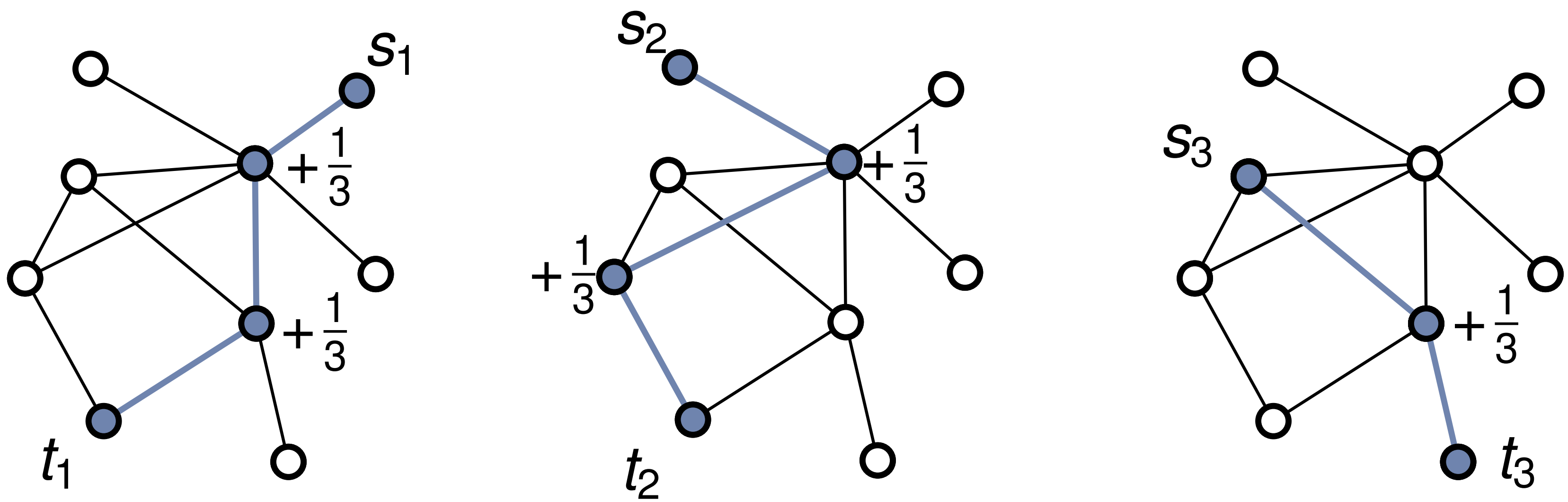
  [Bergamini et al., ALENEX 2015], [Bergamini and Meyerhenke, ESA 2015],

  [Bergamini and Meyerhenke, Internet Mathematics]

Static betweenness approximation



- A set of r **shortest paths** between vertex pairs (s_i, t_i) $\quad i = 1, .., r$ is **sampled**
- $\tilde{c}_B(v)$: **fraction** of sampled paths that **go through** v

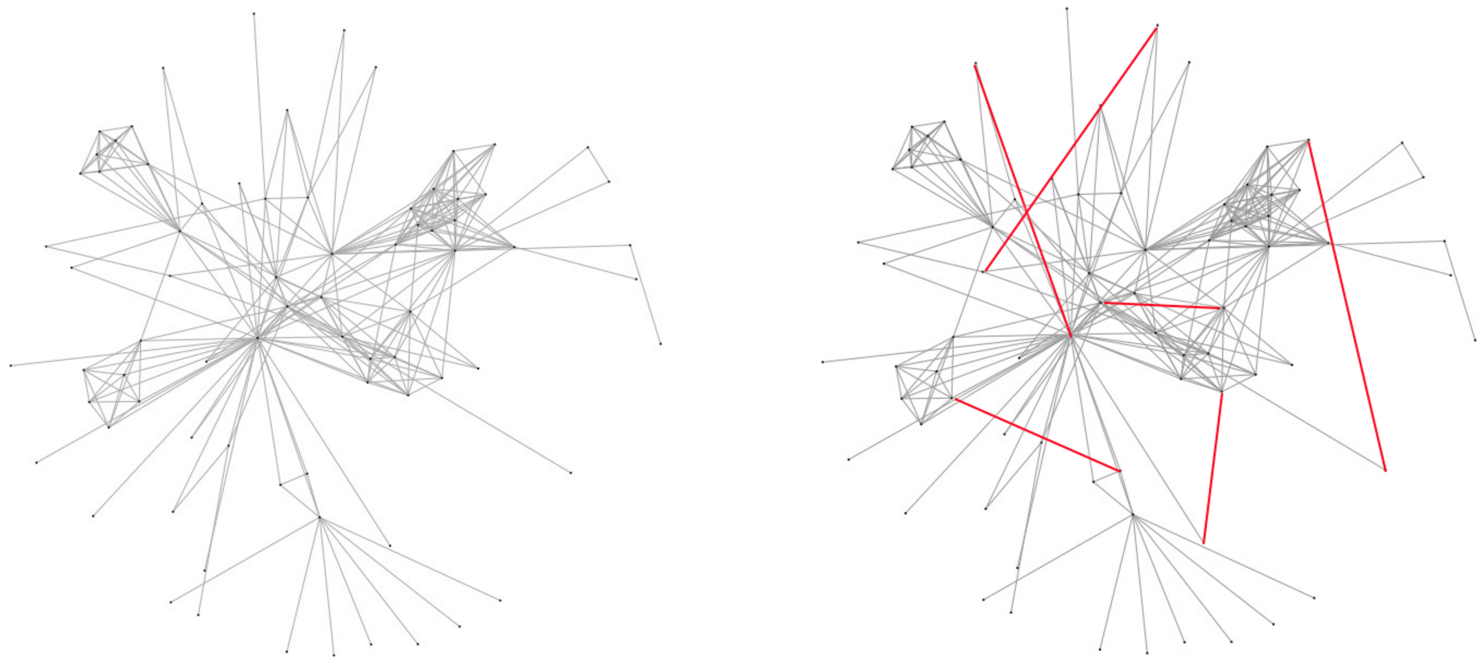


- **Maximum error** guarantee:

$$\left|c_B(v) - \tilde{c}_B(v)\right| < \epsilon \qquad \forall v \in V$$

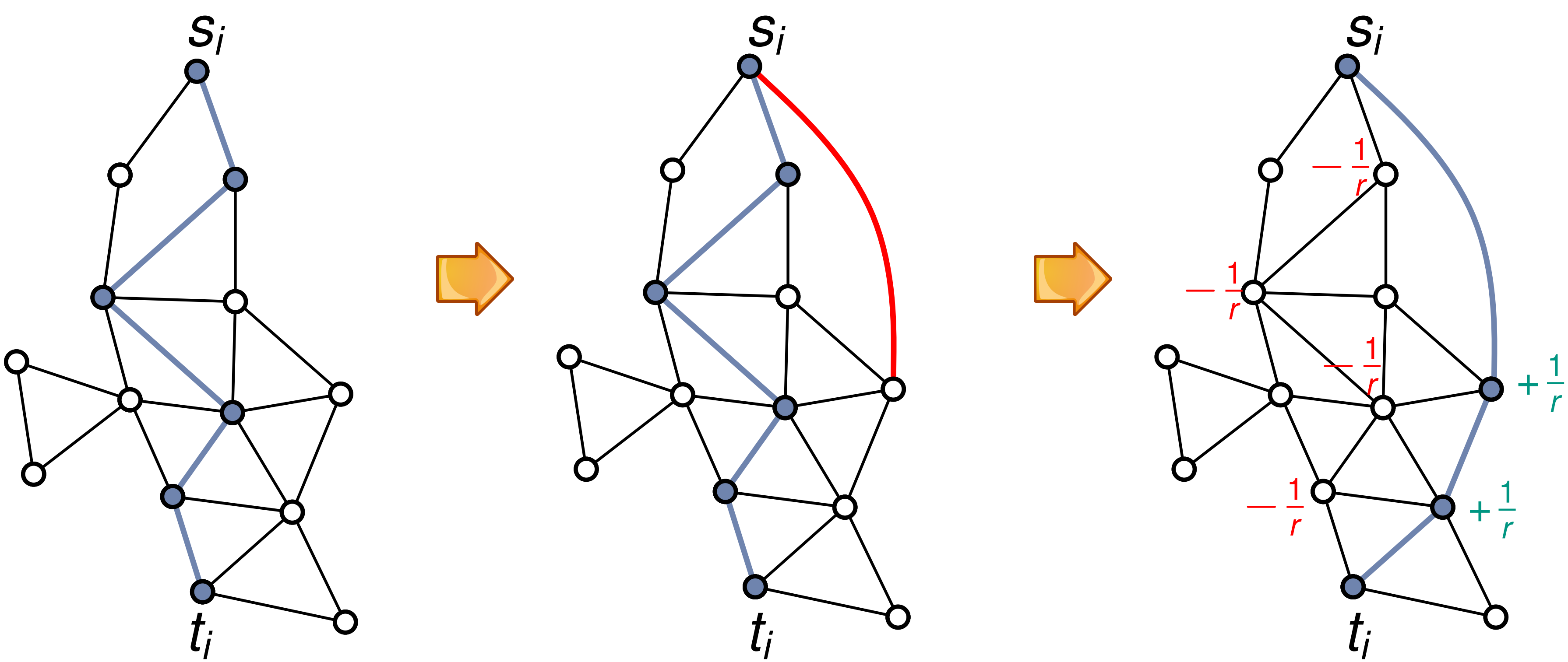
with probability at least δ

Updating betweenness after edge updates



Basic idea:

- we keep track of the sampled shortest paths and **replace** them when necessary

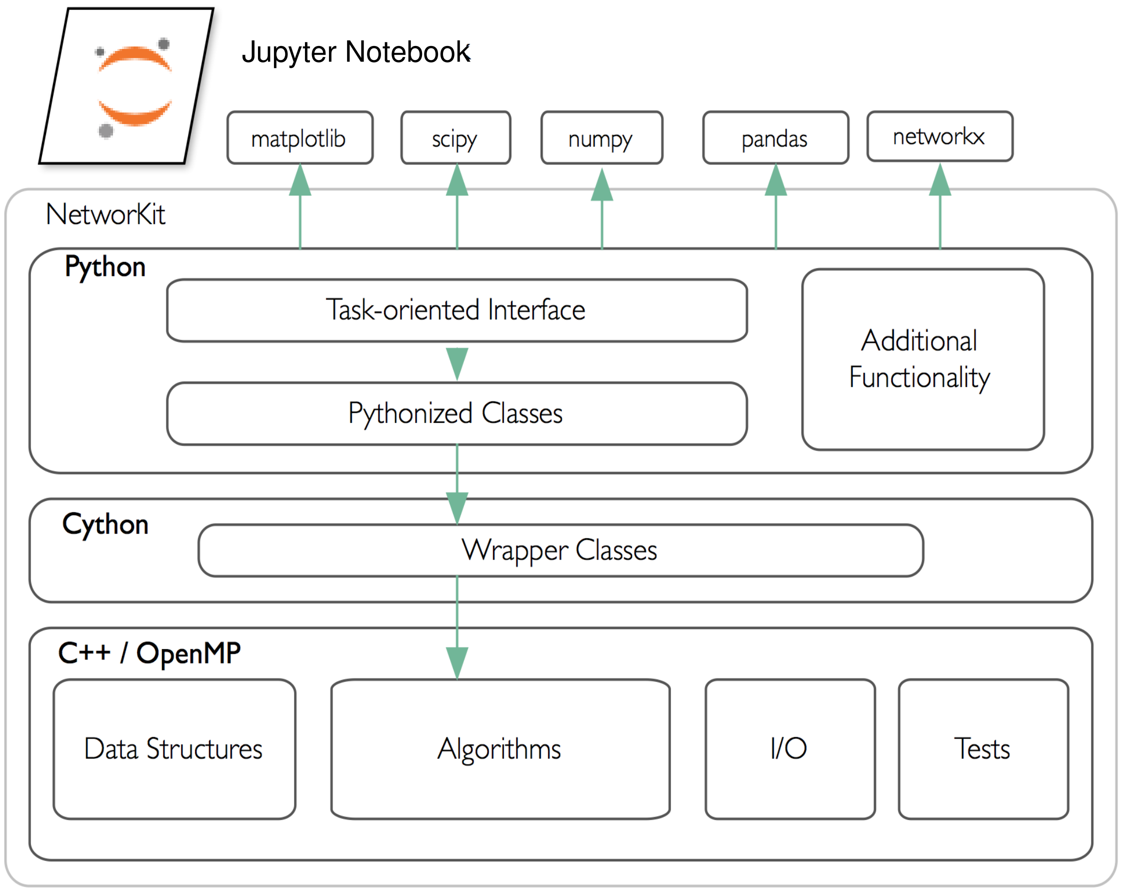


- to ensure the maximum error guarantee, in some cases we **sample new paths**

Experiments

- We implemented our algorithms in **NetworKit**:

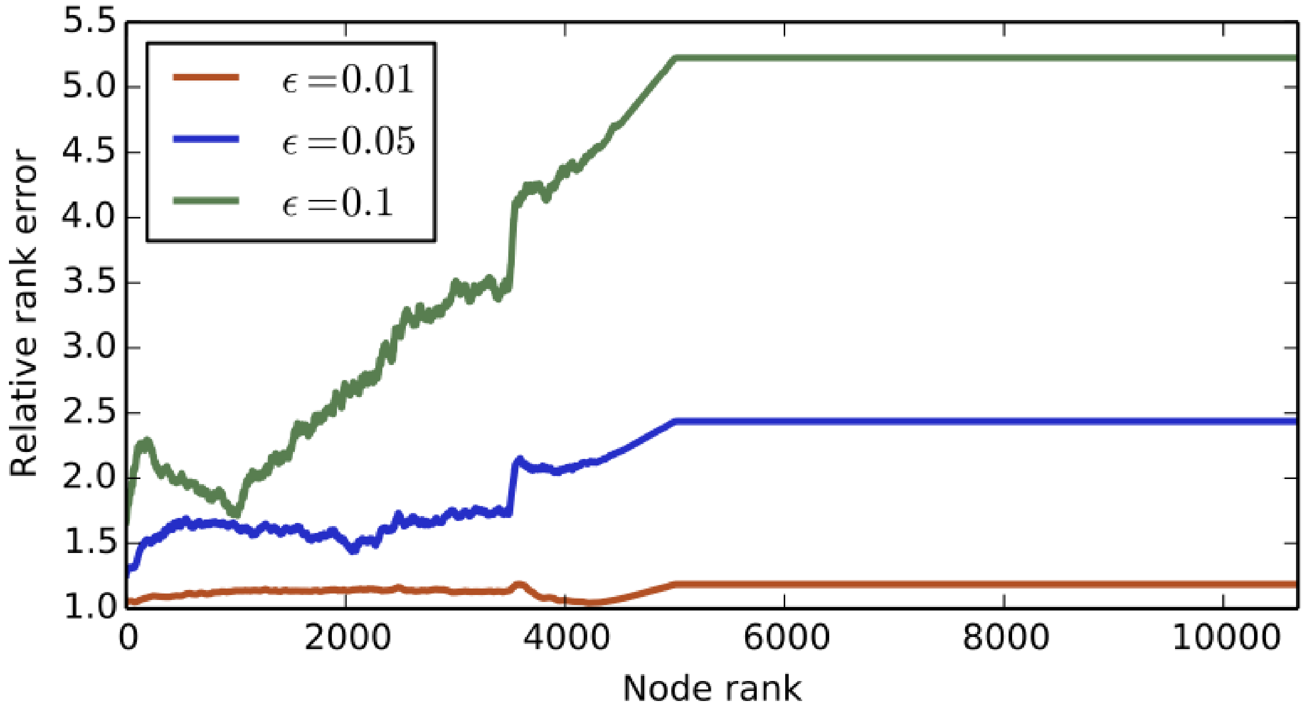
➡ **tool suite for scalable network analysis**
  - parallel algorithms
  - approximation algorithms
- **features** include ...
  - community detection
  - centrality measures
  - graph generators
- **free software**
  - Python package with C++ backend
  - under continuous development
  - download from **http://networkit.iti.kit.edu**



Results | Accuracy

Measured errors

- absolute errors **always lower** than the theoretical guarantees
- average errors **orders of magnitude** smaller



Relative rank errors for `PGPgiantcompo`

Rank error

$$r(v) = \frac{\text{pos}(v)_{\text{exact}}}{\text{pos}(v)_{\text{approx}}}$$

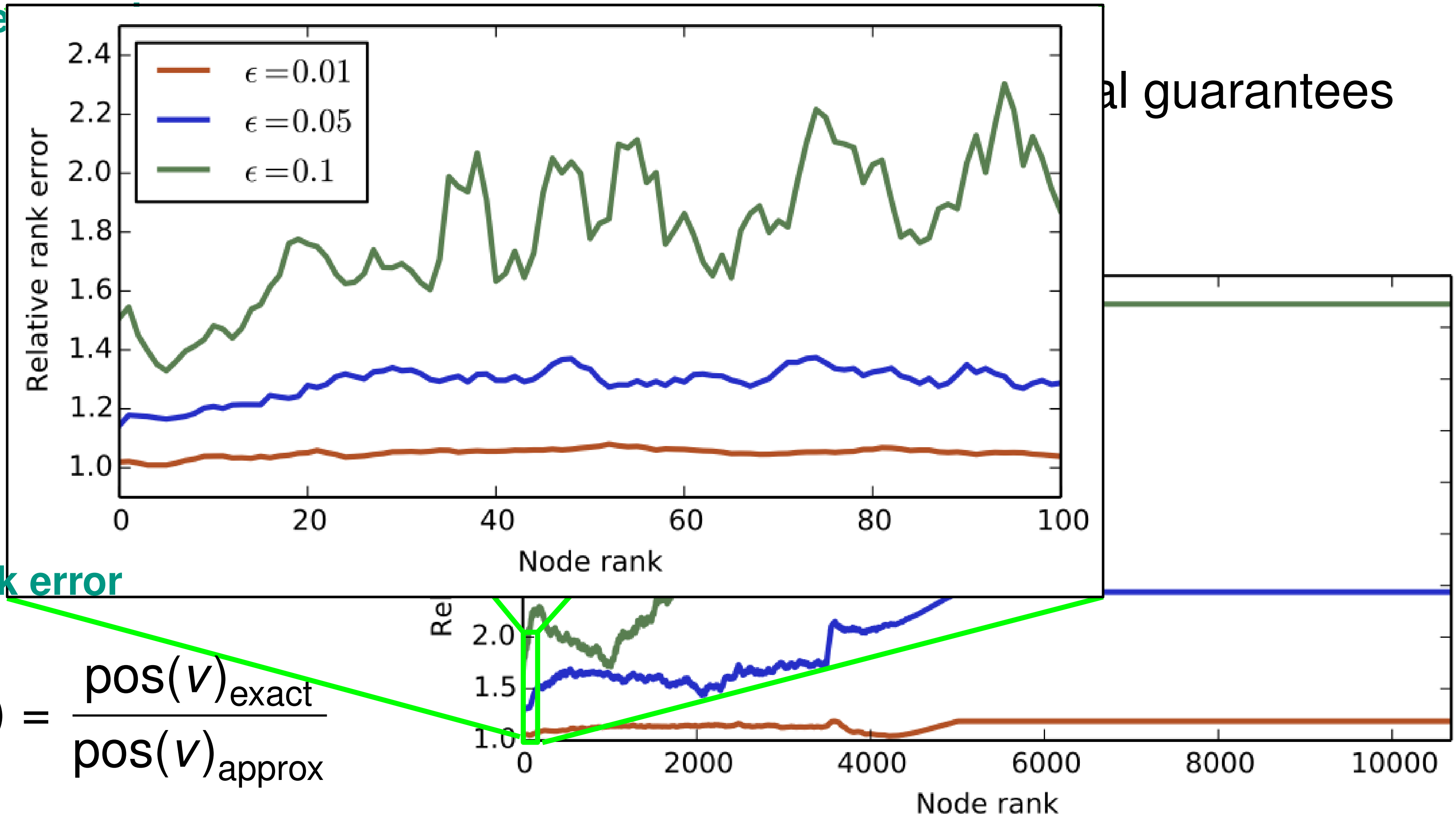
$$\text{err}_{\text{rank}}(v) = \max\{r(v), \frac{1}{r(v)}\}$$

R

very low rank error for nodes with high betweenness

Me

-
- al guarantees



Rank error

$$r(v) = \frac{\mathrm{pos}(v)_{\mathrm{exact}}}{\mathrm{pos}(v)_{\mathrm{approx}}}$$

$$\mathrm{err}_{\mathrm{rank}}(v) = \max\{r(v), \frac{1}{r(v)}\}$$

Relative rank errors for `PGPgiantcompo`

Running times

Dataset

- real dynamic networks, ranging from $\approx$ **85 K** to $\approx$ **36 M** edges
- type: communication, friendship, coauthorship, hyperlink

	Edges	Time DA [s]		Time RK [s]
Graph		$\lvert\beta\rvert = 1$	$\lvert\beta\rvert = 1024$	
`repliesDigg`	85,155	0.078	1.028	5.75
`emailSlashdot`	116,573	0.043	1.055	9.93
`emailLinux`	159,996	0.049	1.412	5.18
`facebookPosts`	183,412	0.023	1.416	13.04
`emailEnron`	297,456	0.368	1.279	24.11
`facebookFriends`	817,035	0.447	1.946	39.25
`arXivCitations`	3,148,447	0.038	0.186	80.71
`englishWikipedia`	36,532,531	1.078	6.735	3818.20

Results | Running times

Dataset

- real dynamic networks, ranging from $\approx$ **85 K** to $\approx$ **36 M** edges
- type: communication, friendship, coauthorship, hyperlink

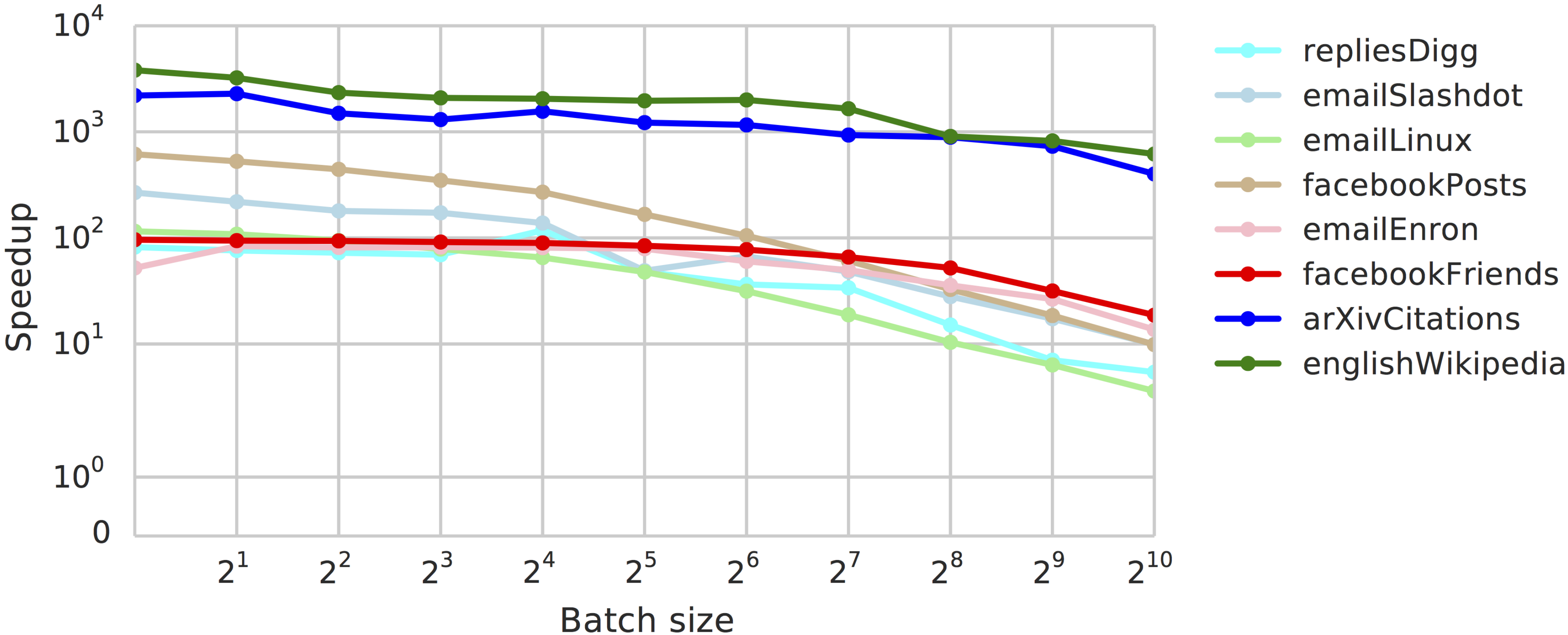
recomputation with dynamic algorithm never takes more than few seconds

		[s]		**Time RK** [s]
Graph		$\lvert\beta\rvert = 1$	$\lvert\beta\rvert = 1024$	
`repliesDigg`	85,155	0.078	1.028	5.75
`emailSlashdot`	116,573	0.043	1.055	9.93
`emailLinux`	159,996	0.049	1.412	5.18
`facebookPosts`	183,412	0.023	1.416	13.04
`emailEnron`	297,456	0.368	1.279	24.11
`facebookFriends`	817,035	0.447	1.946	39.25
`arXivCitations`	3,148,447	0.038	0.186	80.71
`englishWikipedia`	36,532,531	1.078	6.735	3818.20

Results | Speedups on RK

Dataset

- real dynamic networks, ranging from ≈ **85 K** to ≈ **36 M** edges
- type: communication, friendship, coauthorship, hyperlink



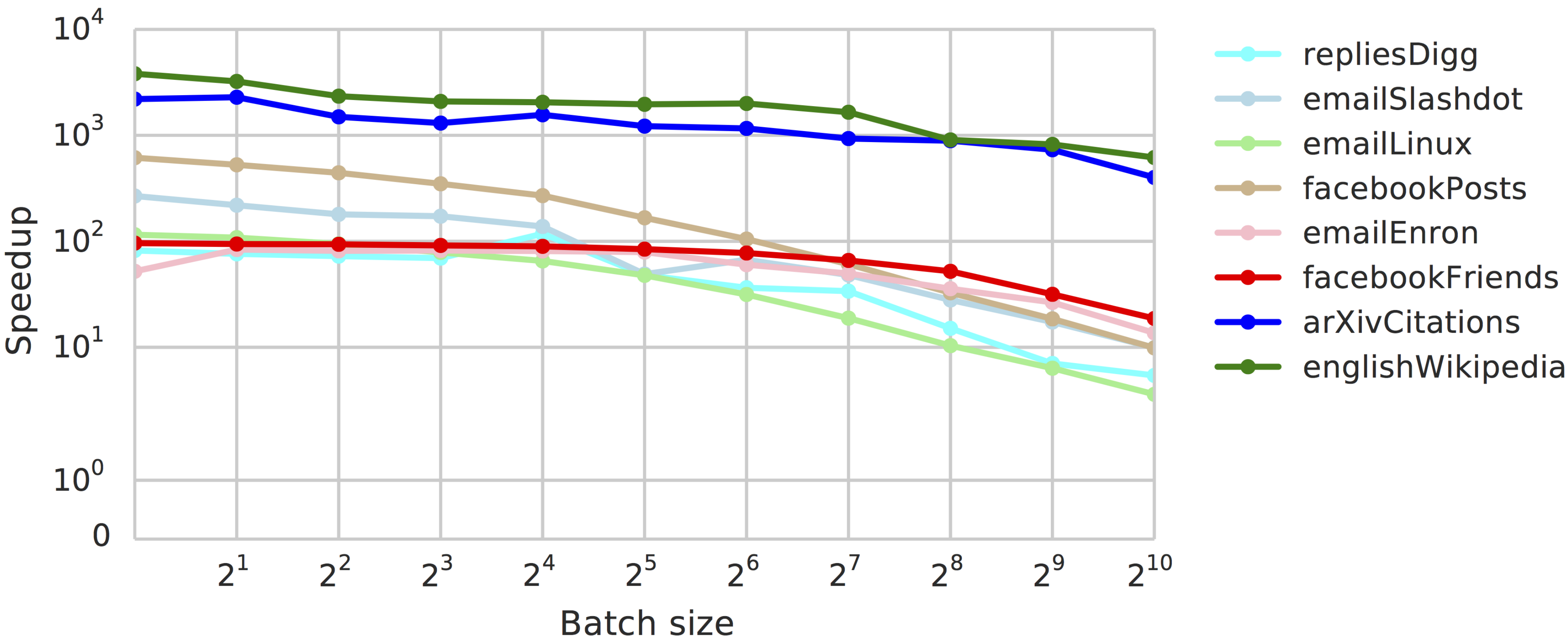
Speedups on static recomputation with RK

Results | Speedups on RK

Dataset

- real … **6 M** edges
- type … erlink

speedups up to more than 10^3 and always faster than recomputation



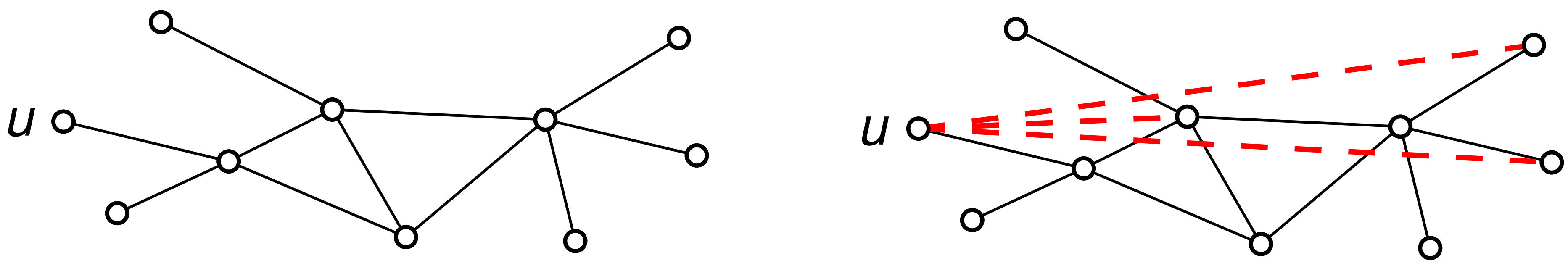
Speedups on static recomputation with RK

Updating betweenness centrality of a single node

Maximum Betweenness Improvement

Maximum Betweenness Improvement (MBI) Problem:

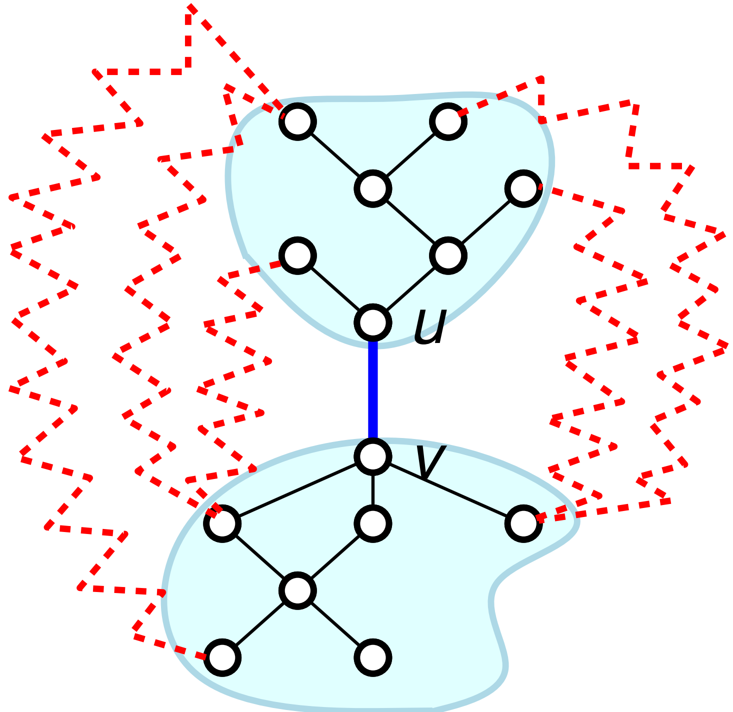
- Given node u and and $k > 0$, add k **new edges** incident to u in order to **maximize** $c_B(u)$
- Motivation: **High betweenness** can be **beneficial** for a node:
  - More “traffic” flowing through a node: more customers
  - Widely studied for PageRank (link farming)



- GREEDY [Crescenzi et al., SEA 2015]: new greedy algorithm
  - Add new edges several times and recompute $c_B(u)$ every time
  - Very expensive: $\Theta(k \cdot |V|^2|E|)$ operations

Single-node betweenness update

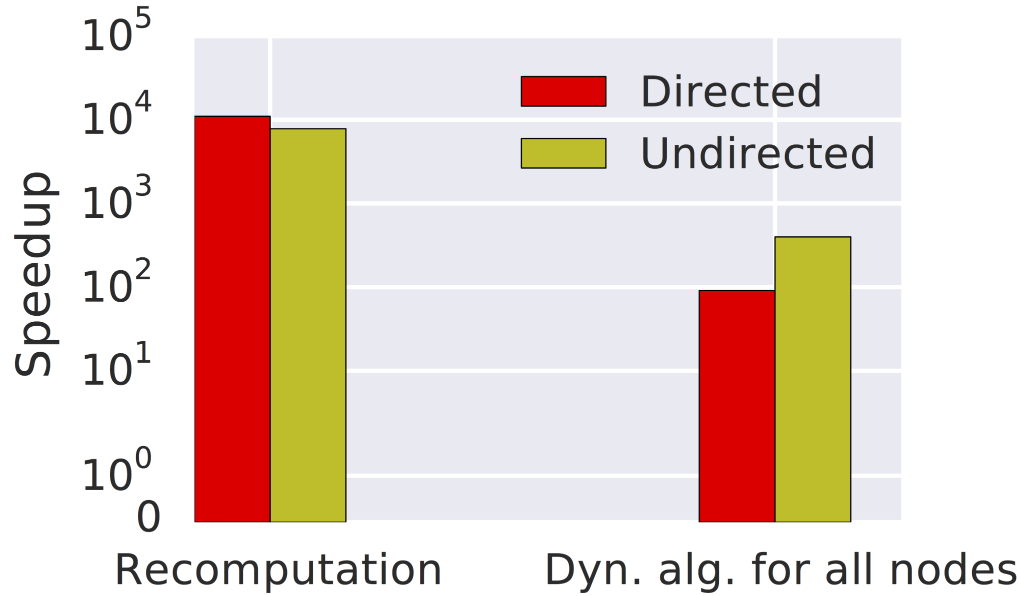
- Use a dynamic algorithm to recompute $c_B(u)$ after each insertion
  - Existing algorithms update score of **all nodes**
  - Observation: If we add edge (u, v), $c_B(u)$ can only **increase**
  - We can just focus on nodes with a **new shortest path** going through (u, v)



- Traditional dynamic betweenness algorithms need to update also betweenness of nodes that **lied in old shortest paths**

Single-node betweenness update

- Compared to existing dynamic algorithms:
  reduced worst-case complexity from $O(|V||E|)$ to $O(|V|^2)$
- Much faster in practice:



- With our new dynamic algorithm, the greedy algorithm for MBI takes **seconds** or **a few minutes** on graphs with up to 10^5 **edges** (before: hours for a few hundreds edges)

Conclusions

- Dynamic algorithms are a necessity for networks that continuously evolve over time
- We considered two different problems:
  - Update an **approximation** of betweenness **for all nodes**
  - Update the betweenness of **one node** after an edge insertion
- Both approaches much faster than static algorithms, but they require **additional memory**: $\Theta(r|V|)$ and $\Theta(|V|^2)$
- Open problems:
  - Can we reduce the memory requirements of dynamic algorithms?
  - Can we devise a faster static algorithm for the betweenness of a single node?

Thank you for your attention.

Aknowledgements

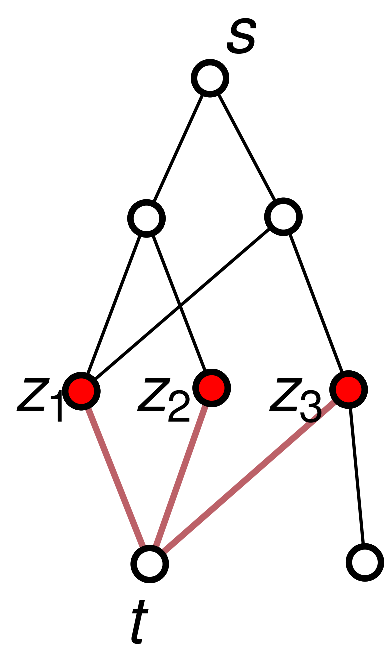
This work was partially supported by DFG grant FINCA within the SPP 1736

Paths sampling

- sample a vertex pair (s, t) uniformly at random $\rightarrow$ ($n(n-1)$ pairs)
- **extended** SSSP from s $\rightarrow$ distances + **number of shortest paths** + **list of predecessors**
- starting from t, **select a predecessor** z with probability

$$\frac{\sigma_z}{\sigma_t}$$

- repeat this until we reach s
- every shortest path between s and t has **the same probability** to be sampled

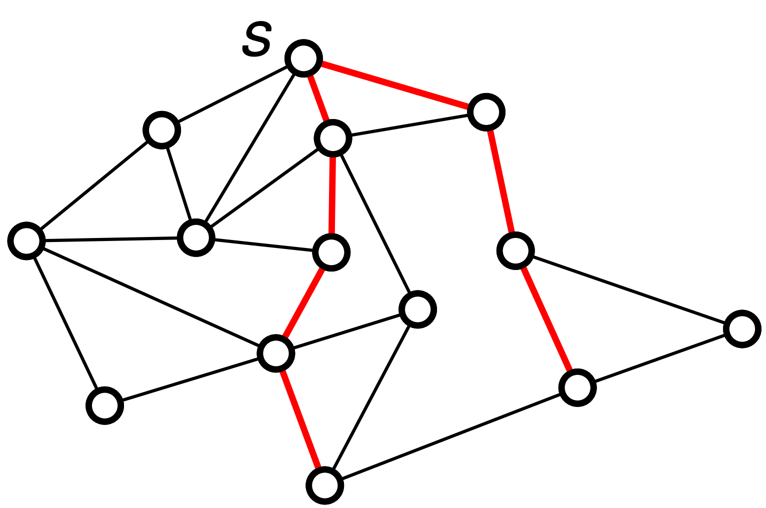


$P(z_1) = \frac{2}{4}$, $P(z_2) = \frac{1}{4}$, $P(z_3) = \frac{1}{4}$

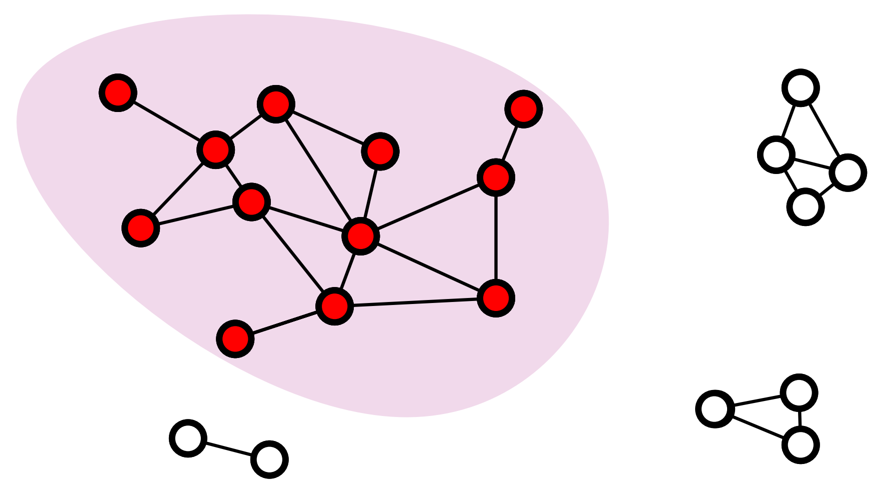
RK algorithm | Vertex diameter

- VD = number of nodes in the shortest path with the **maximum number of nodes**
- unweighted graphs: equal to **diameter** +1,
  weighted graphs: **unrelated**
- exact computation requires APSP $\rightarrow$ **approximation**

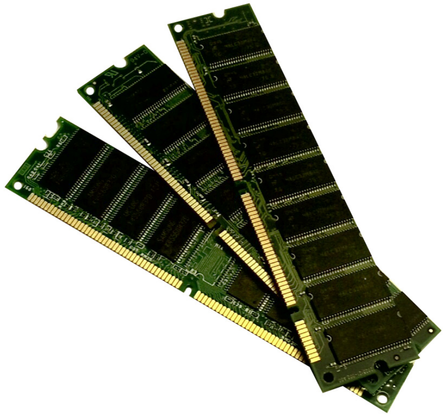
Connected unweighted graphs



Other graphs...



Karlsruhe Institute of Technology



Memory footprints

- dynamic exact algorithms: at least $\Theta(n^2)$
- BA and RK: $\Theta(m)$
- our algorithms: $\Theta(r \cdot n)$

On graphs with millions of edges

- dynamic exact algorithms are limited by their memory requirements
- RK and our algorithm are still fast but their memory requirements may exceed internal memory

➡ both RK and our algorithm could benefit from efficient **external memory** implementations