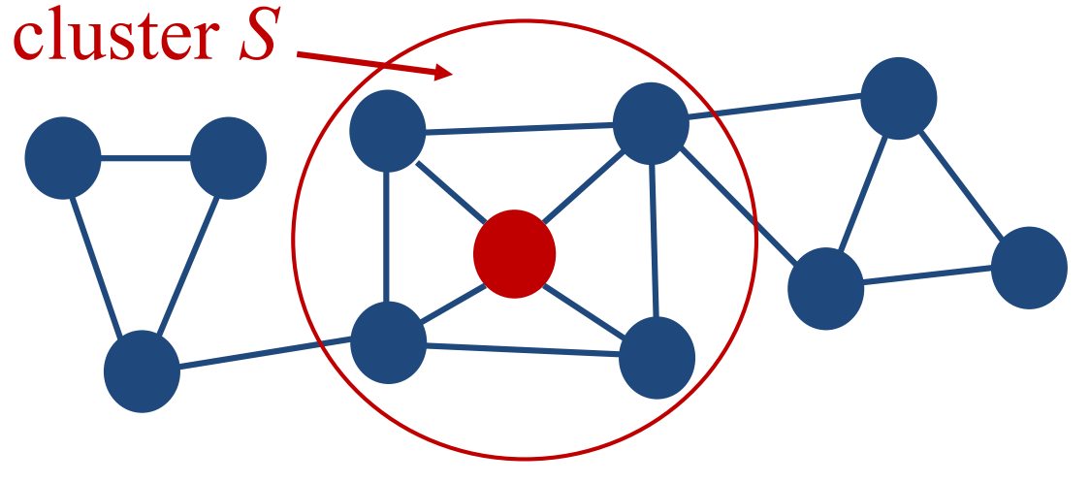
Efficient Estimation of Heat Kernel PageRank for Local Clustering

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

1. Heat Kernel-based Local Clustering

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

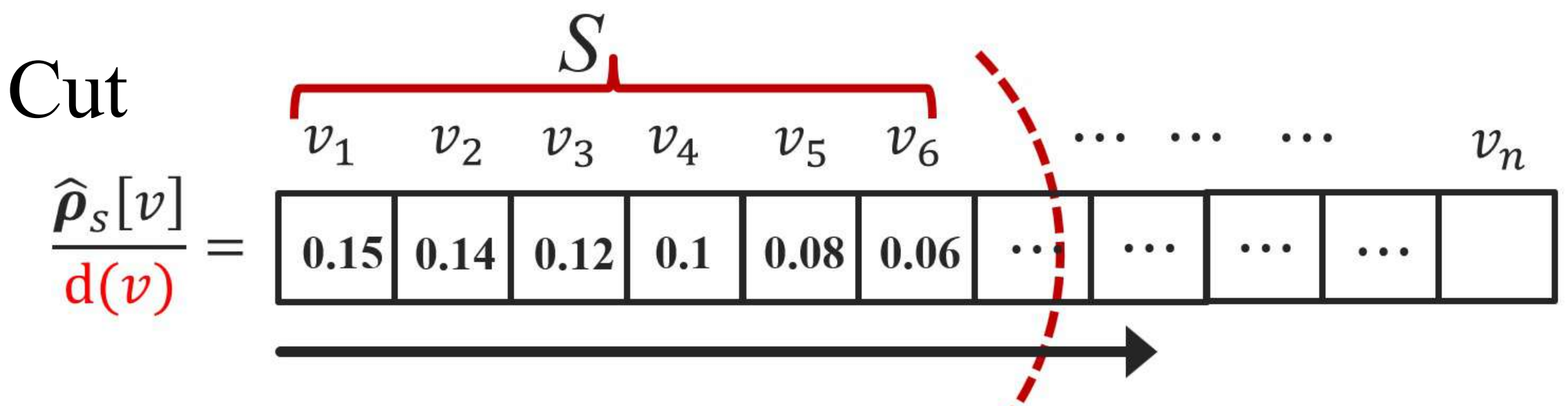


- Applications

community detection · image segmentation · protein grouping

- 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

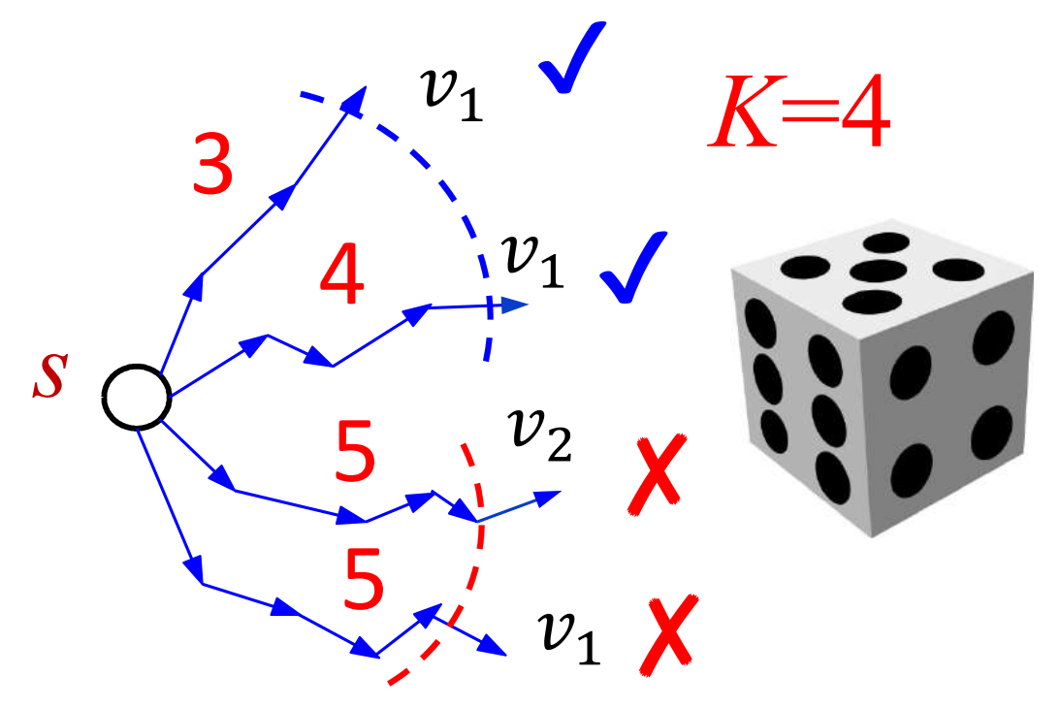
$\psi(k) = \sum_{\ell=k}^{\infty}\eta(\ell)$



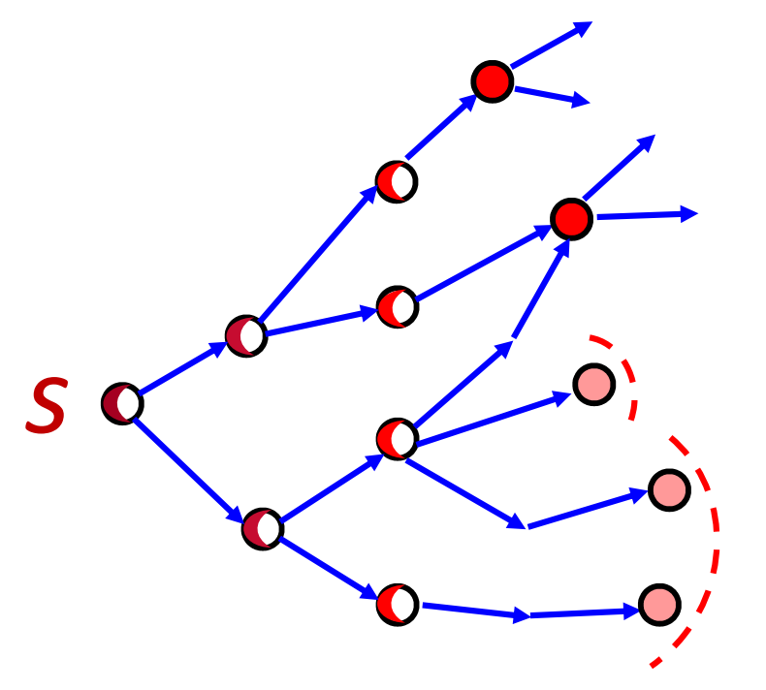
sweep over approximate HKPR vector $\hat{\boldsymbol{\rho}}_s$ to find S with *minimal* $\phi(S)$

2. Existing Approximate Solutions

- ClusterHKPR
  - Sets max random walk length $K = O(\frac{\log(1/\epsilon)}{\log\log(1/\epsilon)})$
  - $16\log(n)/\epsilon^3$ truncated random walks from s
  - $\hat{\rho}_s[v]$= Fraction of random walks stopping at v



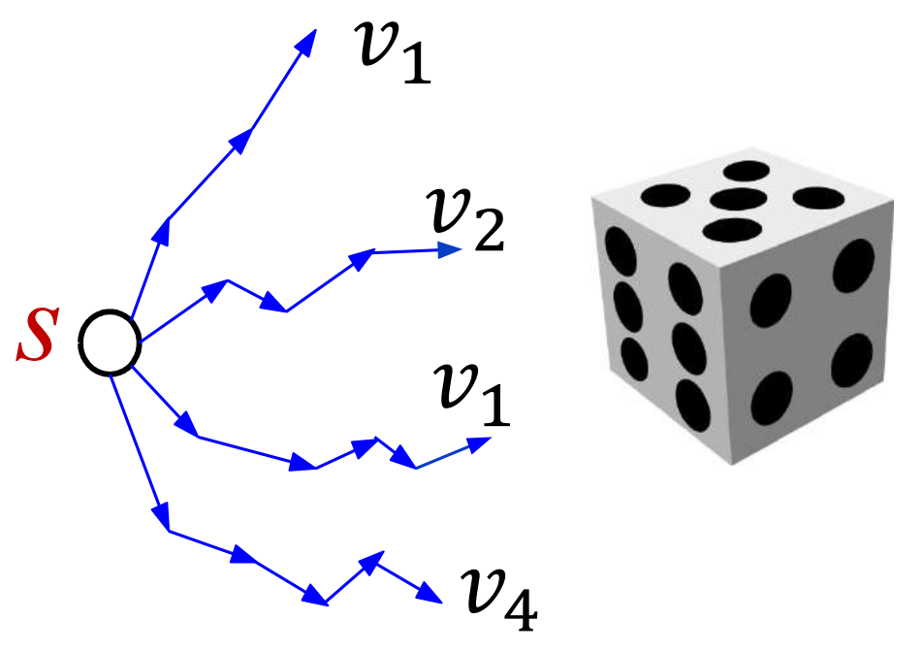
- HK-Relax
  - Sets initial residual $\boldsymbol{r}_s[s,0] = e^{-t}$
  - At k-th hop from s, $\boldsymbol{r}_s[v,k] \rightarrow$ reserve $(k \le 2t\log(\frac{1}{\epsilon_a}))$; distributes $\frac{t}{k+1}\times r_s[v,k]$ to neighbors evenly
  - $\hat{\rho}_s[v]$= Sum of reserves at v



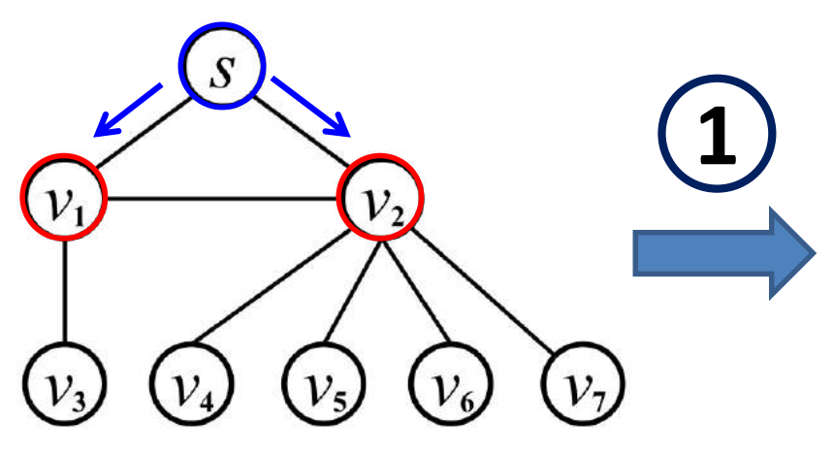
Algorithm	Accuracy Guarantee	Complexity
ClusterHKPR	$\mathbb{P}[\begin{cases}\lvert\hat{\rho}_s[v]-\rho_s[v]\rvert \le \epsilon\cdot\rho_s[v], & \text{if } \rho_s[v] > \epsilon\\ \lvert\hat{\rho}_s[v]-\rho_s[v]\rvert \le \epsilon, & \text{otherwise}\end{cases}] \ge 1-\epsilon$	$O(\frac{t\log(n)}{\epsilon^3})$
HK-Relax	$\lvert\frac{\hat{\rho}_s[v]}{d(v)} - \frac{\rho_s[v]}{d(v)}\rvert \le \epsilon_a$	$O(\frac{te^t\log(1/\epsilon_a)}{\epsilon_a})$
Our solutions	$\mathbb{P}\left[\begin{cases}\left\lvert\frac{\hat{\rho}_s[v]}{d(v)} - \frac{\rho_s[v]}{d(v)}\right\rvert \le \epsilon_r\cdot\frac{\rho_s[v]}{d(v)}, & \text{if } \frac{\rho_s[v]}{d(v)} > \delta\\ \left\lvert\frac{\hat{\rho}_s[v]}{d(v)} - \frac{\rho_s[v]}{d(v)}\right\rvert \le \epsilon_r\cdot\delta\,, & \text{otherwise}\end{cases}\right] \ge 1-p_f$	$O(\frac{t\log(n/p_f)}{\epsilon_r^2\cdot\delta})$

3. The Basic Ideas

- (d, ϵ_r, δ)-approximate HKPR
  - $\forall\, v \in G\ \ s.t.\ \boldsymbol{\rho}_s[v]/\mathrm{d}(v) > \delta$, $\quad \left\lvert\frac{\hat{\boldsymbol{\rho}}_s[v]}{d(v)} - \frac{\boldsymbol{\rho}_s[v]}{d(v)}\right\rvert \le \epsilon_r\cdot\frac{\boldsymbol{\rho}_s[v]}{d(v)}$;
  - $\forall\, v \in G\ \ s.t.\ \boldsymbol{\rho}_s[v]/\mathrm{d}(v) \le \delta$, $\quad \left\lvert\frac{\hat{\boldsymbol{\rho}}_s[v]}{d(v)} - \frac{\boldsymbol{\rho}_s[v]}{d(v)}\right\rvert \le \epsilon_r\cdot\delta$.
- Monte-Carlo Random Walks
  - At k-th hop, stops with probability $\frac{\eta(k)}{\psi(k)}$; otherwise, jumps to a random neighbor. $\omega = \frac{2(1+\epsilon_r/3)\log(n/p_f)}{\epsilon_r^2\delta}$ random walks.
  - $\hat{\rho}_s[v]$= Fraction of random walks stopping at v. $\hat{\rho}_s[v]$ is a (d, ϵ_r, δ)-approximate HKPR vector.

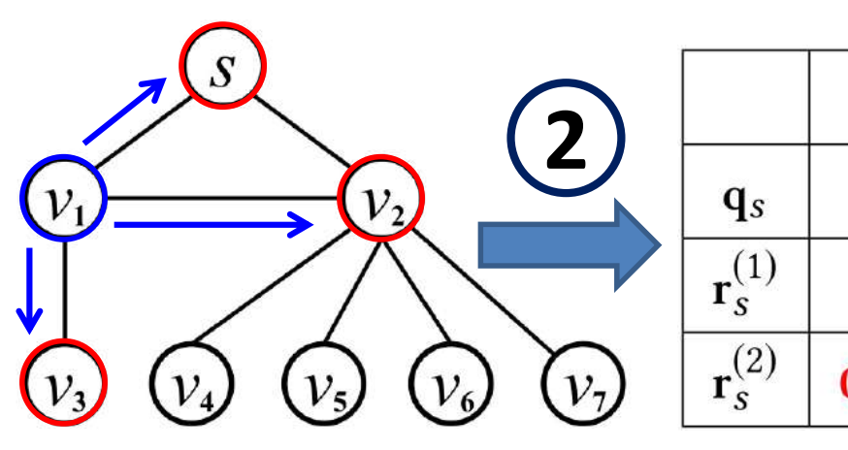


- HK-Push
  - $\boldsymbol{\rho}_s[v] = \underbrace{\boxed{\boldsymbol{q}_s[v]}}_{\text{estimation}} + \underbrace{\boxed{\sum_{u\in G}\sum_{k=0}^{K}\boldsymbol{r}_s^{(k)}[u]\cdot\boldsymbol{h}_u^{(k)}[v]}}_{\text{error!}}$
  - Each node v: a *reserve* $\boldsymbol{q}_s[v]$ and a k-hop residue $\boldsymbol{r}_s^{(k)}[v]$
  - Initially, sets $\boldsymbol{r}_s^{(0)}[s] = 1$; at k-th hop, converts $\frac{\eta(k)}{\psi(k)}\times\boldsymbol{r}_s^{(k)}[v] \rightarrow \boldsymbol{q}_s[v]$, *pushes* the remaining residue to neighbors evenly
  - $\hat{\rho}_s[v]$= Sum of reserves at v



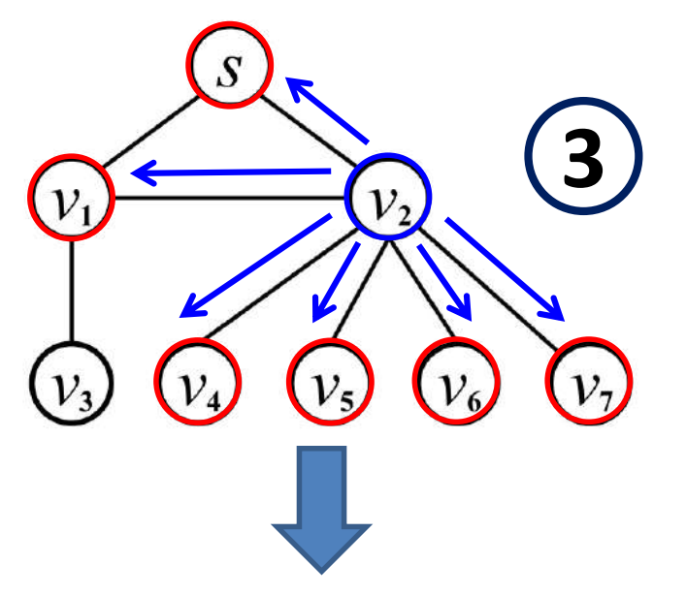
①

	s	v_1	v_2	v_3	v_4	v_5	v_6	v_7
q_s	0.05	0	0	0	0	0	0	0
$\mathrm{r}_s^{(1)}$	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
q_s	0.05	0.075	0	0	0	0	0	0
$\mathrm{r}_s^{(1)}$	0	0	0.475	0	0	0	0	0
$\mathrm{r}_s^{(2)}$	0.133	0	0.133	0.133	0	0	0	0

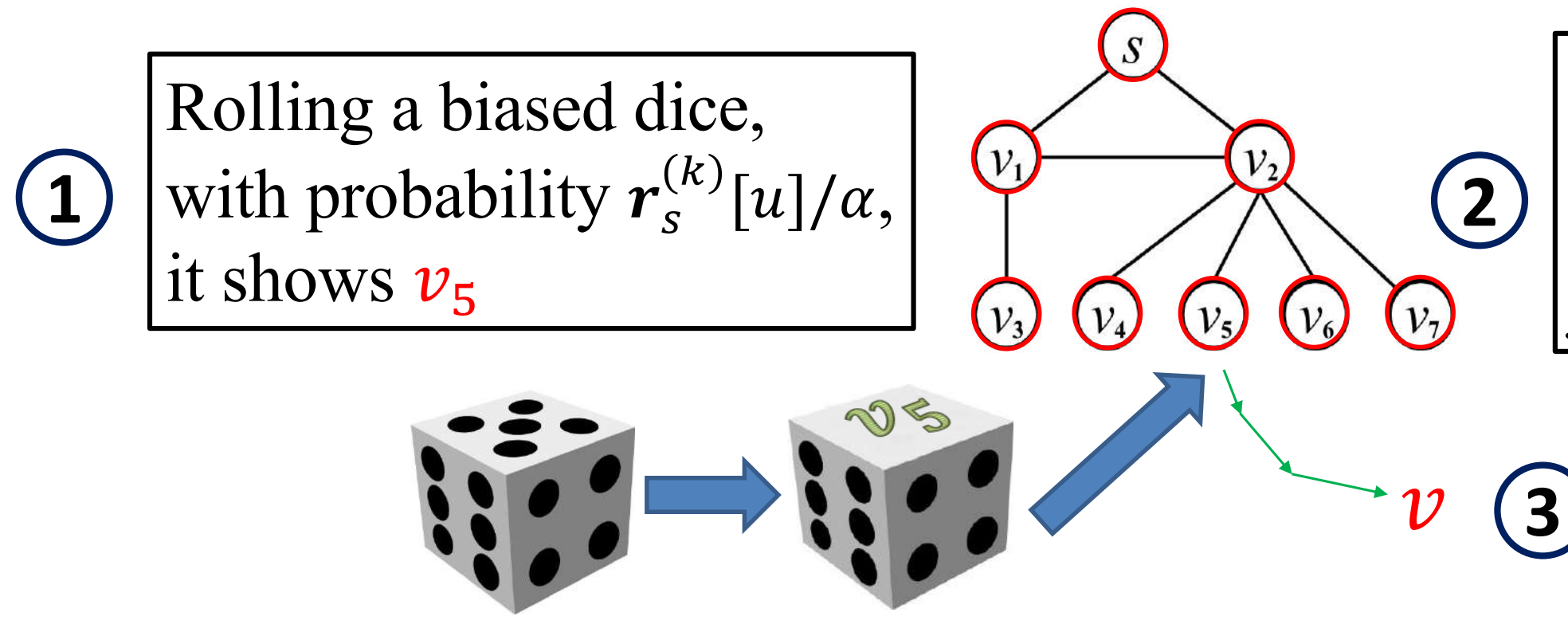


③

	s	v_1	v_2	v_3	v_4	v_5	v_6	v_7
q_s	0.05	0.075	0.075	0	0	0	0	0
$\mathrm{r}_s^{(1)}$	0	0	0	0	0	0	0	0
$\mathrm{r}_s^{(2)}$	0.2	0.067	0.133	0.133	0.067	0.067	0.067	0.067

4. The TEA+ Algorithm

- A combination of HK-Push & Random Walks
  - $\boldsymbol{\rho}_s[v] = \boldsymbol{q}_s[v] + \boxed{\sum_{u\in G}\sum_{k=0}^{K}\boldsymbol{r}_s^{(k)}[u]\cdot\boldsymbol{h}_u^{(k)}[v]} \Rightarrow \mathbb{E}[\mathrm{X}] = \frac{\#(\text{random walks ends at } v)}{\#(\text{all random walks})}$
  - $\boxed{\alpha = \sum_{u\in G}\sum_{k=0}^{K}\boldsymbol{r}_s^{(k)}[u]}$ Total portion of random walks that not stopped yet

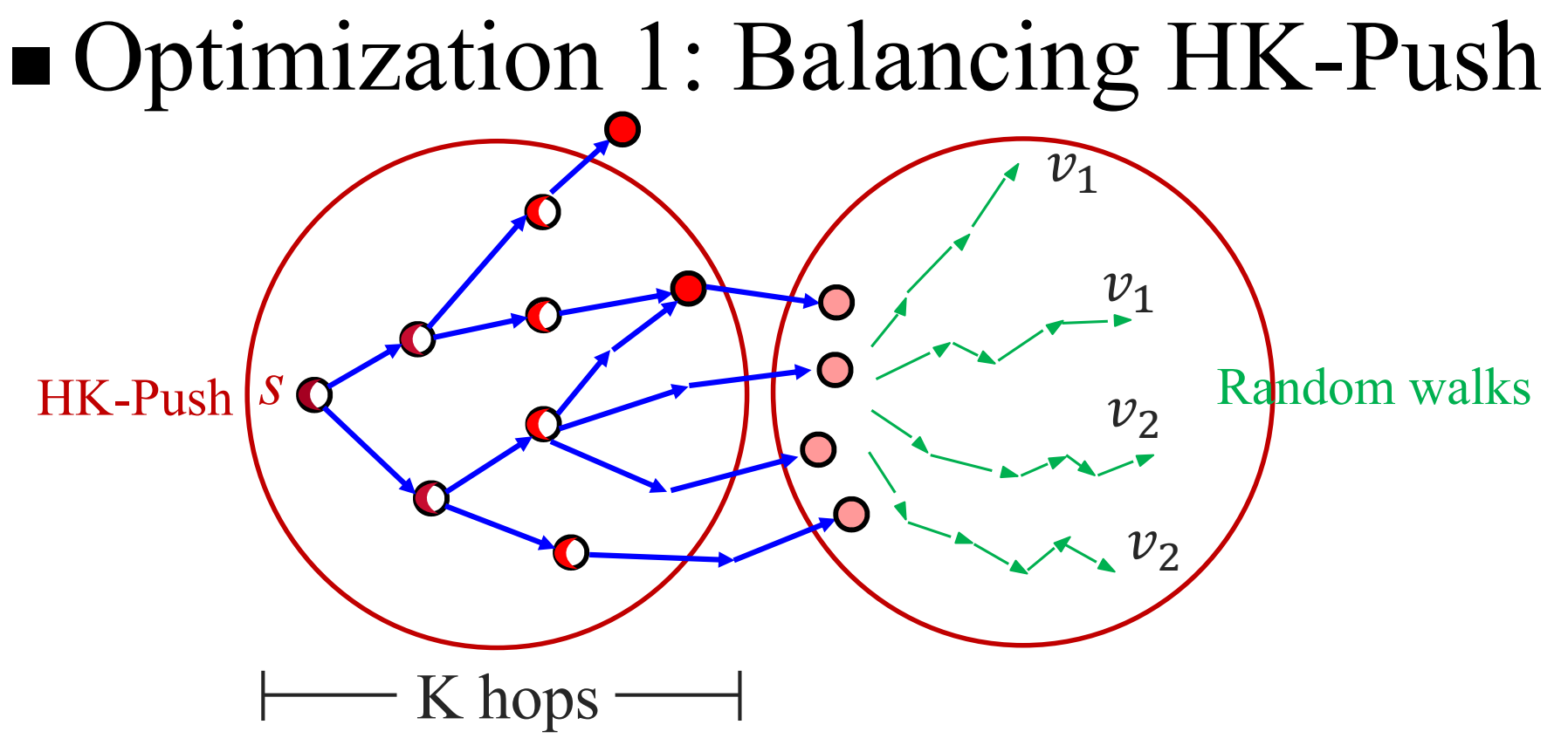


① Rolling a biased dice, with probability $\boldsymbol{r}_s^{(k)}[u]/\alpha$, it shows v_5

② 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 $\rightarrow O(\alpha\omega t)$ time
- Optimization 1: Balancing HK-Push and random walks



- Max #hops K: $\frac{1}{m/n}^{K/c} = \epsilon_r\cdot\delta$
- If cost(HK-Push) > 0.5 * cost(random walks), switch to random walks

- Optimization 2: Pruning random walks
  - If $\boldsymbol{r}_s^{(k)}[u]$ becomes 0, meaning it's accurate and no need for random walks
  - Otherwise, pruning $\beta_k\cdot\epsilon_r\delta\cdot d(u)$ portion of random walks

$$\boldsymbol{\rho}_s[v] = \boldsymbol{q}_s[v] + \boxed{\sum_{u\in G}\sum_{k=0}^{K}\boldsymbol{r}_s^{(k)}[u]\cdot\boldsymbol{h}_u^{(k)}[v]}$$

$$\boxed{\sum_{u\in G}\sum_{k=0}^{K}\boldsymbol{r}_s^{(k)}[u]\cdot\boldsymbol{h}_u^{(k)}[v]} \quad \boxed{\sum_{u\in G}\sum_{k=0}^{K}\boldsymbol{rb}_s^{(k)}[u]\cdot\boldsymbol{h}_u^{(k)}[v]}$$

$\max\{0, \boldsymbol{r}_s^{(k)}[u] - \beta_k\cdot\epsilon_r\delta\cdot d(u)\}$ $\quad$ $\min\{\boldsymbol{r}_s^{(k)}[u],\ \beta_k\cdot\epsilon_r\delta\cdot d(u)\}$

Estimated as $0.5\cdot\epsilon_r\delta\cdot d(u)$

$\in [0, \epsilon_r\delta\cdot d(v)]$

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

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

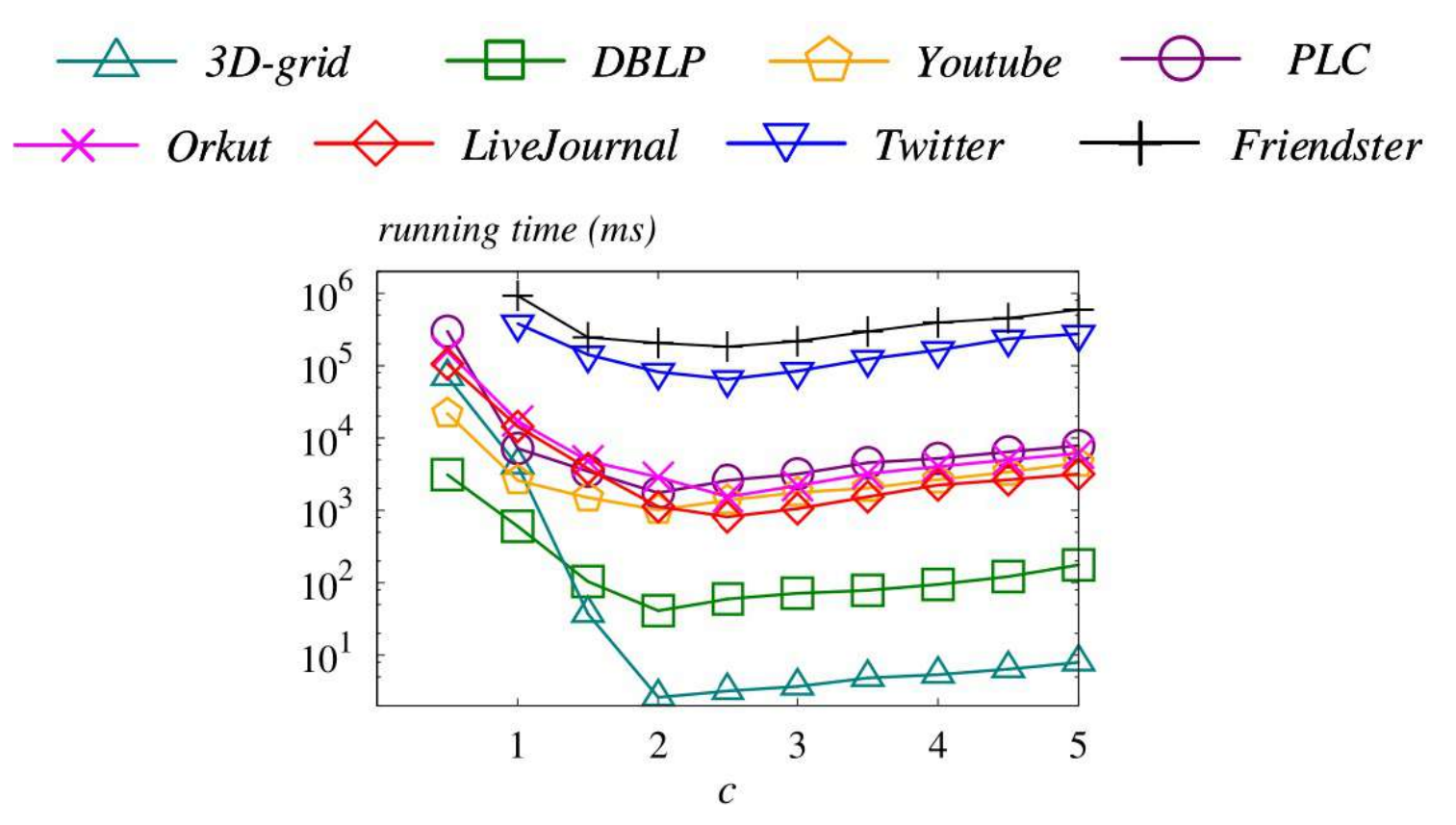


Figure 1: Running time of TEA+ vs c.

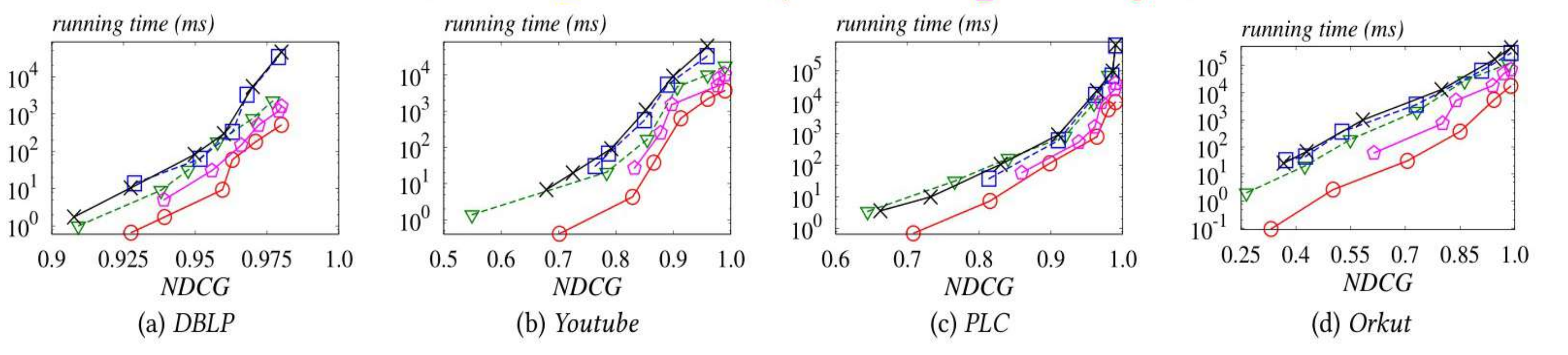


Figure 6: Running time vs. NDCG for computing normalized HKPR (best viewed in color).

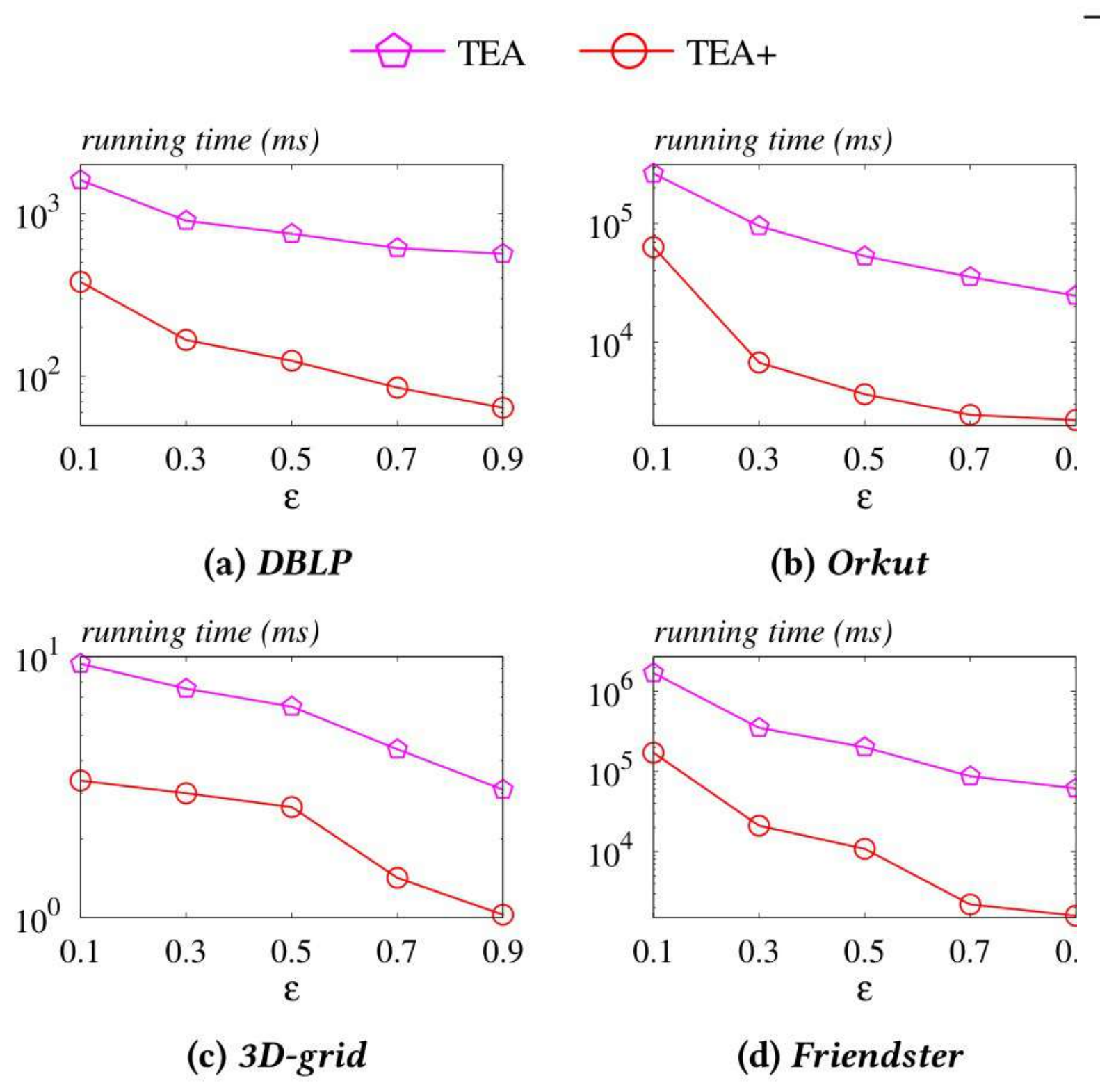


Figure 2: Running time vs ϵ_r.

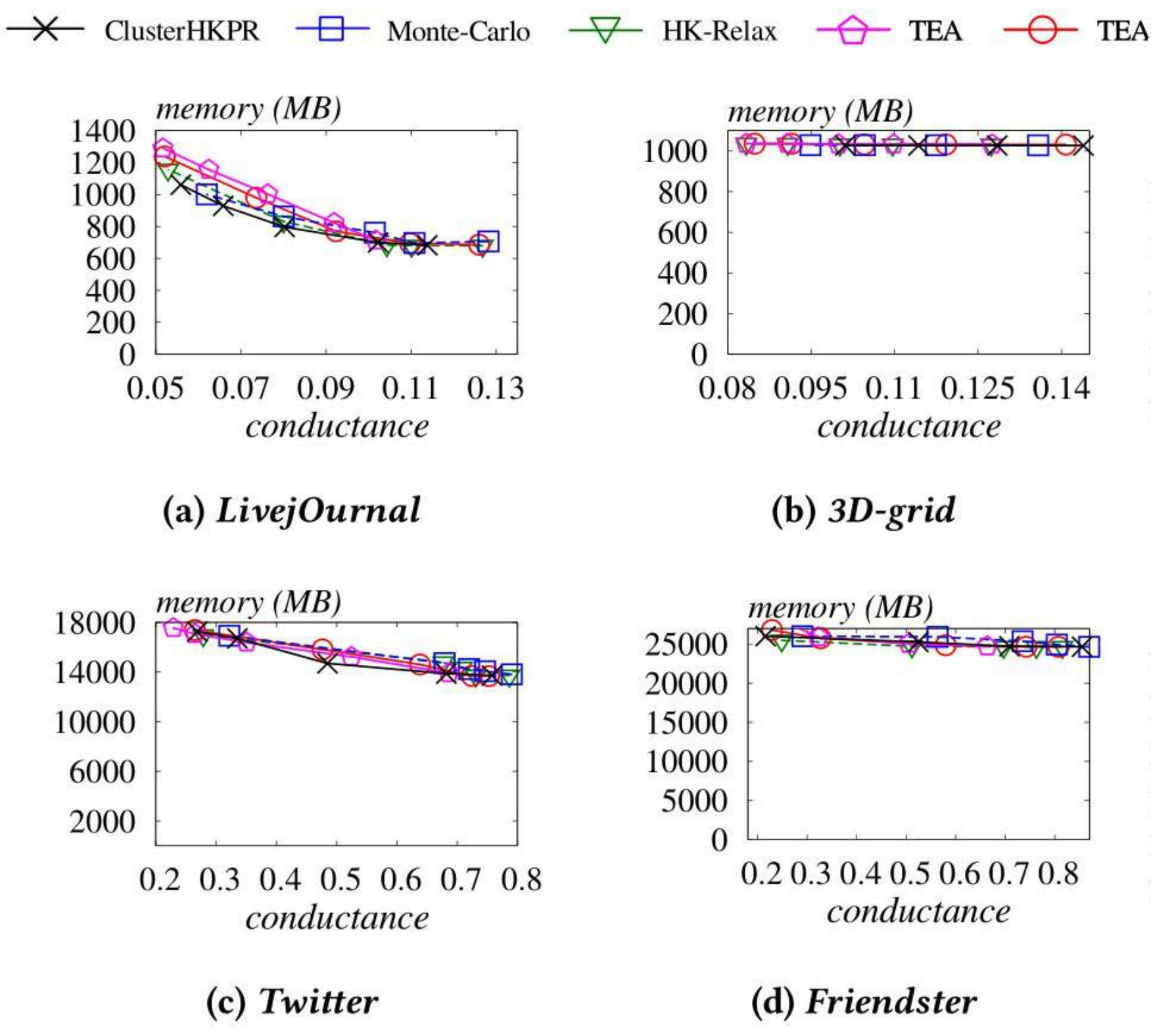


Figure 4: Memory cost vs. conductance.

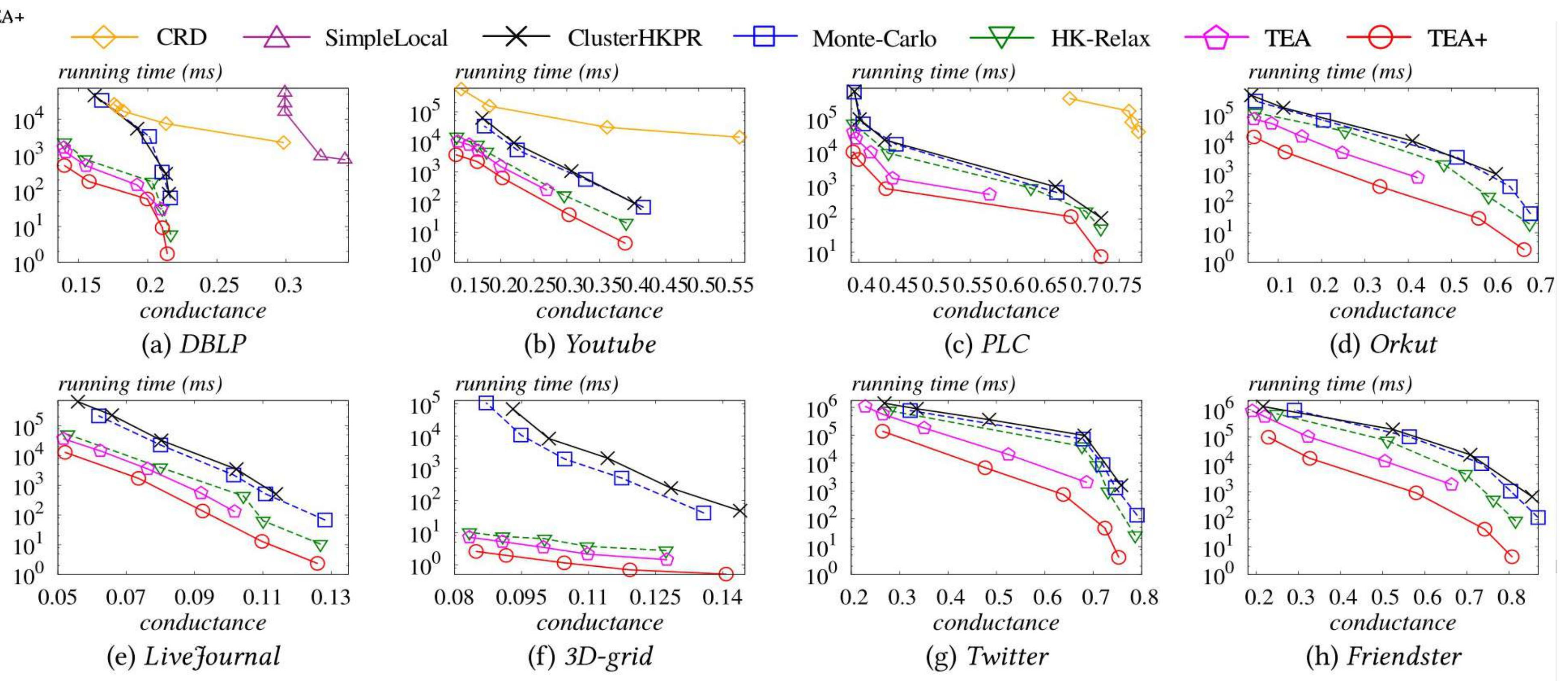


Figure 3: Running time vs conductance for local clustering queries (best viewed in color).