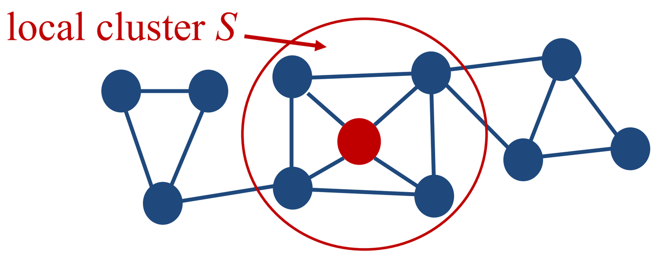
Efficient Estimation of Heat Kernel PageRank for Local Clustering

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

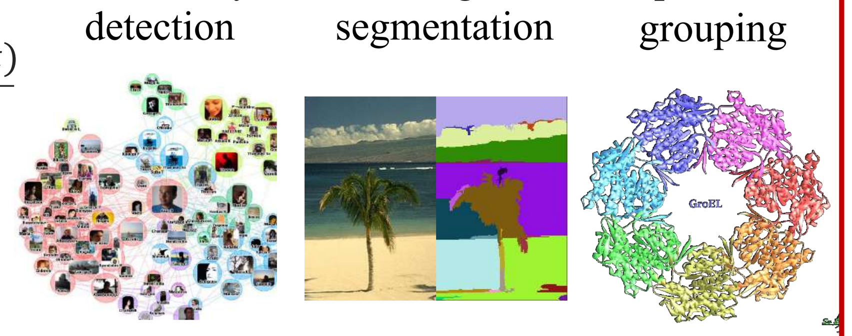
NANYANG TECHNOLOGICAL UNIVERSITY
NUS National University of Singapore School of Computing
中国人民大学 RENMIN UNIVERSITY OF CHINA
北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

1. Heat Kernel-based Local Clustering

- Local Clustering
  - Explodes the local neighbourhood around the seed node only to find S
  - S has min conductance $\phi(S) = \frac{\#(edges\ cut)}{\sum_{u \in S} d(u)}$



- Applications



- The Heat Kernel PageRank (HKPR) from s to v is
  - $\boldsymbol{\rho}_s[v] = \mathbb{P}$[Random walk of length-$k$ from s stops at v]
  - k follows a Poisson distribution with mean t; k's probability: $\eta(k) = \frac{e^{-t}t^k}{k!}$
- Sweep Cut

$\frac{\boldsymbol{\rho}_s[v]}{d(v)}$

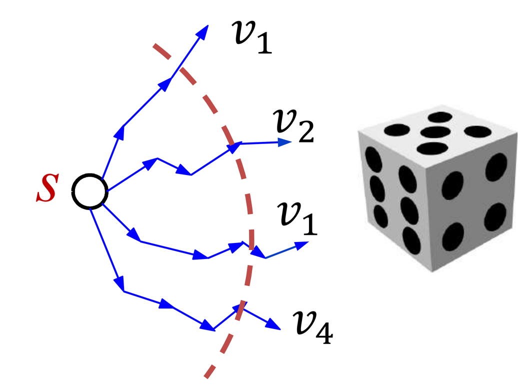
v_1	v_2	v_3	v_4	v_5	v_6	…	…	…	v_n
0.15	0.14	0.12	0.1	0.08	0.06	…	…	…	

S = $\{v_1, \dots, v_6\}$

A sweep over $\boldsymbol{\rho}_s$ to find a cluster with *minimal* $\phi(S)$

2. Existing Approximate Solutions

- ClusterHKPR
  - Samples a random walk length $k \le O\left(\frac{\log(1/\epsilon)}{\log\log(1/\epsilon)}\right)$
  - Starts the k-step truncated random walk from s
  - Repeats the above process for $16\log n/\epsilon^3$ times



- HK-Relax
  - Injects initial residual ($\boldsymbol{r}_s[s,0] = e^{-t}$) to the seed node s
  - At k-th step from s ($k \le K = 2t\log(1/\epsilon_a)$), converts $\boldsymbol{r}_s[v,k] \to \boldsymbol{\rho}_s[v]$; distributes $\frac{t}{k+1} \cdot r_s[v,k]$ to neighbors evenly
  - Stops until all $\frac{r_s[v,k]}{d(v)} \le \frac{e^t \epsilon_a}{2K} / \left(\sum_{i=0}^{K-k} \frac{k!}{(k+i)!} t^i\right)$

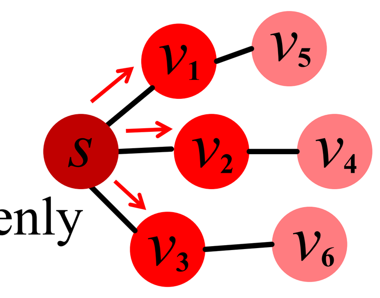


Table 1: Theoretical guarantee of our solution against that of the state-of-the-art solutions.

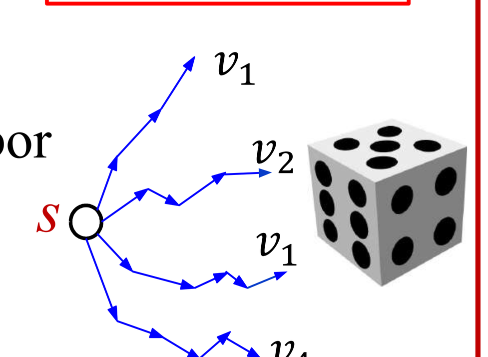
Algorithm	Accuracy Guarantee	Time Complexity
ClusterHKPR [11]	with probability at least $1-\epsilon$, $\begin{cases} \lvert\widehat{\boldsymbol{\rho}}_s[v] - \boldsymbol{\rho}_s[v]\rvert \le \epsilon \cdot \boldsymbol{\rho}_s[v], & \text{if } \boldsymbol{\rho}_s[v] > \epsilon \\ \lvert\widehat{\boldsymbol{\rho}}_s[v] - \boldsymbol{\rho}_s[v]\rvert \le \epsilon, & \text{otherwise,} \end{cases}$	$O\left(\frac{t\log(n)}{\epsilon^3}\right)$
HK-Relax [17]	$\frac{1}{d(v)} \lvert\widehat{\boldsymbol{\rho}}_s[v] - \boldsymbol{\rho}_s[v]\rvert < \epsilon_a$	$O\left(\frac{te^t\log(1/\epsilon_a)}{\epsilon_a}\right)$
Our solutions	with probability at least $1-p_f$, $\begin{cases} \frac{1}{d(v)} \lvert\widehat{\boldsymbol{\rho}}_s[v] - \boldsymbol{\rho}_s[v]\rvert \le \epsilon_r \cdot \frac{\boldsymbol{\rho}_s[v]}{d(v)}, & \text{if } \frac{\boldsymbol{\rho}_s[v]}{d(v)} > \delta \\ \frac{1}{d(v)} \lvert\widehat{\boldsymbol{\rho}}_s[v] - \boldsymbol{\rho}_s[v]\rvert \le \epsilon_r \cdot \delta, & \text{otherwise,} \end{cases}$	$O\left(\frac{t\log(n/p_f)}{\epsilon_r^2 \cdot \delta}\right)$

3. The Basic Ideas

- (d, ϵ_r, δ)-approximate HKPR
  - $\forall\, v \in G\ s.t.\ \boldsymbol{\rho}_s[v]/d(v) > \delta$, $\quad \left|\frac{\widehat{\boldsymbol{\rho}}_s[v]}{d(v)} - \frac{\boldsymbol{\rho}_s[v]}{d(v)}\right| \le \epsilon_r \cdot \frac{\boldsymbol{\rho}_s[v]}{d(v)}$;
  - $\forall\, v \in G\ s.t.\ \boldsymbol{\rho}_s[v]/d(v) \le \delta$, $\quad \left|\frac{\widehat{\boldsymbol{\rho}}_s[v]}{d(v)} - \frac{\boldsymbol{\rho}_s[v]}{d(v)}\right| \le \epsilon_r \cdot \delta$.
- Monte-Carlo Random Walks
  - Starting from s, at k-th step, with probability $\frac{\eta(k)}{\psi(k)}$, the random walk stops; OTRW, jumps to a random neighbor

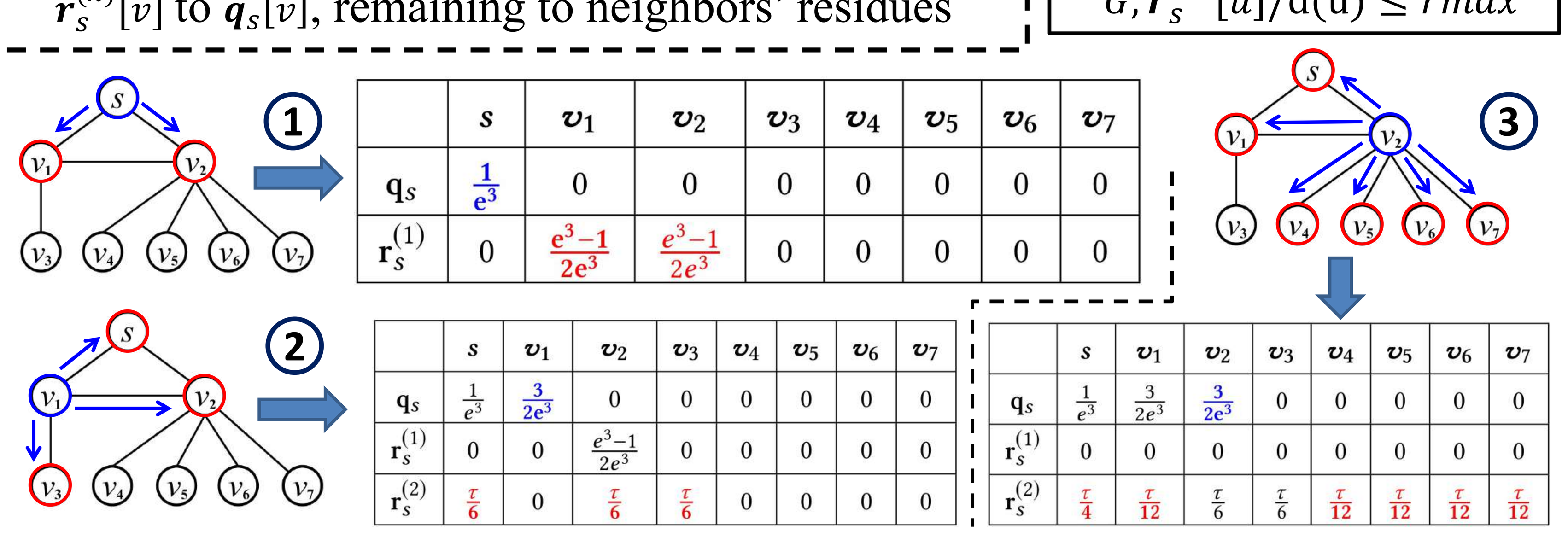
$\psi(k) = \sum_{\ell=0}^{\infty} \eta(\ell)$

  - Performing $\omega = \frac{2(1+\epsilon_r/3)\log(n/p_f)}{\epsilon_r^2 \delta}$ random walks from s produces a (d, ϵ_r, δ)-approximate HKPR vector



- HK-Push
  - $\boldsymbol{\rho}_s[v] = \boxed{\boldsymbol{q}_s[v]} + \boxed{\sum_{u \in G} \sum_{k=0}^{K} \boldsymbol{r}_s^{(k)}[u] \cdot \boldsymbol{h}_u^{(k)}[v]}$ (estimation; error!)
  - $\boldsymbol{h}_u^{(k)}[v] = \sum_{\ell=0}^{\infty} \frac{\eta(k+\ell)}{\varphi(k)} \cdot \mathbf{P}^{\ell}[u,v]$
  - A traversal of G from s, $\frac{\eta(k)}{\psi(k)}$ portion of residue $\boldsymbol{r}_s^{(k)}[v]$ to $\boldsymbol{q}_s[v]$, remaining to neighbors' residues

- Each node v: a *resreve* $\boldsymbol{q}_s[v]$, and *residue* $\boldsymbol{r}_s^{(k)}[v]$ at k-th step
- It first set $\boldsymbol{r}_s^{(0)}[s] = 1$ and stops pushing until $\forall u \in G, \boldsymbol{r}_s^{(k)}[u]/d(u) \le rmax$



①

	s	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$\mathbf{q}_s$	$\frac{1}{e^3}$	0	0	0	0	0	0	0
$\mathbf{r}_s^{(1)}$	0	$\frac{e^3-1}{2e^3}$	$\frac{e^3-1}{2e^3}$	0	0	0	0	0

②

	s	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$\mathbf{q}_s$	$\frac{1}{e^3}$	$\frac{3}{2e^3}$	0	0	0	0	0	0
$\mathbf{r}_s^{(1)}$	0	0	$\frac{e^3-1}{2e^3}$	0	0	0	0	0
$\mathbf{r}_s^{(2)}$	$\frac{\tau}{6}$	0	$\frac{\tau}{6}$	$\frac{\tau}{6}$	0	0	0	0

③

	s	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$\mathbf{q}_s$	$\frac{1}{e^3}$	$\frac{3}{2e^3}$	$\frac{3}{2e^3}$	0	0	0	0	0
$\mathbf{r}_s^{(1)}$	0	0	0	0	0	0	0	0
$\mathbf{r}_s^{(2)}$	$\frac{\tau}{4}$	$\frac{\tau}{12}$	$\frac{\tau}{6}$	$\frac{\tau}{6}$	$\frac{\tau}{12}$	$\frac{\tau}{12}$	$\frac{\tau}{12}$	$\frac{\tau}{12}$

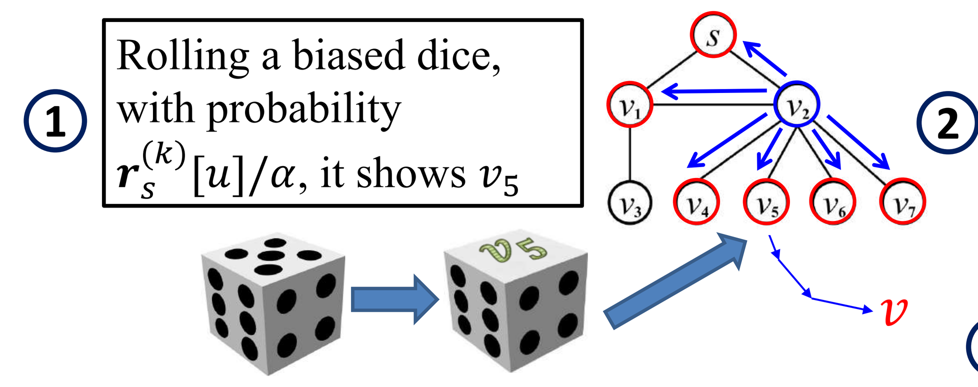
4. The TEA+ Algorithm

- A Combination of HK-Push and Random Walks
  - $\boldsymbol{\rho}_s[v] = \boldsymbol{q}_s[v] + \sum_{u \in G} \sum_{k=0}^{K} \boldsymbol{r}_s^{(k)}[u] \cdot \boxed{\boldsymbol{h}_u^{(k)}[v]} \Rightarrow \mathbb{E}[X]$ — Estimated by random variable X
  - $\alpha = \sum_{u \in G} \sum_{k=0}^{K} \boldsymbol{r}_s^{(k)}[u]$

① Rolling a biased dice, with probability $\boldsymbol{r}_s^{(k)}[u]/\alpha$, it shows v_5

② Starting from v_5, at ℓ-th step, with probability $\frac{\eta(k+\ell)}{\psi(k+\ell)}$, the random walk stops, otherwise, jumps to a random neighbor

③ If the random walk ends at v, X = 1, otherwise 0



  - needs $\alpha\omega$ random walks $\to O(\alpha\omega t)$ time
- Optimizations
  - Balancing HK-Push and random walks via
    - Maximum #hops from the seed node s, $K = c \cdot \frac{\log(1/\delta/\epsilon_r)}{\log(d)}$ — Tune c experimentally
    - Maximum #push-operations $n_p = \omega \cdot t/2$ — If cost(HK-Push) > cost(random walks)/2, switch to random walks
  - Pruning random walks
    - $\boldsymbol{rb}_s^{(k)}[u] = \min\{\boldsymbol{r}_s^{(k)}[u],\ \beta_k \cdot \epsilon_r \delta \cdot d(u)\}$, $\boldsymbol{r}_s^{(k)}[u] = \boldsymbol{r}_s^{(k)}[u] - \boldsymbol{rb}_s^{(k)}[u]$
    - $\boldsymbol{\rho}_s[v] = \boldsymbol{q}_s[v] + \boxed{\sum_{u \in G} \sum_{k=0}^{K} \boldsymbol{rb}_s^{(k)}[u] \cdot \boldsymbol{h}_u^{(k)}[v]} + \boxed{\sum_{u \in G} \sum_{k=0}^{K} \boldsymbol{r}_s^{(k)}[u] \cdot \boldsymbol{h}_u^{(k)}[v]}$

! NO random walks for this | directly estimated as $0.5 \cdot \epsilon_r \delta \cdot d(v)$ with an error of $0.5 \cdot \epsilon_r \delta \cdot d(v)$ | estimated by $\alpha \cdot \omega$ random walks with a much smaller $\boldsymbol{\alpha}$

- Analysis
  - Time: $O(n_p) + O(\alpha\omega t) \to \boldsymbol{O}\left(\frac{t\log(n/p_f)}{\epsilon_r^2 \delta}\right)$, Space: $\boldsymbol{O}\left(\boldsymbol{m} + \boldsymbol{n} + \frac{t\log(n/p_f)}{\epsilon_r^2 \delta}\right)$

5. Experimental Results

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

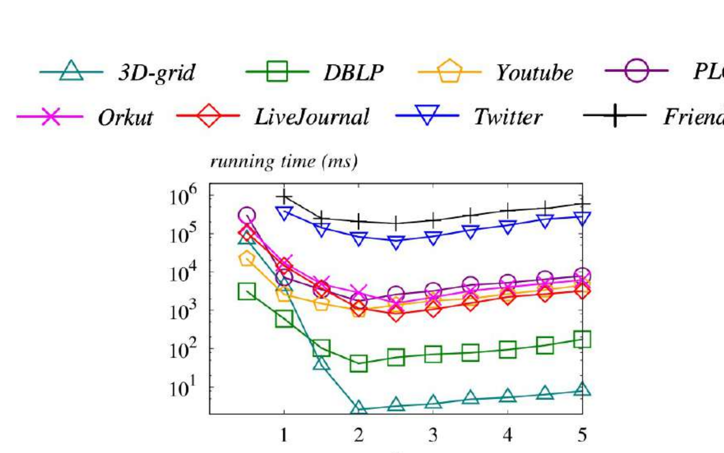


Figure 1: Running time of TEA+ vs c.

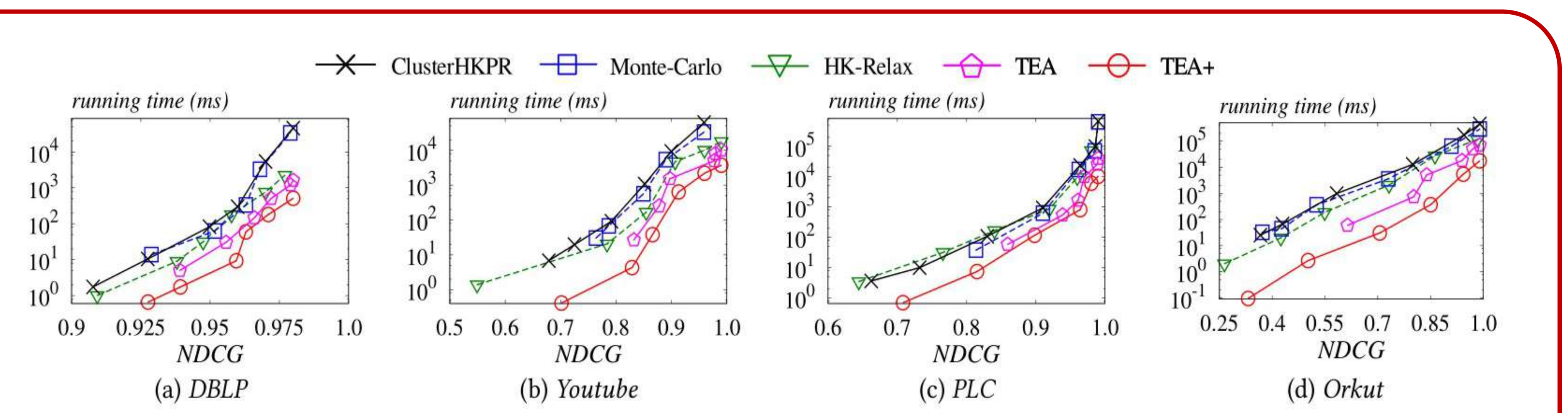


Figure 6: Running time vs. NDCG for computing normalized HKPR (best viewed in color).

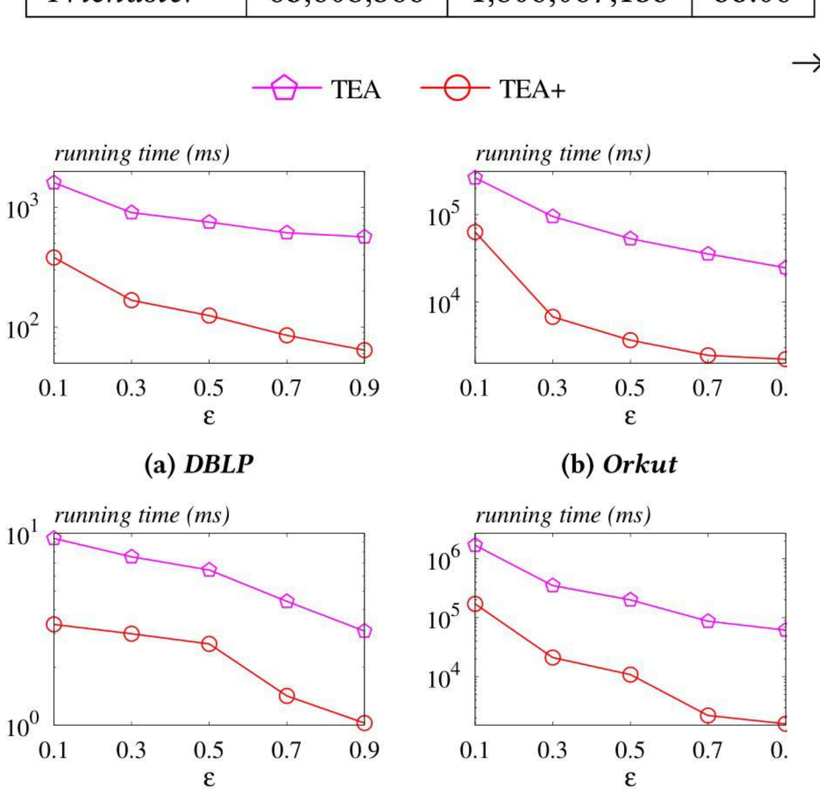


Figure 2: Running time vs ϵ_r.

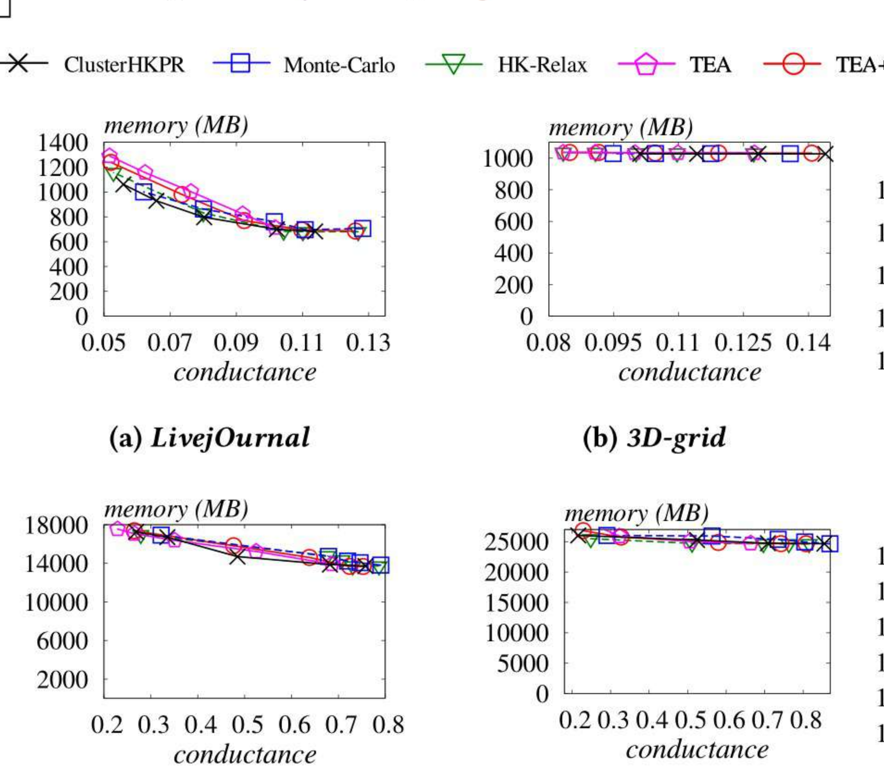


Figure 4: Memory cost vs. conductance.

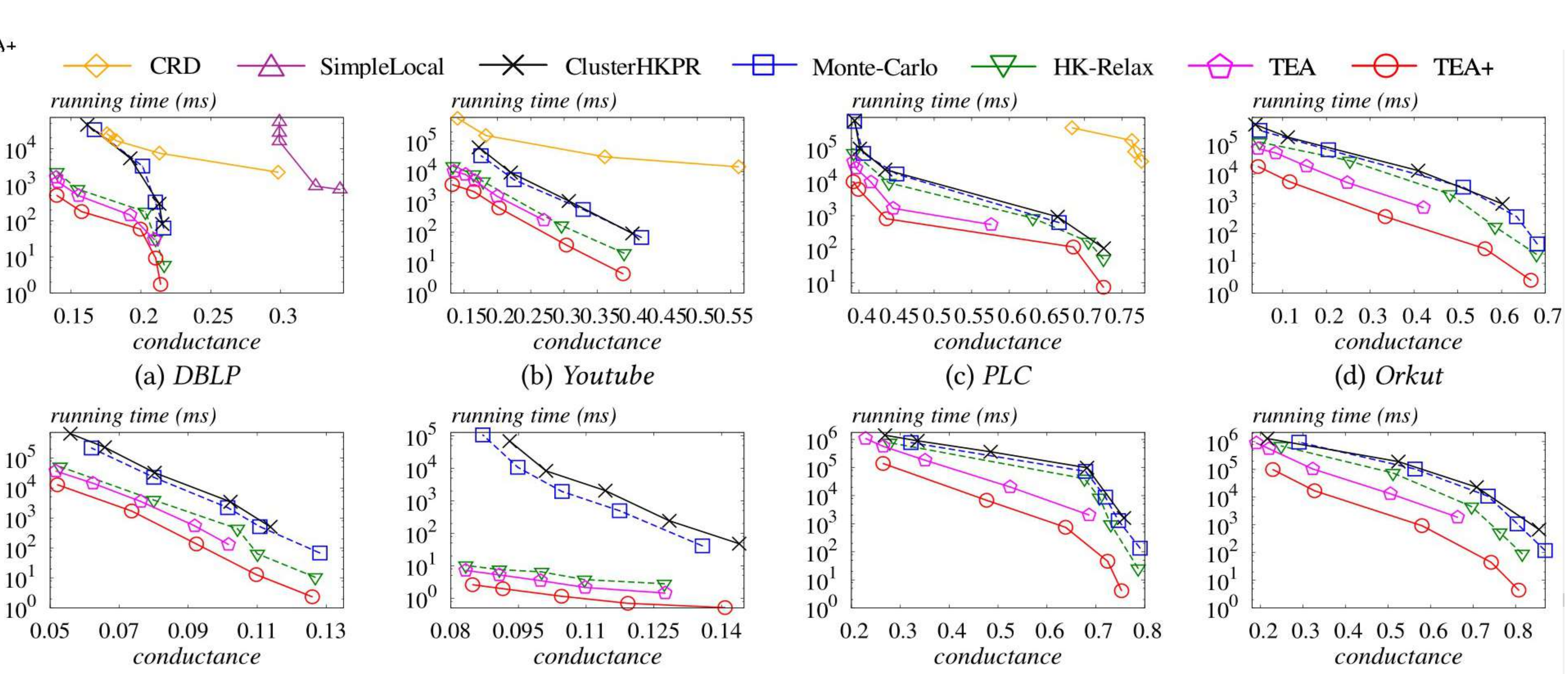


Figure 3: Running time vs conductance for local clustering queries (best viewed in color).