Efficient and Effective Similarity Search over Bipartite Graphs

Renchi Yang ▪ National University of Singapore

Outline

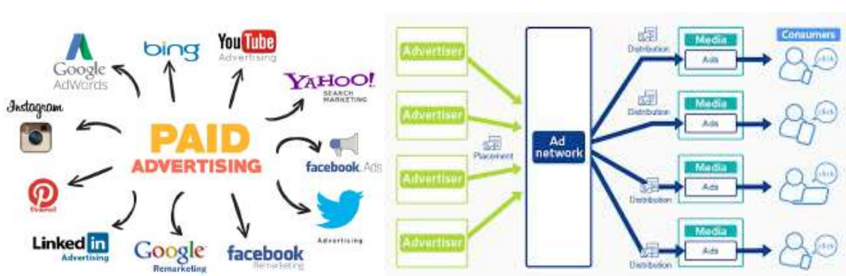
- Background
  - Problem Definition
  - Baseline Solutions and Challenges
- Proposed Solution Approx-BHPP
  - An Overview
  - Selective and Sequential Push
  - Power Iteration-based Push
- Experiments
  - Query Rewriting and Item Recommendation
  - Efficiency Evaluation

Background

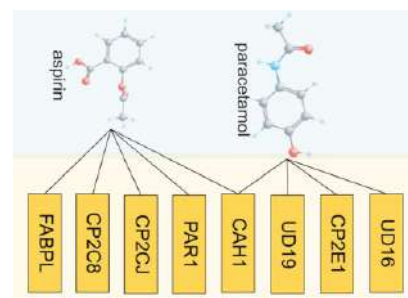
Similarity Search over Bipartite Graphs

Query Rewriting in Search Engine

Product Recommendation



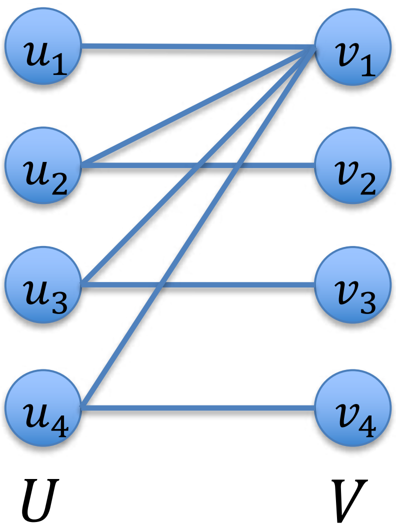
Online Advertising



Drug–target Prediction

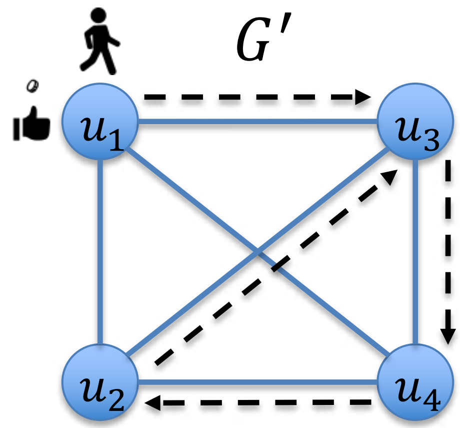
BHPP

- Hidden Personalized PageRank (HPP)



A bipartite graph G

$O(|U|^2)$ cost
in the worst case



- α probability to stop at current node
- $(1-\alpha)$ probability to jump to randomly jump to an out-neighbour

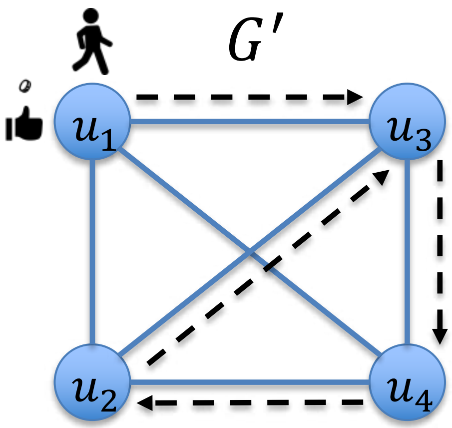
Steps: 0 1 2 3 4

$W(u_1)$: $u_1 \to u_3 \to u_4 \to u_2 \to u_3$

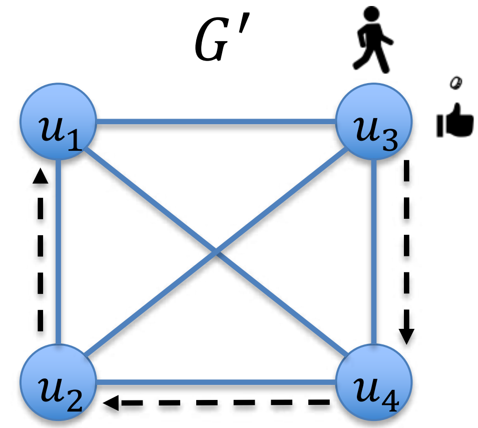
$$\pi(u_1, u_3) = \Pr[W(u_1) \text{ stops at } u_3]$$

BHPP

- Bidirectional Hidden Personalized PageRank



- α probability to stop at current node
- $(1-\alpha)$ probability to jump to randomly jump to an out-neighbour



Steps: 0 1 2 3 4

$W(u_1): u_1 \rightarrow u_3 \rightarrow u_4 \rightarrow u_2 \rightarrow u_3$

Steps: 0 1 2 3

$W(u_3): u_3 \rightarrow u_4 \rightarrow u_2 \rightarrow u_1$

A BHPP $\beta(u_1 + u_3) = \pi(u_1, u_3) + \pi(u_3, u_1)$ measures the similarity between nodes u_1 and u_3 from the perspectives of both

Problem Definition

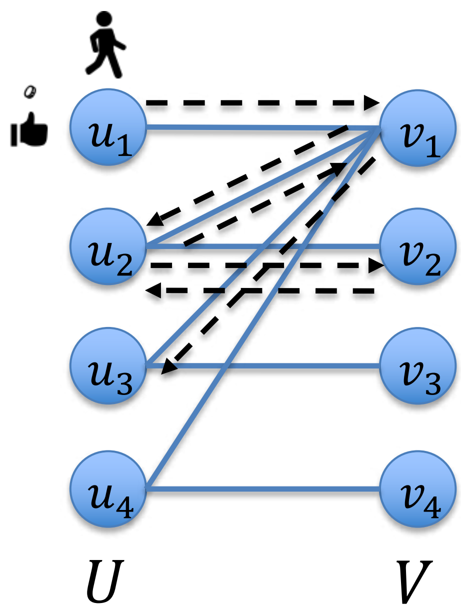
- ϵ-Approximate BHPP Query
- Input: A bipartite graph G with
  - 2 disjoint node sets U and V
  - a query node $u \in U$
  - an absolute error threshold ϵ

- Output: $\forall u_i \in U$, an approximate BHPP value $\beta'(u, u_i)$ such that

$$|\beta'(u, u_i) - \beta(u, u_i)| \leq \epsilon$$

Baseline Solutions

- Monte Carlo



A bipartite graph G

Steps: 0 1 2 3 4 5 6

$W(u_1): u_1 \to v_1 \to u_2 \to v_2 \to u_2 \to v_1 \to u_3$

- If current node $x \in U$
  - α probability to stop at current node
  - $(1-\alpha)$ probability to jump to randomly jump to an out-neighbour
- Otherwise
  - jump to randomly jump to an out-neighbour

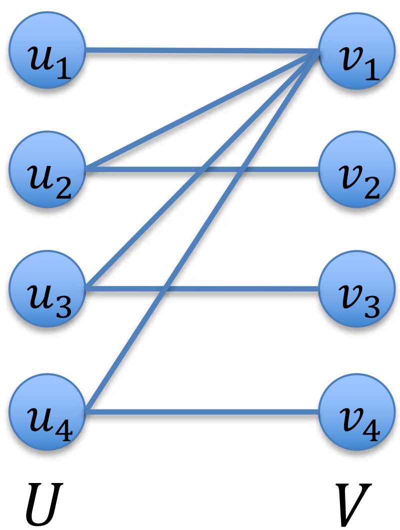
$$\pi_f(u_1, u_3) = \#\text{walks ending at } u_3 / \#\text{walks}$$

$$\left|\pi_f(u, u_i) - \pi(u, u_i)\right| \leq \epsilon \; \forall u_i \in U$$

Too many random walks

Baseline Solutions

- Power Iteration



A bipartite graph G

$$\begin{array}{c} \begin{array}{cccc} u_1 & u_2 & u_3 & u_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \end{array} \times \begin{array}{c} \begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline u_1 & 1 & 0 & 0 & 0 \\ u_2 & 0.5 & 0.5 & 0 & 0 \\ u_3 & 0.5 & 0 & 0.5 & 0 \\ u_4 & 0.5 & 0 & 0 & 0.5 \end{array} \\ \mathbf{U} \end{array} \times \begin{array}{c} \begin{array}{cccc|c} u_1 & u_2 & u_3 & u_4 & \\ \hline 1/4 & 1/4 & 1/4 & 1/4 & v_1 \\ 0 & 1 & 0 & 0 & v_2 \\ 0 & 0 & 1 & 0 & v_3 \\ 0 & 0 & 0 & 1 & v_4 \end{array} \\ \mathbf{V} \end{array}$$

$$\Downarrow$$

$$\begin{bmatrix} 1/4 & 1/4 & 1/4 & 1/4 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0.5 & 0 & 0 & 0.5 \end{bmatrix} \times \begin{bmatrix} 1/4 & 1/4 & 1/4 & 1/4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\vdots$$

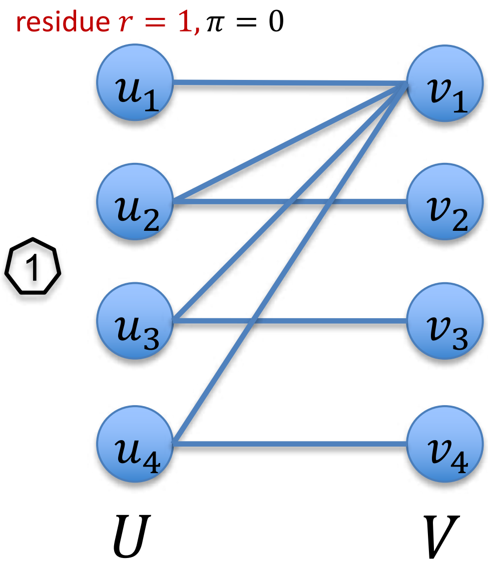
weighted sum

$$\left|\pi_f(u, u_i) - \pi(u, u_i)\right| \leq \epsilon \ \forall u_i \in U$$

Too many iterations

Baseline Solutions

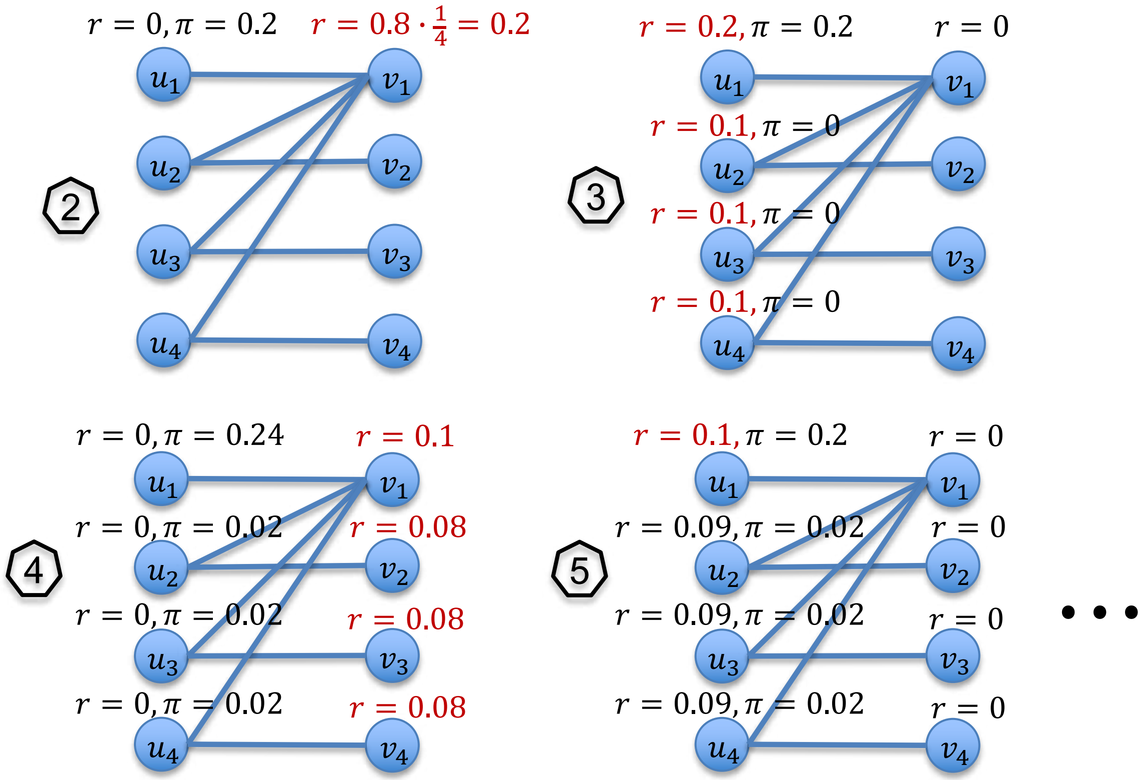
- Selective Push



A bipartite graph G

$\alpha = 0.2, \epsilon = 0.09$

until residue ≤ 0.09



$$|\pi_b(u_i, u) - \pi(u_i, u)| \leq \epsilon \; \forall u_i \in U$$

Baseline Solutions

- Monte Carlo + Selective Push (MCSP)
  - Random walks from u: $\left|\pi_f(u, u_i) - \pi(u, u_i)\right| \leq \epsilon/2 \; \forall u_i \in U$
  - Selective pushes from u: $|\pi_b(u_i, u) - \pi(u_i, u)| \leq \epsilon/2 \; \forall u_i \in U$
  - Let $\beta'(u, u_i) = \pi_f(u, u_i) + \pi_b(u_i, u)$ be approximate BHPP

- Power Iteration + Selective Push (PISP)
  - Power iterations from u: $\left|\pi_f(u, u_i) - \pi(u, u_i)\right| \leq \epsilon/2 \; \forall u_i \in U$
  - Selective pushes from u: $|\pi_b(u_i, u) - \pi(u_i, u)| \leq \epsilon/2 \; \forall u_i \in U$
  - Let $\beta'(u, u_i) = \pi_f(u, u_i) + \pi_b(u_i, u)$ be approximate BHPP

Challenges

- Monte Carlo
  - Too many random walks needed
  - Time complexity: $O(\frac{\log(|U|/\epsilon)}{\epsilon^2})$
- Power Iteration
  - Too many iterations of matrix-vector multiplications
  - Time complexity: $O(|E| \cdot \log(\frac{1}{\epsilon}))$
- Selective Push
  - Practically efficient except the cases
    - ϵ is very small
    - graphs have high average degrees
  - Time complexity: $O(|E| \cdot \frac{1}{\epsilon})$ in the worst case

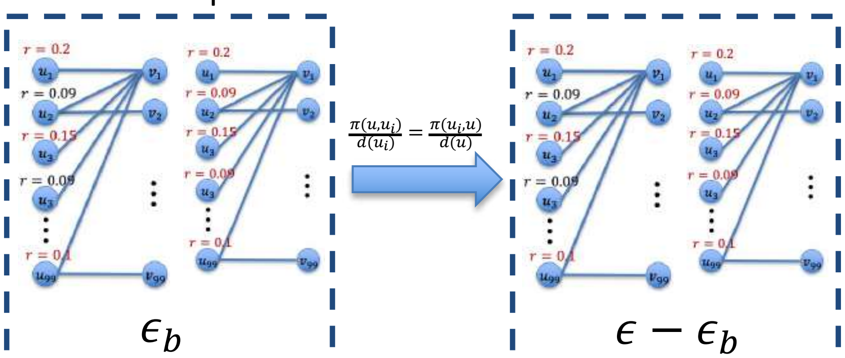
How?

Proposed Solution: An Overview

- A lemma: $\frac{\pi(u,u_i)}{d(u_i)} = \frac{\pi(u_i,u)}{d(u)}$, $d(u)$ is the degree of node u
  - Invoking Selective Push to compute $\pi_b(u_i, u)\ \forall u_i \in U$
  - No need to compute $\pi_f(u, u_i)\ \forall u_i \in U$ from scratch
- How to ensure accuracy guarantee & improve time complexity & retain practical efficiency? A combination approach

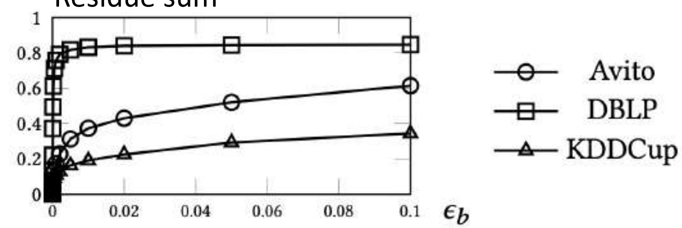
Selective and Sequential Push

Power Iteration-based Push



balance the costs

Residue sum



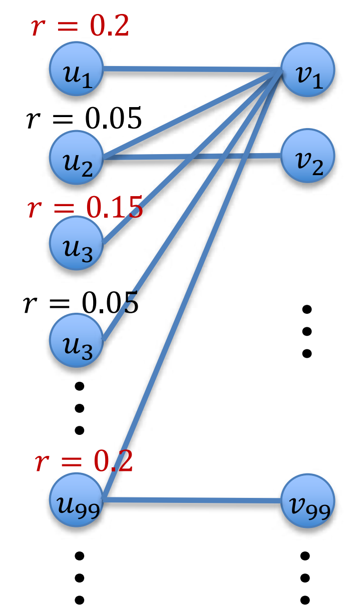
$$\epsilon_b = \frac{|E| - \sqrt{|U|\cdot|V|}}{2|E| - \sqrt{|U|\cdot|V|}} \cdot \epsilon$$

Proposed Solution

Selective and Sequential Push

- Drawbacks of the Selective Push
  - $u_2, u_4, \cdots$ are not selected here but will be selected in next round
    - More push operations are caused
      - More rounds of pushes needed for v_1
      - In each round, v_1 performs 99 pushes
    - Bad memory access patterns
      - Selecting nodes leads to random access to node list

$\boldsymbol{\alpha = 0.2, \epsilon_b = 0.06}$

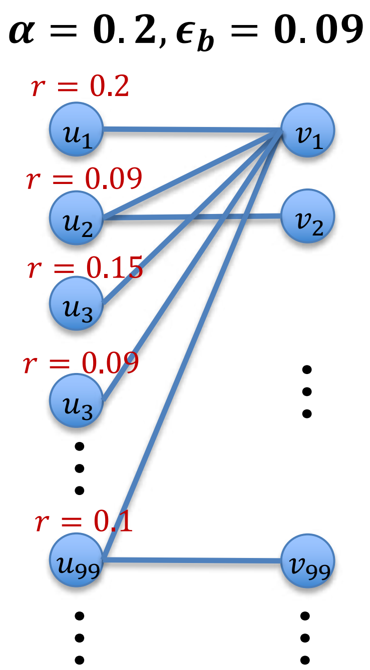


Proposed Solution

Selective and Sequential Push

- Solution:
  - If the #pushes conducted > the cost of power iterations
  - Switch to the sequential push, i.e., performing pushes from every node with a positive residue, until
    - every residue $\leq \epsilon_b$ or
    - the sum of residues $\leq \epsilon_b$
- Result:
  - Time complexity is bounded by

$$O(|E| \cdot \log(\frac{1}{\epsilon}))$$



Proposed Solution

Power Iteration-based Push

- A lemma: $\frac{\pi(u,u_i)}{d(u_i)} = \frac{\pi(u_i,u)}{d(u)}$, $d(u)$ is the degree of node u
  - No need to compute $\pi_f(u, u_i)\ \forall u_i \in U$ from scratch
- Steps:
  - Let $\pi_b(u_i, u), r(u_i)\ \forall u_i \in U$ be the output of the Selective and Sequential Push
  - Transform: $\pi_f(u, u_i) = \frac{d(u_i)}{d(u)} \cdot \pi_b(u_i, u)\ \forall u_i \in U$
  - Perform selective pushes until
    - every residue $r(u_i) \leq \frac{d(u_i)}{d(u)} \cdot \frac{\epsilon - \epsilon_b}{\lambda}$ or
    - the #pushes conducted > the cost of power iterations
      - switch to performing t power iterations
      - t is determined by $\epsilon - \epsilon_b$ and residues

$\Rightarrow O(|E| \cdot \log(\frac{1}{\epsilon}))$

Experiments

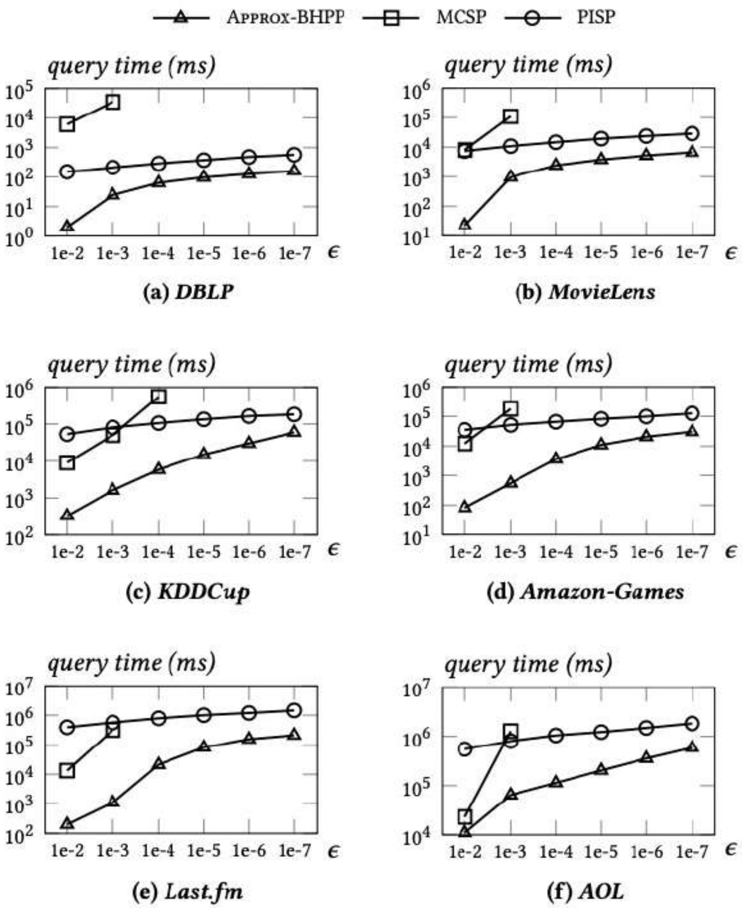
Table 1: Statistics of click graphs.

Name	$\|U\|$	$\|V\|$	$\|E\|$	#clicks	#impressions
Avito [6]	27,736	16,589	67,028	278,960	18,121,561
KDDCup [4]	255,170	1,848,114	2,766,393	8,217,633	121,232,353
AOL [2]	4,811,647	1,632,788	10,741,953	19,442,625	69,745,428,949

Table 2: Statistics of user-item graphs.

Name	$\|V\|$	$\|U\|$	$\|E\|$	weight
DBLP [26]	6,001	1,308	29,256	#papers
MovieLens [1]	6,040	3,706	1,000,209	ratings
Last.fm [3]	359,349	160,168	17,559,530	#plays
Amazon-Games [5]	826,767	50,210	1,324,753	ratings

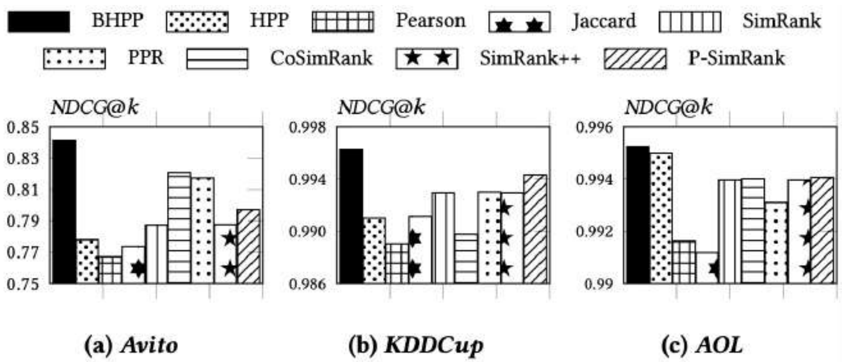
Query Efficiency



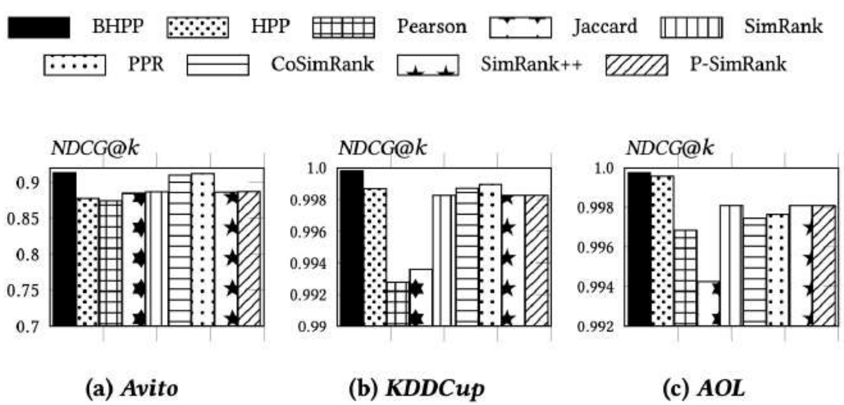
- MCSP=Monte Carlo + Selective Push
- PISP=Power Iteration + Selective Push
- $\alpha = 0.15, p_f = 10^{-6}$

- Result: ApproxBHPP outperforms all competitors, often by an order of magnitude

Query Rewriting



$k = 10$



$k = 5$

- Setup:
  - 20% edges removed
  - evaluate the top-k ordering of queries via NDCG
- Result
  - BHPP consistently outperforms other similarity measures
  - on Avito, at least 2% over state-of-the-art results

Item Recommendation

$k = 10$

Similarity	DBLP		Movielens		Last.fm		Amazon-Games	
	precision@k	*recall@k*	*precision@k*	*recall@k*	*precision@k*	*recall@k*	*precision@k*	*recall@k*
BHPP	**0.167**	**0.164**	**0.405**	**0.289**	**0.313**	**0.231**	**0.248**	**0.187**
HPP	0.14	0.138	0.224	0.161	0.305	0.223	0.194	0.15
Pearson	0.037	0.037	0.106	0.074	0.126	0.095	0.056	0.044
Jaccard	0.158	0.157	0.272	0.194	0.287	0.213	0.08	0.062
SimRank	0.151	0.15	0.245	0.177	0.239	0.169	0.127	0.084
CoSimRank	0.115	0.113	0.186	0.137	0.304	0.216	0.156	0.121
PPR	0.149	0.146	0.342	0.245	0.28	0.206	0.188	0.143
SimRank++	0.127	0.126	0.243	0.176	0.241	0.171	0.171	0.118
P-SimRank	0.127	0.127	0.221	0.164	0.226	0.159	0.14	0.088

$k = 5$

Similarity	DBLP		Movielens		Last.fm		Amazon-Games	
	precision@k	*recall@k*	*precision@k*	*recall@k*	*precision@k*	*recall@k*	*precision@k*	*recall@k*
BHPP	**0.165**	**0.115**	**0.609**	**0.22**	**0.441**	**0.163**	**0.36**	**0.136**
HPP	0.15	0.097	0.291	0.105	0.416	0.15	0.28	0.108
Pearson	0.095	0.064	0.091	0.031	0.178	0.067	0.104	0.039
Jaccard	0.139	0.095	0.322	0.114	0.307	0.093	0.112	0.041
SimRank	0.157	0.109	0.325	0.118	0.356	0.112	0.209	0.088
CoSimRank	0.152	0.102	0.322	0.108	0.415	0.152	0.243	0.098
PPR	0.127	0.098	0.475	0.17	0.393	0.145	0.272	0.104
SimRank++	0.15	0.101	0.325	0.118	0.367	0.12	0.277	0.103
P-SimRank	0.15	0.1	0.32	0.112	0.343	0.108	0.226	0.094

- Remove 20% edges and evaluate top-k recommendation performance via $precision@k$ and $recall@k$
- BHPP consistently yields the best performance

Thanks

Efficient and Effective Similarity Search over Bipartite Graphs