ISWC 2021 Research Track, October 24-28, Virtual Conference

Fast ObjectRank for Large Knowledge Databases

Hiroaki Shiokawa

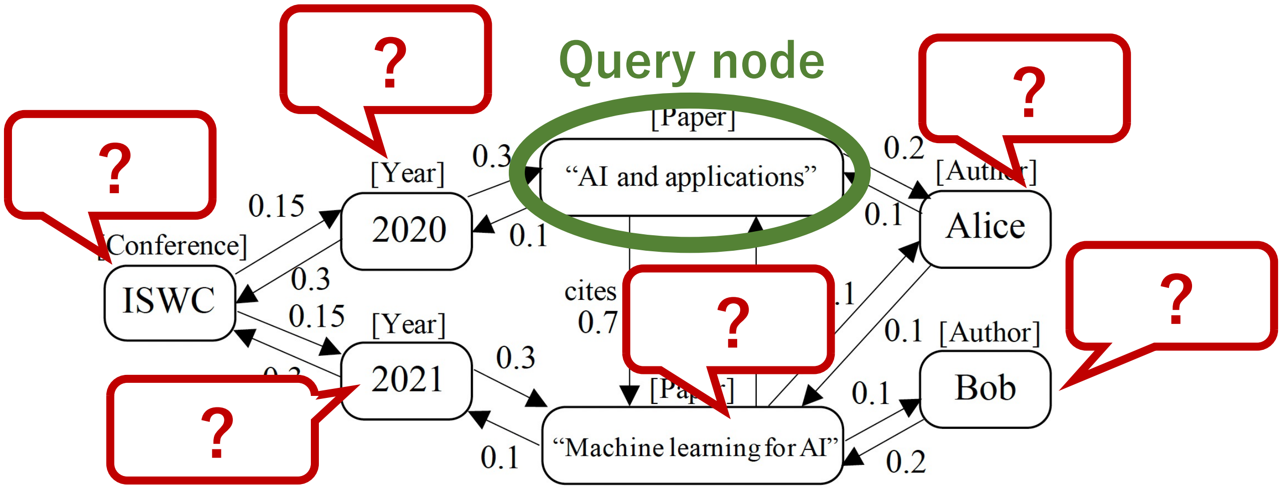
Database Group at Center for Computational Sciences,
University of Tsukuba, Japan

筑波大学
University of Tsukuba

筑波大学
計算科学研究センター
Center for Computational Sciences

ObjectRank [Hiristidis, Hwang and Papakonstantinou, 2008]

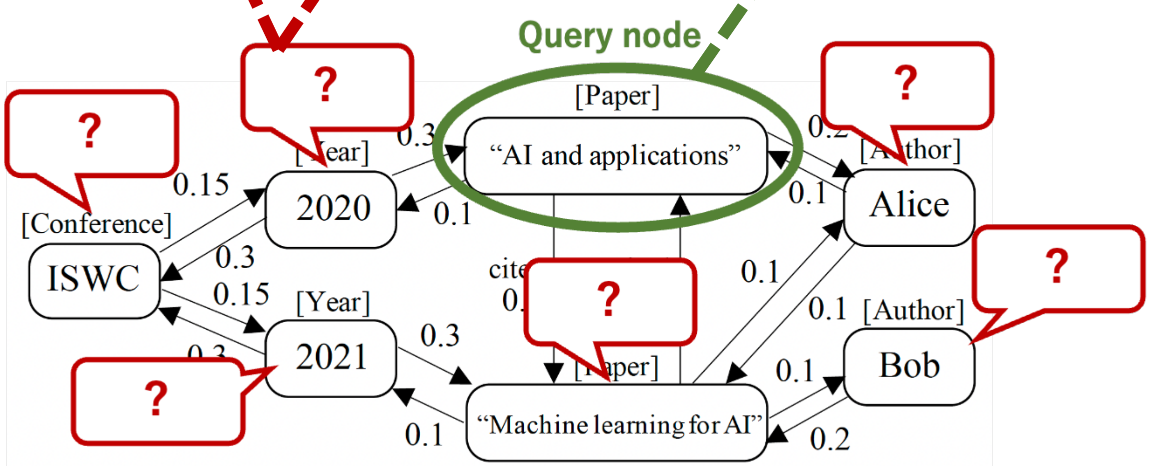
- **Relevant entity search algorithm for KGs**
  - **Given:** a KG $G_D(V_D, E_D, W_D)$, and a query nodes $\boldsymbol{V_q} \subseteq \boldsymbol{V_D}$
  - **Return:** an importance vector $\mathbf{r} = (\boldsymbol{r_1}, \boldsymbol{r_2}, \ldots, \boldsymbol{r_{|V_D|}})^{\mathbf{T}}$, where r_i denotes a relevance between $v_i \in V_D$ and V_q



Importance Computation

- **Iterative Random-walk with Restart (RWR)**
  - ObjectRank iteratively runs the following a matrix-vector multiplication until $\mathbf{r}$ converges.

$$\mathbf{r} = \alpha \mathbf{A}\mathbf{r} + (1-\alpha)\mathbf{q}$$



Efficiency Problem of ObjectRank

- **Iterative RWR incurs expensive costs**
  - ObjectRank needs to update the importance vector for all entities (nodes) included in V_D until convergence.
  - It incurs $\boldsymbol{O}((|\boldsymbol{V_D}| + |\boldsymbol{E_D}|)\boldsymbol{t_D})$ time, where $|V_D|, |E_D|$ and t_D are # of nodes and edges in a KG, and # of iterations, respectively.

- **Recent Semantic Web applications**
  - KGs are becoming larger and larger…
  - In many cases, we need to handle at least $|V_D| \geq 10^6$ entities.

Can ObjectRank handle such massive KGs?

Goal & Contributions

How can we efficiently compute ObjectRank without sacrificing the top-k search quality?

- **Proposed Method:** ***SchemaRank***
  - Schema-aware top-k search algorithm for fast/exact ObjectRank.
- **Contributions**
  - **Efficient:** Up to 644.7 times faster than SOTAs.
  - **Exact:** Guarantees the same results as those of ObjectRank.
  - **Easy to deploy:** Requires no user-specified parameters.
  - **Codes are available:**

https://github.com/LazyShion/SchemaRank

Key Observation

- **The skewness in the importance distribution**
  - Real graphs have **highly skewed importance distribution**.
  - The vast majority of nodes practically yield low importance.

Question: Can we find such nodes with low importance **before** the importance evaluation?

- **ShemaRank: Schema-aware two-phase RWRs**

① Coarse-grained RWR

It estimates the importance of node-types from a schema of KGs.

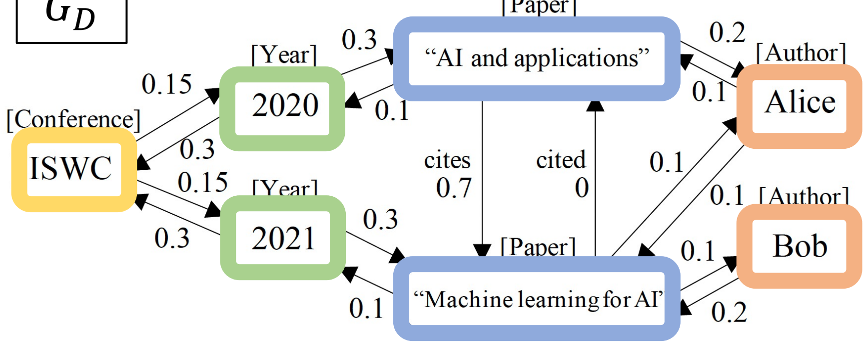
② Coarse-grained RWR

It incrementally prunes unpromising nodes to find top-k nodes using the importance of node-types.

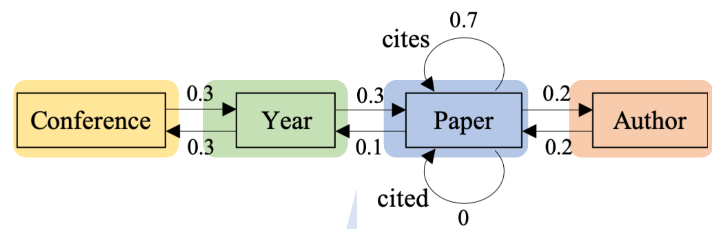
Coarse-grained RWR (CR)

- **Schema-level importance estimation**
  - Finds node-types that yield low importance score from a schema.

G_D



G_S(a schema graph of G_D)

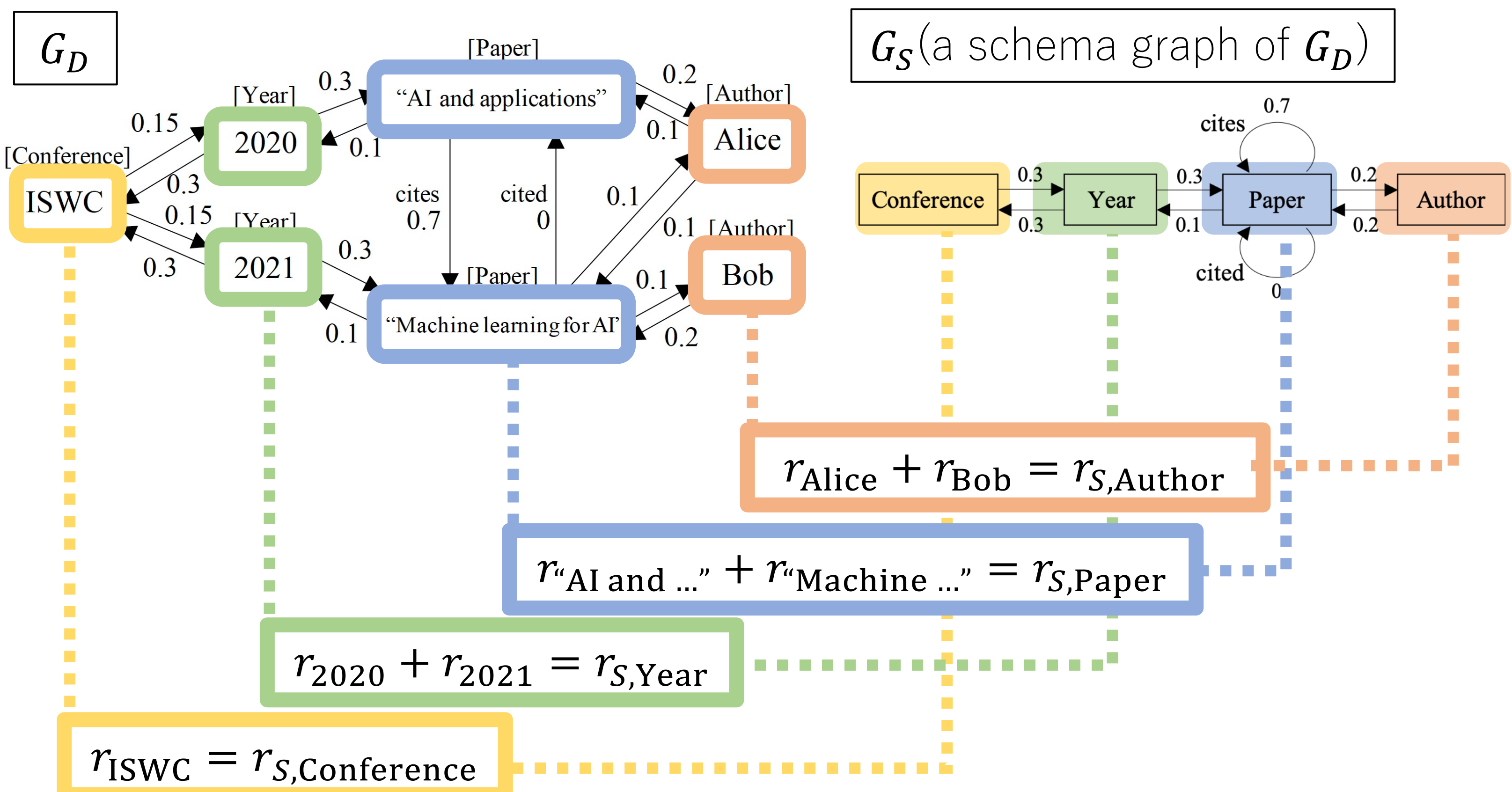


$$\mathbf{r}_S = \alpha \mathbf{A}_S \mathbf{r}_S + (1 - \alpha)\mathbf{q}_S$$

$\mathbf{r}_S$ and $\mathbf{q}_S$ are the vectors projected from G_D to G_S.
$\mathbf{A}_S$ is the adjacency matrix of G_S.

Important Property of CR

- $\mathbf{r}_S$ is a good approximation of $\mathbf{r}$



Fine-grained RWR (FR) (1/2)

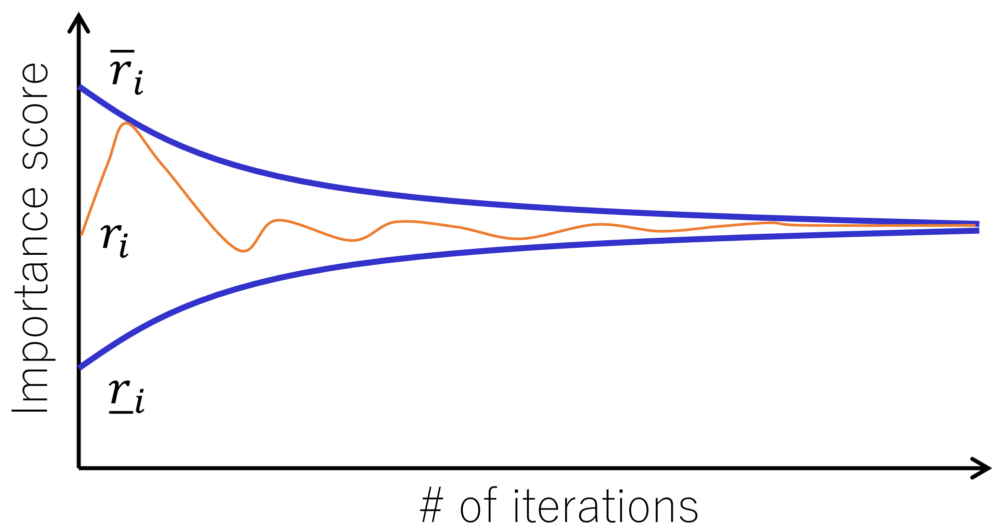
- **Explores top-k important nodes on a KG using $\mathbf{r}_S$**
  - SchemaRank derives the following bounds from $\mathbf{r}_S$.

Lower bound of r_i

$$\underline{r}_i^{(t)} = \begin{cases} (1-\alpha)q_i & (t=0) \\ \underline{r}_i^{(t-1)} + (1-\alpha)\alpha^t p_i^{(t)} & (t>0) \end{cases},$$

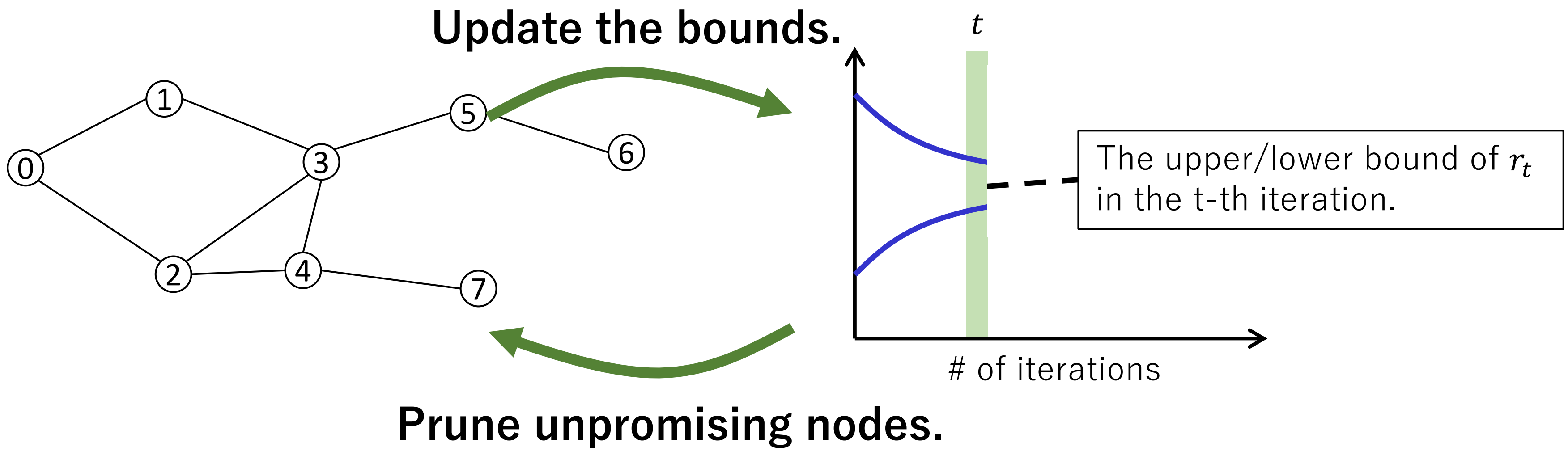
Upper bound of r_i

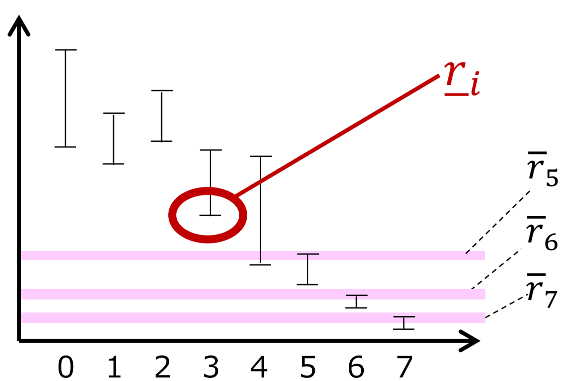
$$\overline{r}_i^{(t)} = \begin{cases} b_i^{(0)} + \frac{\alpha}{1-\alpha}\overline{A}_i & (t=0) \\ \underline{r}_i^{(t-1)} + \alpha^t b_i^{(t)} + \frac{\alpha^{t+1}}{1-\alpha}\Delta^{(t)}\overline{A}_i & (t>0) \end{cases},$$



$$\underline{r}_i^{(\infty)} = \overline{r}_i^{(\infty)} = r_i$$

Fine-grained RWR (FR) (2/2)





Prune nodes whose upper bound is smaller than the k-th largest lower bound.

Runtime Efficiency

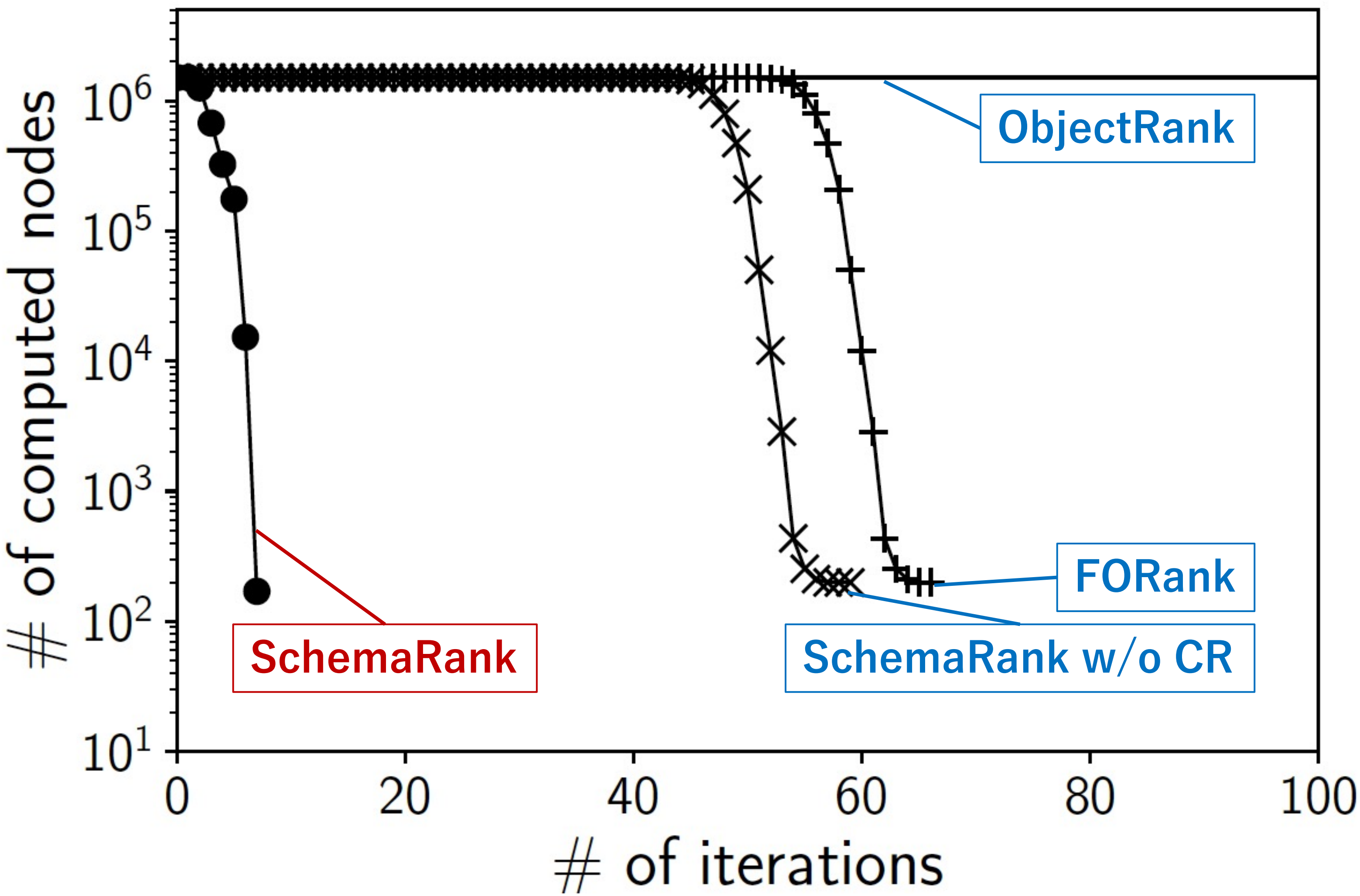
(b) DBLP (small)

Methods	Skewed	Uniform	Pre-comp.
SchemaRank ($k=10^2$)	**1.91** (±0.003) **sec.**	**2.54** (±0.003) **sec.**	—
SchemaRank ($k=10^3$)	**2.08** (±0.002) **sec.**	**2.63** (±0.003) **sec.**	—
ObjectRank	22.8 (±0.011) sec.	22.7 (±0.009) sec.	—
BinRank	6.42 (±0.009) sec.	7.22 (±0.011) sec.	17.9 hours
LORank	16.4 (±0.008) sec.	17.2 (±0.008) sec.	—
SimMat ($k=10^2$)	N/A	N/A	$>$24 hours
SimMat ($k=10^3$)	N/A	N/A	$>$24 hours
FORank ($k=10^2$)	6.77 (±0.002) sec.	7.45 (±0.003) sec.	—
FORank ($k=10^3$)	8.68 (±0.003) sec.	9.03 (±0.004) sec.	—

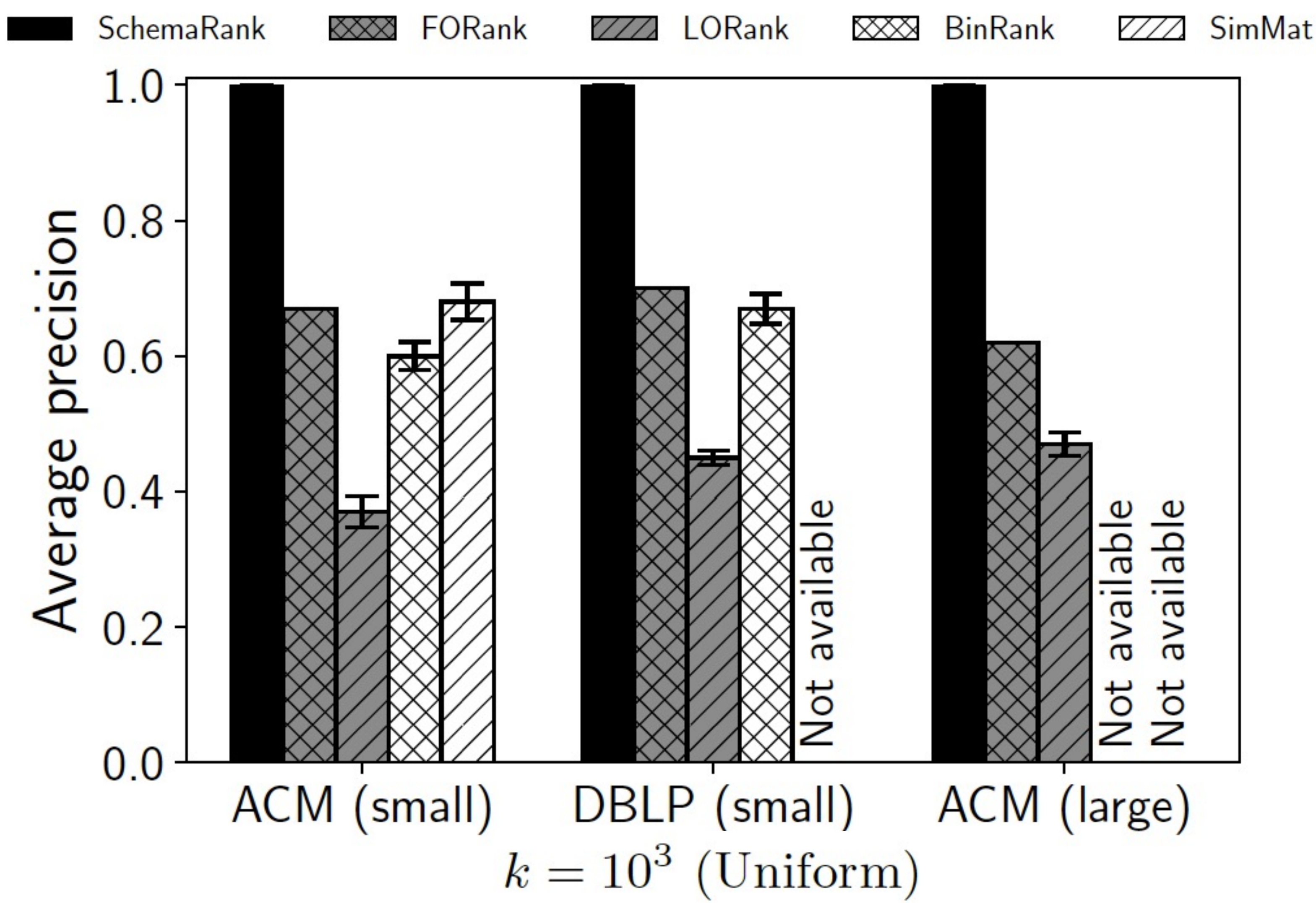
(d) DBLP (large)

Methods	Skewed	Uniform	Pre-comp.
SchemaRank ($k=10^2$)	**134** (±0.195) **sec.**	**169** (±0.106) **sec.**	—
SchemaRank ($k=10^3$)	**168** (±0.173) **sec.**	**241** (±0.122) **sec.**	—
ObjectRank	$>$24 hours	$>$24 hours	—
BinRank	N/A	N/A	$>$24 hours
LORank	$>$24 hours	$>$24 hours	—
SimMat ($k=10^2$)	N/A	N/A	$>$24 hours
SimMat ($k=10^3$)	N/A	N/A	$>$24 hours
FORank ($k=10^2$)	2,209 (±1.908) sec.	2,677 (±1.966) sec.	—
FORank ($k=10^3$)	2,431 (±1.912) sec.	3,137 (±1.903) sec.	—

How CR Effectively Works?



Average Precision (Top-K)



Summary

- **Research Question**
  - How can we efficiently compute ObjectRank without sacrificing the top-k search quality on large knowledge databases?
- **Proposed method: *SchemaRank***
  - Schema-aware top-k search algorithm for fast/exact ObjectRank.
- **Contributions**
  - **Efficient:** Up to 644.7 times faster than SOTAs.
  - **Exact:** Guarantees the same results as those of ObjectRank.
  - **Easy to deploy:** Requires no user-specified parameters.
  - **Codes are available:**

https://github.com/LazyShion/SchemaRank