Efficient Estimation of Heat Kernel PageRank for Local Clustering

**Renchi Yang, Xiaokui Xiao, Zhewei Wei,
Sourav S Bhowmick, Jun Zhao, Rong-Hua Li**

July 2019

NANYANG TECHNOLOGICAL UNIVERSITY
NUS National University of Singapore School of Computing
中国人民大學 RENMIN UNIVERSITY OF CHINA
北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

Outline

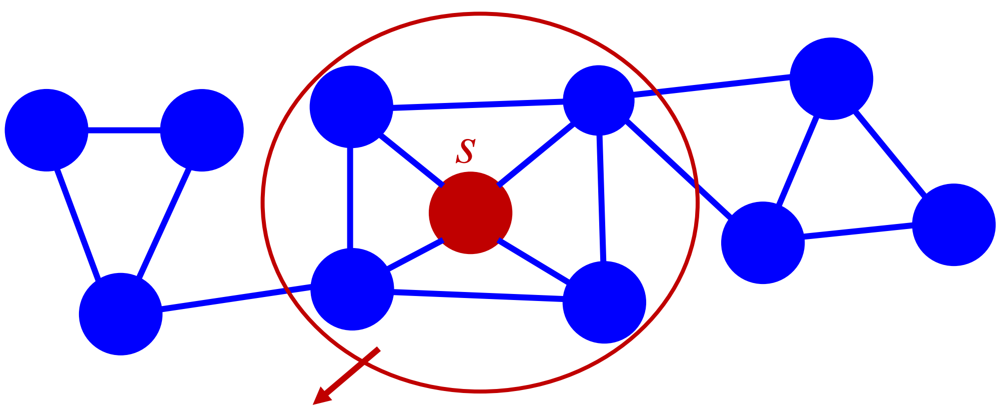
- Problem Definition & Applications
- Deficiencies of Existing Solutions
- Proposed solutions: TEA & TEA+
- Experiments

NANYANG TECHNOLOGICAL UNIVERSITY

Problem Definition: Local Graph Clustering

- Given an undirected graph G and seed node s, local clustering only explores a *local* neighborhood of s
- finds a cluster C with *minimal* conductance

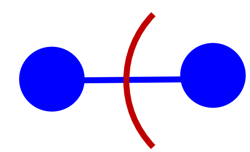
$$\phi(C) = \frac{\#(edges\ cut)}{\min\{\sum_{u \in C} d(u), |E| - \sum_{u \in C} d(u)\}} = \frac{3}{15}$$



local cluster C

$G = (V, E)$

edge cut:



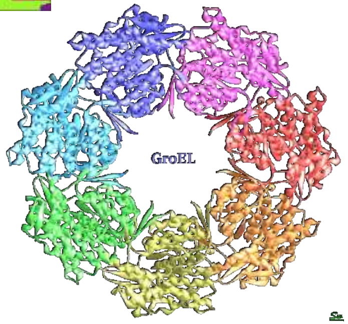
NANYANG TECHNOLOGICAL UNIVERSITY

Applications

- Community detection
  - [Leskovec *et al*. WWW'2010]
  - [Wang *et al*. VLDB'2015]

- Image segmentation
  - [Felzenszwalb *et al*. IJCV'2004]
  - [Tolliver *et al*. CVPR'2006.]

- Protein grouping
  - [Voevodski *et al*. BMC Bioinformatics'2009]
  - [Liao *et al*. Bioinformatics'2009]



Problem Definition: Heat-Kernel-based Local Clustering

- Heat Kernel PageRank (HKPR)
  - $\boldsymbol{\rho}_s[v] = \mathbb{P}[\text{random walk of length-}k \text{ from } s \text{ stops at } v]$
  - k follows a Poisson distribution with mean t, *i.e.*,

$$k \xrightarrow{\text{is sapmled with}} \eta(k) = \frac{e^{-t}t^k}{k!}$$

- Sweep Cut
  - A sweep over a sorted approximate degree-normalized HKPR vector to find C with *minimal* $\phi(C)$

$$\frac{\hat{\boldsymbol{\rho}}_s[v]}{\mathrm{d}(v)} = \overbrace{\begin{array}{|c|c|c|c|c|c|}\hline 0.15 & 0.14 & 0.12 & 0.1 & 0.08 & 0.06 \\ \hline\end{array}}^{C} \begin{array}{|c|c|c|c|c|}\hline \cdots & \cdots & \cdots & \cdots & \\ \hline\end{array}$$

v_1	v_2	v_3	v_4	v_5	v_6	…	…	…		v_n
0.15	0.14	0.12	0.1	0.08	0.06	…	…	…	…	

Deficiencies of Existing Solutions

- HK-Relax
  - Promises *same absolute-error* guarantees in terms of the degree-normalized HKPR of all nodes

Exact HKPR

	v_1	v_2
$\frac{\rho_{s[v]}}{d(v)}$	0.1 $<$	0.12

HK-Relax's Results

	v_1	v_2
$\frac{\hat{\rho}_{s[v]}}{d(v)}$	0.12 $>$	0.1

absolute error 0.02 for both, regardless of their degree-normalized HKPR

NANYANG TECHNOLOGICAL UNIVERSITY

Deficiencies of Existing Solutions

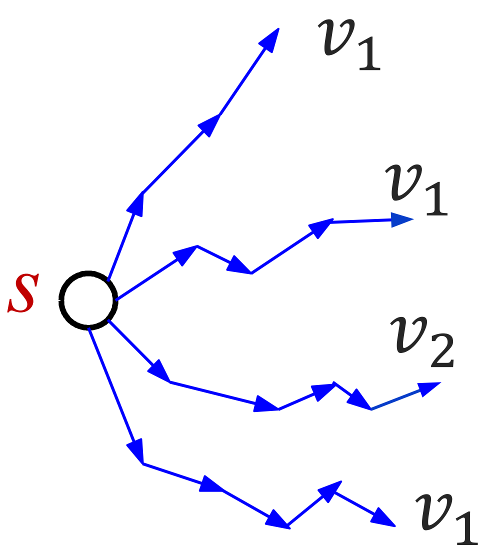
- ClusterHKPR
  - Promises *relative-error* guarantees in terms of the degree-normalized HKPR of some nodes
  - Incurs a high complexity due to a large number of random walks

Our Solution

- (d, ϵ_r, δ)-approximate HKPR
  - For important nodes ($i.e.$, $\frac{\rho_s[v]}{d(v)} > \delta$)
$$\left|\frac{\hat{\rho}_s[v]}{d(v)} - \frac{\rho_s[v]}{d(v)}\right| \le \epsilon_r * \frac{\rho_s[v]}{d(v)}$$
  - For other nodes
$$\left|\frac{\hat{\rho}_s[v]}{d(v)} - \frac{\rho_s[v]}{d(v)}\right| \le \epsilon_r * \delta$$
  - Time Complexity: $O\left(\frac{t\log(n/p_f)}{\epsilon_r^2 * \delta}\right)$

NANYANG TECHNOLOGICAL UNIVERSITY

Monte-Carlo Random Walks



At k-th hop from s

- Stops with a certain probability that relates to k
- Jumps to a random neighbour

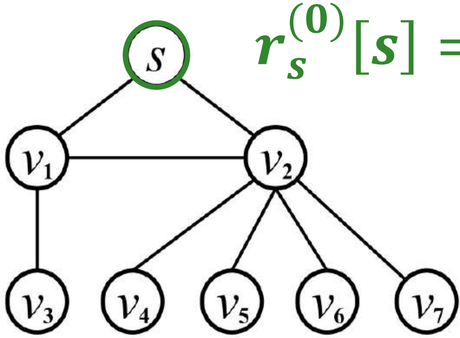
$\hat{\rho}_s[v]$= Fraction of random walks stopping at v

$$\boxed{\omega = \frac{2\left(1+\frac{\epsilon_r}{3}\right)\log\left(\frac{n}{p_f}\right)}{\epsilon_r^2\delta} \text{ random walks}}$$

Inefficient!

HK-Push: deterministic graph traversal

- HK-Push $t = 3, K = 2, n_p = \frac{\omega * t}{2}$
  - Each node v:
    - a *reserve* $\hat{\rho}_s[v]$: the portion of random walks stopped at v
    - a k-hop *residue* $r_s^{(k)}[v]$: the portion of random walks of length k currently at node v (*not* stopped yet)



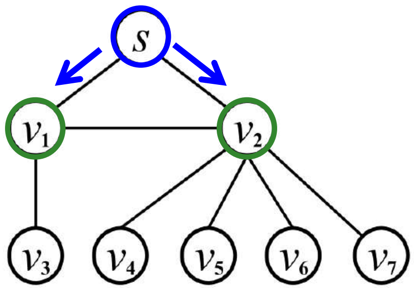
$\boldsymbol{r_s^{(0)}[s] = 1}$

At k-th step from *s*

- $\frac{\eta(k)}{\varphi(k)} * r_s^{(k)}[v] \rightarrow \hat{\rho}_s[v]$
- *Push* remaining portion to neighbours

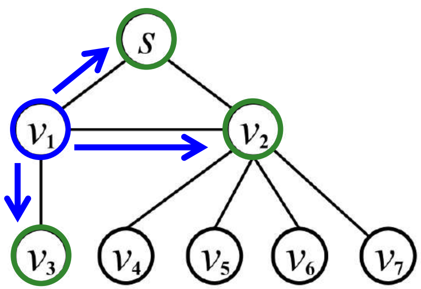
HK-Push: deterministic graph traversal

①



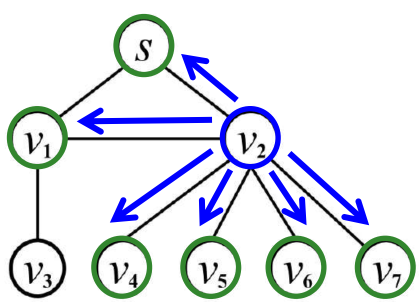
	s	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$\hat{\rho}_s$	**0.05**	0	0	0	0	0	0	0
$r_s^{(1)}$	0	**0.475**	**0.475**	0	0	0	0	0

②



	s	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$\hat{\rho}_s$	**0.05**	**0.075**	0	0	0	0	0	0
$r_s^{(1)}$	0	0	**0.475**	0	0	0	0	0
$r_s^{(2)}$	**0.133**	0	**0.133**	**0.133**	0	0	0	0

③



	s	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$\hat{\rho}_s$	**0.05**	**0.075**	**0.075**	0	0	0	0	0
$r_s^{(1)}$	0	0	0	0	0	0	0	0
$r_s^{(2)}$	**0.2**	**0.067**	**0.133**	**0.133**	**0.067**	**0.067**	**0.067**	**0.067**

Challenges

- HK-Push generates a *rough* approximation of $\boldsymbol{\rho}_s$ *fast*
- Monte-Carlo produces an *accurate* approximation of $\boldsymbol{\rho}_s$ *inefficiently*
- How to make use both of them to ensure
  - Improved **efficiency**
  - Strong theoretical **accuracy** guarantees
  - Strong theoretical **time complexity**

NANYANG TECHNOLOGICAL UNIVERSITY

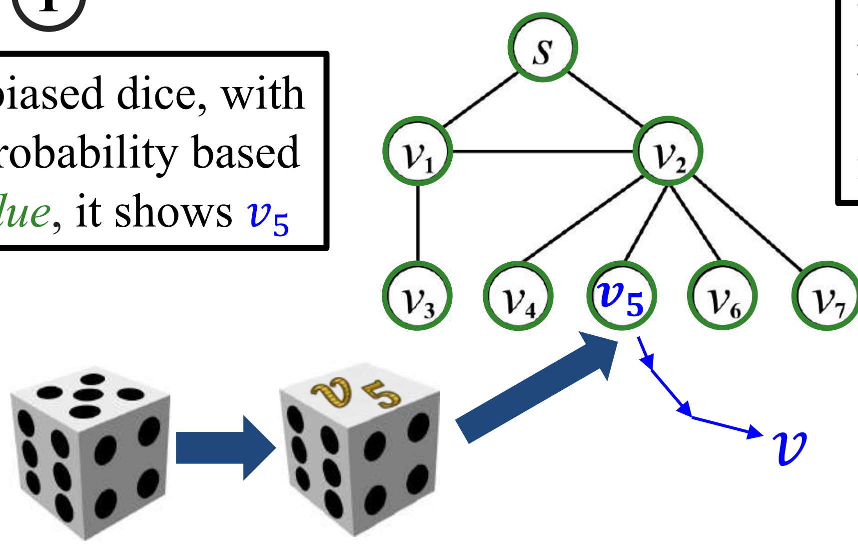
TEA: HK-Push + Random Walks

1

Rolling a biased dice, with a certain probability based on its *residue*, it shows v_5

2

At ℓ-th step, with a certain probability, the random walk stops, otherwise, jumps to a random neighbor



3

If the random walk ends at v, increase v's approximate HKPR by a constant, otherwise 0

TEA: Analysis

- Time complexity: $O(\frac{t\log(n/p_f)}{\epsilon_r^2 * \delta})$
- Space complexity

$$O(m + n + \frac{t\log(n/p_f)}{\epsilon_r^2 * \delta})$$

Still many random walks to perform.
How to reduce the number of random walks *without*

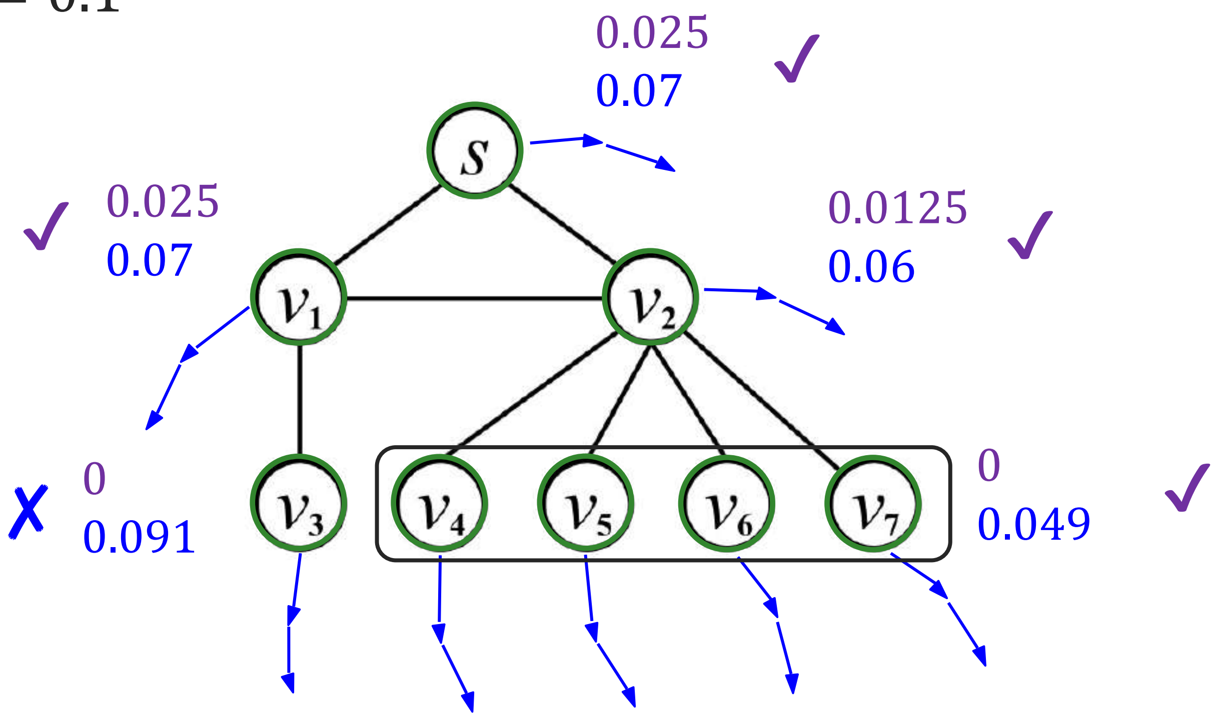
- *increasing the cost of* HK-Push, &
- *degrading theoretical guarantees* ?

NANYANG TECHNOLOGICAL UNIVERSITY

Our Observation

$t = 3, \epsilon_r = 0.9, \delta = 0.1$

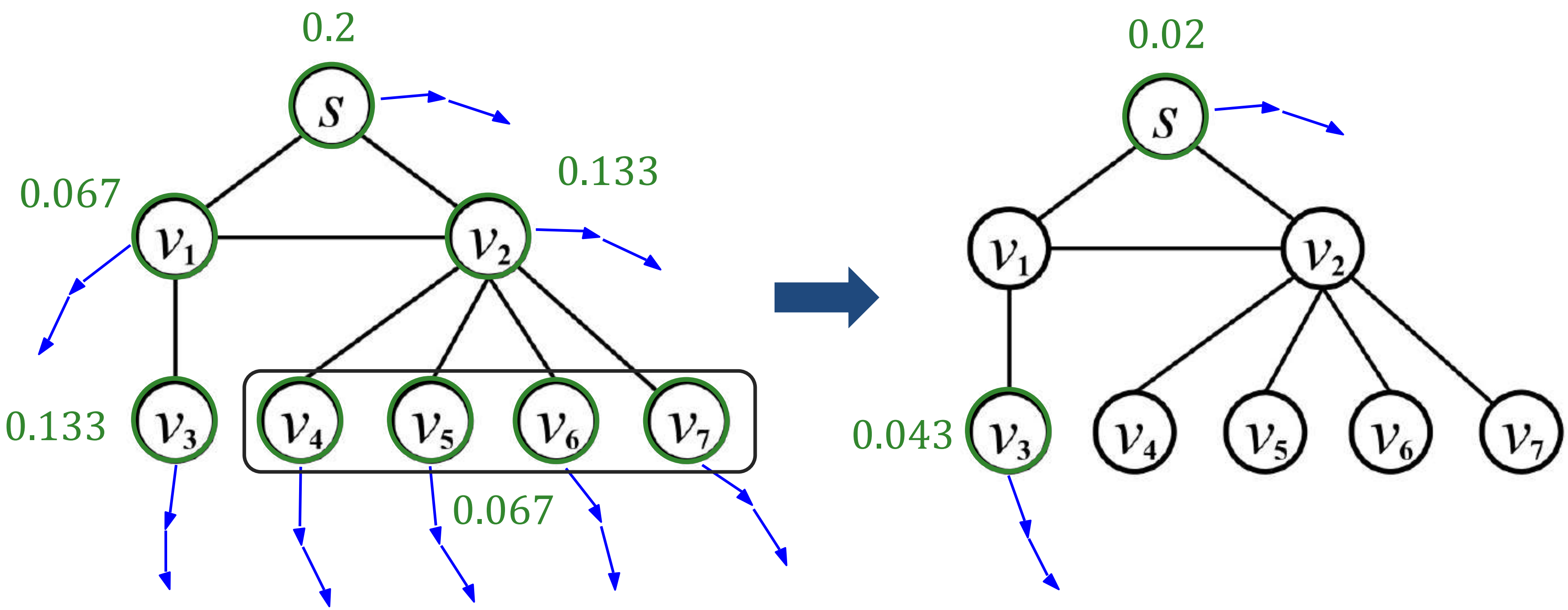
- Approximate normalized HKPR
- Exact normalized HKPR



Only node v_3 does *not* have an accurate approximate normalized HKPR

TEA+: Pruning Random Walks

$t = 3, \epsilon_r = 0.9, \delta = 0.1$



$\epsilon_r * \delta = 0.09$

$r_s^{(k)}[v] = \max\{0, r_s^{(k)}[v] - 0.09 * d(v)\}$

TEA+: Analysis

- Time complexity: $O(\frac{t\log(n/p_f)}{\epsilon_r^2 * \delta})$
- Space complexity

$$O(m + n + \frac{t\log(n/p_f)}{\epsilon_r^2 * \delta})$$

Experimental Setup

- **Environment**: a Linux server with a Intel 2.60GHz CPU and 64GB RAM.
- **Query set**: 50 seed nodes uniformly at random as our query sets.
- **Our methods**
  - TEA and TEA+ ($t = 5, p_f = 10^{-6}$)
- **Competitors**
  - Monte-Carlo: *random walks* (HKPR)
  - ClusterHKPR: *truncated random walks* (HKPR)
  - HK-Relax: *deterministic graph traversal* (HKPR)
  - SimpleLocal: *three-stage local max flow*
  - CRD: *capacity releasing diffusion*

NANYANG TECHNOLOGICAL UNIVERSITY

Datasets

Table 7: Statistics of graph datasets.

Dataset	n	m	$\bar{d}$
DBLP	317,080	1,049,866	6.62
Youtube	1,134,890	2,987,624	5.27
PLC	2,000,000	9,999,961	9.99
Orkut	3,072,441	117,185,083	76.28
LiveJournal	3,997,962	34,681,189	17.35
3D-grid	9,938,375	29,676,450	5.97
Twitter	41,652,231	1,202,513,046	57.74
Friendster	65,608,366	1,806,067,135	55.06

NANYANG TECHNOLOGICAL UNIVERSITY

Comparisons with Competitors

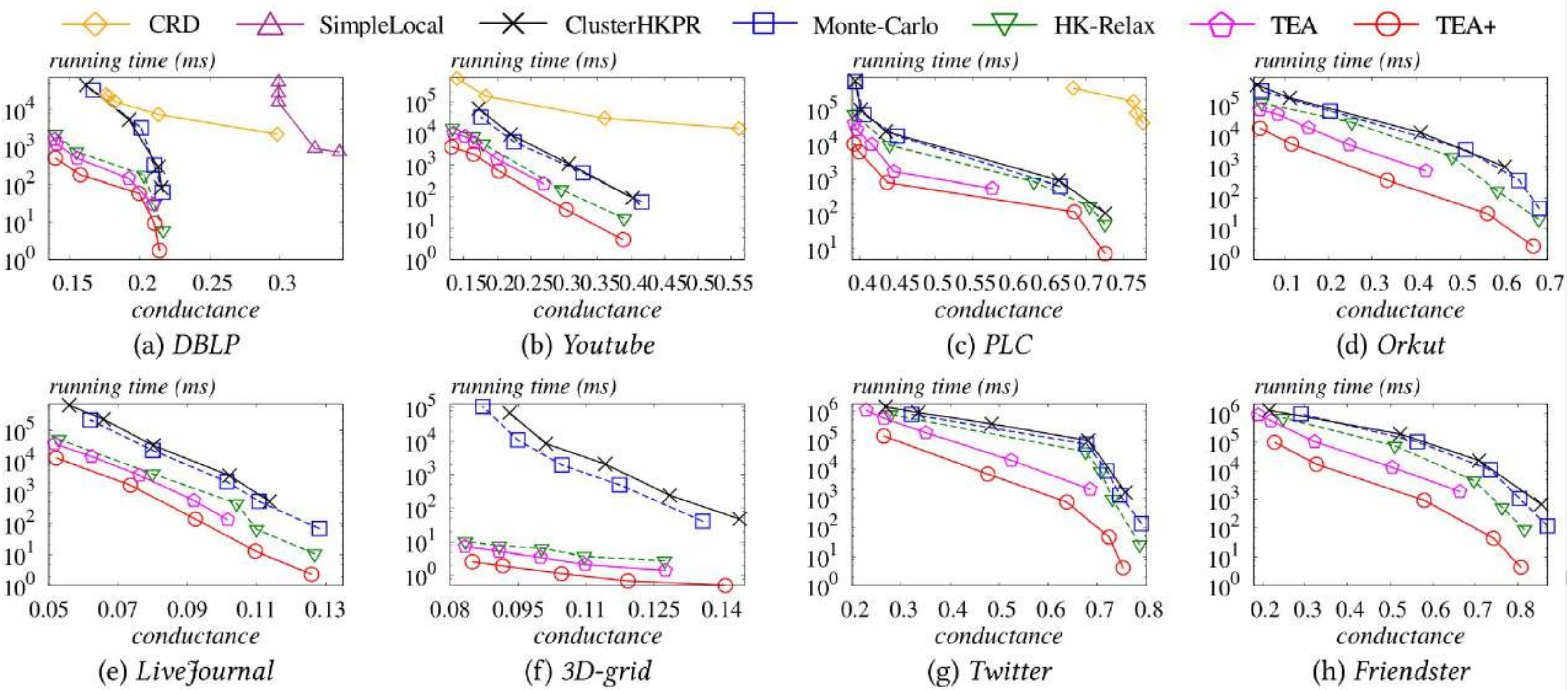


Figure 3: Running time vs conductance for local clustering queries (best viewed in color).

Thanks

Q & A

NANYANG TECHNOLOGICAL UNIVERSITY