Fast ObjectRank for Large Knowledge Databases

Hiroaki Shiokawa(✉)

Center for Computational Sciences, University of Tsukuba, Tsukuba, Japan
shiokawa@cs.tsukuba.ac.jp

Abstract. ObjectRank is an essential tool to evaluate an importance of nodes for a user-specified query in heterogeneous graphs. However, existing methods are not applicable to massive graphs because they iteratively compute all nodes and edges. This paper proposes SchemaRank, which detects the exact top-k important nodes for a given query within a short running time. SchemaRank dynamically excludes unpromising nodes and edges, ensuring that it detects the same top-k important nodes as ObjectRank. Our extensive evaluations demonstrate that the running time of SchemaRank outperforms existing methods by up to two orders of magnitude.

Keywords: Heterogeneous graph · Search algorithm · Big data

1 Introduction

ObjectRank [9] is an essential tool to analyze heterogeneous graphs composed of *multiple node-types*. It evaluates an importance of nodes in a graph for a user-specified query by performing *random walk with restarts* (RWR). Unlike traditional graph similarities [26], ObjectRank captures not only (1) the structural closeness but also (2) the relationships among multiple node-types in the graph. Because heterogeneous graphs are currently the most common data model to represent various knowledge resources (*e.g.,* knowledge graphs), ObjectRank is employed in various Semantic Web applications due to its effectiveness.

For instance, ObjectRank plays important roles in question-answering tasks [2] and representation learning tasks [4]. To improve the accuracy, these tasks must capture the relationships among data entities in their learning models (*e.g.,* deep neural networks). However, understanding how strongly entities affect each other is not a trivial task since various data resources such as QA sites and knowledge bases are formed as heterogeneous graphs [16,27]. To address this issue, recent approaches have employed the random-walk analysis such as ObjectRank [13]. By evaluating the importance with respect to query entities, these approaches successfully capture representative relationships to improve their models.

Similarly, ObjectRank is also applicable for Semantic Web search [3]. For example, Serene [3] applied ObjectRank to ranking entities of knowledge graphs

A. Hotho et al. (Eds.): ISWC 2021, LNCS 12922, pp. 217–234, 2021.
https://doi.org/10.1007/978-3-030-88361-4_13

based on onthology by using heuristic weight assignments. It first converts user-specified query-keywords into query nodes of ObjectRank by picking up nodes having a subset of the keywords. Then, Serene performs ObjectRank and returns top-k entities to users, each of which is likely relevant to the query-keywords. Analogously, ObjectRank can be applicable in other recent applications such as recommendation [12], sentiment analysis [30], and bioinformatics [29].

Although it is effective in many applications, ObjectRank is computationally expensive because all nodes and edges are computed iteratively. If $|V|$ and $|E|$ are the number of nodes and edges, respectively, ObjectRank requires $O((|V|+|E|)t)$ time, where t is the number of iterations. In the mid-2000s, ObjectRank was applied to small graphs such as user query logs, which had a few thousand nodes at most. By contrast, recent Semantic Web applications must handle large knowledge graphs with a few million nodes [15,21–23]. That is, the applications suffer from a long computation time due to the expensive costs of ObjectRank.

1.1 Existing Works and Challenges

The expensive cost in ObjectRank has led to various efficient approaches. *Indexing methods* are the most successful to date [1,10]. Instead of just-in-time importance computations, these methods pre-compute the node-importance for several query patterns. After receiving a query, the methods approximately compute the node-importance utilizing the pre-computed results according to the query patterns. By employing indexing approaches, the expensive cost is successfully moderated. However, [20] recently pointed out that indexing methods require a long runtime for pre-computation. For instance, in our evaluations, the methods required more than 17 h for pre-computing 1.5 million nodes.

Recently, *pruning methods* have become popular to reduce the large runtime for various RWR-based algorithms [5,7]. The main idea is to remove nodes with a low importance during iterative computations for a given query. For example, Fujiwara *et al.* proposed a fast top-k algorithm, *SimMat* [7], for SimRank [11]. SimMat reduces the graph by pruning nodes with a low importance through SVD. Hence, it can efficiently find top-k important nodes for a given query.

However, SimMat and its variants [5,6] are not applicable to ObjectRank for two reasons. First, they require high costs to estimate nodes with a low importance because their SVD-based method incurs $\Omega(|V|^3)$ time [8]. Second, their pruning approaches degrade the accuracy of ObjectRank. Since they are designed for homogeneous graphs, they do not guarantee the accuracy of the node-importance on heterogeneous graphs. For example, Sato *et al.* proposed a fast ObjectRank algorithm called *FORank* [20] by extending the pruning methods [6]. However, as shown in [20], FORank cannot find the same top-k nodes as ObjectRank. Thus, a fast and exact algorithm for ObjectRank remains elusive.

1.2 Our Approaches and Contributions

We present a novel fast ObjectRank algorithm called *SchemaRank*. Given a user-specified query, SchemaRank efficiently detects the same top-k important nodes

as ObjectRank. SchemaRank is based on the property of real-world graphs in which the distribution of node-importance is highly skewed [24]. That is, the vast majority of nodes have a low importance in real-world graphs.

Based on the above property, SchemaRank dynamically excludes unpromising nodes that cannot become the top-k important nodes. To determine which nodes to exclude, SchemaRank employs the following two-step RWRs: First, *coarse-grained RWR* estimates the distribution of importance at the node-type level. If specific node-types yield a low importance, SchemaRank prunes all nodes with those node-types in the second step. Second, *fine-grained RWR* prunes unpromising nodes by incrementally refining the estimated distribution at the node level. Once a node outputs a lower importance than the k-th highest node-importance, SchemaRank incrementally removes the node from the graph. This leads to the following characteristics:

- **Fast:** SchemaRank is faster than the state-of-the-art algorithms proposed in the last few years (Sect. 4.1). In our experiments, the running time of SchemaRank outperforms them by up to two orders of magnitude (Table 3).
- **Exact:** We theoretically guarantee that SchemaRank always outputs the same top-k nodes as ObjectRank (Theorem 2), although it prunes nodes with a low importance. We experimentally verified that it returns the same top-k ranking importance as ObjectRank (Fig. 4).
- **Easy to deploy:** SchemaRank does not require pre-computations (Algorithm 1); given a query, it quickly outputs top-k nodes on the fly.

SchemaRank is the first solution that achieves a fast and exact top-k importance evaluation on massive heterogeneous graphs. For instance, SchemaRank returns the exact top-k nodes within three seconds on a DBLP graph with 1.5 million nodes. Although ObjectRank effectively enhances the quality of Semantic Web applications, it is difficult to apply to massive graphs. By providing our fast and exact algorithm, SchemaRank will enhance many applications.

2 Preliminary

Here, we briefly introduce the background. Table 1 lists the main symbols and their definitions.

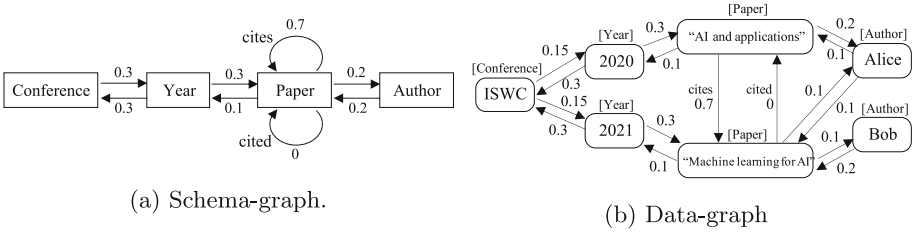
2.1 Data Model

ObjectRank transforms the given data entities into a heterogeneous graph for importance evaluation [9]. The generated graph is two-fold: *schema-graph* and *data-graph*.

Schema-Graph: A schema-graph is a *user-specified* graph that defines the relationships among entity-types (node-types). We denote a schema-graph as $G_S = (V_S, E_S, W_S)$, where V_S, E_S, and W_S are sets of nodes, edges, and edge-weights, respectively. V_S represents node-types, and E_S models their relationships. For each edge $e \in E_S$, an edge-weight $w_S(e) \in W_S$ must be defined such that $w_S(e) \in [0, 1]$ to emphasize the relevance between two node-types.

Table 1. Definitions of symbols.

Symbol	Definition
G_S	Schema-graph
V_S	Set of nodes in a schema-graph G_S
E_S	Set of edges in a schema-graph G_S
W_S	Set of edge-weights in a schema-graph G_S
$w_S(e)$	Edge-weight of $e \in E_S$.
G_D	Data-graph
V_D	Set of nodes in a data-graph G_D
$V_D(i)$	Set of nodes having a node-type i in V_D
E_D	Set of edges in a data-graph G_D
W_D	Set of edge-weights in a data-graph G_D
$w_D(e)$	Edge-weight of $e \in E_D$.
V_q	Set of query nodes *s.t.* $V_q \subseteq V_D$
$V_q(i)$	Set of query nodes having a node-type i in V_q
$\mathbf{r}$	Importance vector of G_D
$\mathbf{q}$	Query vector of G_D
$\mathbf{A}$	Transition matrix of G_D
α	Dumping factor *s.t.* $\alpha \in [0, 1]$
$\mathbf{r}_S$	Importance vector of G_S in Definition 2
$\mathbf{q}_S$	Query vector of G_D in Definition 1
$\mathbf{S}$	Transition matrix of G_D in Definition 2
$\underline{r}_i^{(t)}$	Lower bound of r_i in the t-th iteration
$\overline{r}_i^{(t)}$	Upper bound of r_i in the t-th iteration
$\epsilon^{(t)}$	k-th highest lower bound in the t-th iteration.



(a) Schema-graph.

(b) Data-graph

Fig. 1. A schema-graph and its corresponding data-graph.

Figure 1 (a) depicts a schema-graph example for bibliographic entities. The graph consists of four node-types (Conference, Year, Paper, and Author). Because "cites" has the largest edge-weight in W_S, herein the importance evaluation emphasizes paper-citation relationships.

Data-Graph: ObjectRank generates a data-graph $G_D = (V_D, E_D, W_D)$ from a schema-graph to materialize actual entity relationships, where V_D, E_D, and W_D are sets of nodes, edges, and edge-weights, respectively. V_D represents entities, each of which has a node-type defined in the schema-graph. We denote $V_D(i)$ as a set of nodes with node-type i in V_D. ObjectRank links pairs of nodes u and v with node-types i and j, respectively, if the schema-graph has an edge between node-types i and j. For each edge $e_{u,v}$ linking u to v, it assigns an edge-weight, $w_D(e_{u,v}) \in W_D$, which is equal to the edge-weight $w_S(e_{i,j})$ divided by the number of nodes in $V_D(j)$ connected from node u.

Figure 1 (b) shows an example of a data-graph instantiated from the schema-graph in Fig. 1 (a), where each node represents an actual entity (*e.g.,* paper title and author name, associated with the corresponding node-type). In Fig. 1 (b), Alice has an edge-weight of 0.1 for each Paper node because (1) the edge-weight between Author and Paper is 0.2 in the schema-graph and (2) Alice is adjacent to two Paper nodes.

2.2 Importance Evaluation

ObjectRank computes the importance of nodes on the data-graph. Given a set of query nodes $V_q \subseteq V_D$, it returns an importance vector $\mathbf{r} = (r_1, r_2, \ldots, r_{|V_D|})^{\mathrm{T}}$, where $r_i \in \mathbb{R}$ is the importance of node $i \in V_D$. By letting $\alpha \in [0, 1]$ and $r_i = 1/|V_D|$, $\mathbf{r}$ can be obtained by iteratively applying the following equation until $\mathbf{r}$ converges:

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

where $\mathbf{A} \in \mathbb{R}^{|V_D| \times |V_D|}$ is a transition matrix of G_D, whose (i, j)-th element A_{ij} is equal to $w_D(e_{i,j}) \in W_D$. Additionally, $\mathbf{q} \in \mathbb{R}^{|V_D|}$ is a query vector, whose i-th element q_i is $1/|V_q|$ if node $i \in V_q$. Otherwise $q_i = 0$. Although the convergence of Eq. (1) is guaranteed [14], it requires $O((|V_D| + |E_D|)t)$ time until it converges, where t is the number of iterations.[1]

3 Proposed Method: SchemaRank

We present SchemaRank that efficiently detects exact top-k important nodes for a query. First, we overview the ideas and then give a full description.

3.1 Basic Ideas

Our goal is to efficiently find the same top-k important nodes as ObjectRank. ObjectRank iteratively computes all nodes in a data-graph until $\mathbf{r}$ converges. By contrast, SchemaRank dynamically prunes unpromising nodes that cannot become the top-k nodes after convergence. As discussed by Sun *et al.*, real-world

[1] ObjectRank can easily handle updates of graphs (*e.g.,* nodes/edges insertion or deletion, and changes of weights) by using the Gauss-Southwell algorithm. Thus, this work does not consider such updates.

graphs have a highly skewed importance distribution [24]. The vast majority of nodes practically yield a low importance. Hence, we design SchemaRank to prune such unpromising nodes using a two-step approach: *coarse-grained RWR* and *fine-grained RWR*. First, the coarse-grained RWR roughly estimates the importance distribution at the node-type level. Then the fine-grained RWR prunes unpromising nodes by incrementally refining the distribution at the node level. Once a specific node-type yields a low importance, all nodes with that node-type are removed from the data-graph. Instead of computing all nodes, SchemaRank computes only essential nodes that are likely to become the top-k nodes.

Our approach has several advantages. (1) SchemaRank finds the top-k nodes with a short running time on real-world graphs. Our approach successfully handles the skewness of the importance distribution in real-world graphs [24] because SchemaRank increases its performance if a lot of nodes output a low importance. This leads to computation efficiency. (2) SchemaRank finds the same top-k nodes as ObjectRank. We theoretically demonstrated that SchemaRank safely discards unpromising nodes. Thus, SchemaRank does not sacrifice the quality of ObjectRank. (3) Because SchemaRank does not require any pre-computations, it can efficiently find the exact top-k nodes as ObjectRank on the fly. Hence, SchemaRank provides users with a simple solution using ObjectRank.

3.2 Coarse-Grained RWR

As the first step, SchemaRank roughly estimates the importance distribution at the node-type level. To estimate the importance for each node-type, SchemaRank performs RWR on the schema-graph G_S. Recall that the schema-graph represents the relationships among node-types. By performing RWR on G_S, our algorithm detects node-types with a low importance in the importance evaluation.

To estimate the importance distribution at the node-type level, SchemaRank first constructs the following vector:

Definition 1 (Query vector $\boldsymbol{q}_S$). *Let $V_q(i)$ be a set of query nodes with a node-type $i \in V_S$. $\boldsymbol{q}_S \in \mathbb{R}^{|V_S|}$ is a query vector whose i-th element is $q_{S,i} = |V_q(i)|/|V_q|$.*

To illustrate an example of $\mathbf{q}_S$, let's say $V_q = \{\text{Alice}, \text{ISWC}\}$ is a set of user-specified query nodes selected from the data-graph in Fig. 1. We clearly have $|V_q(\text{Author})| = 1$ from the corresponding schema-graph. Thus, $q_{S,\text{Author}} = 0.5$.

SchemaRank then estimates the distribution as follows:

Definition 2 (Importance vector of G_S). *SchemaRank computes the following importance vector $\boldsymbol{r}_S \in \mathbb{R}^{|V_S|}$:*

$$\boldsymbol{r}_S = \alpha \boldsymbol{S} \boldsymbol{r}_S + (1-\alpha)\boldsymbol{q}_S, \tag{2}$$

where $\alpha \in [0,1]$, and $\boldsymbol{S} \in \mathbb{R}^{|V_S| \times |V_S|}$ is a transition matrix of G_S, whose (i,j)-th element $S_{ij} = w_S(e_{i,j})$ in W_S.

In Definition 2, SchemaRank estimates the importance distribution at the node-type level by performing RWR on the schema-graph G_S. Definition 2 has the following property:

Lemma 1. *Let $V_D(i)$ be a set of nodes in V_D with a node-type $i \in V_S$. $r_{S,i} = \sum_{u \in V_D(i)} r_u$ always holds, where $r_{S,i}$ is the i-th element of $\boldsymbol{r}_S$.*

Proof: From Eq. (2), we have the following

$$r_{S,i} = \alpha \sum_{j \in V_S} S_{ij} r_{S,j} + (1-\alpha) q_{S,i}. \tag{3}$$

From Definition 1, $q_{S,i} = \sum_{u \in V_D(i)} q_u$. Also, from Sect. 2.1, $\sum_{j \in V_S} S_{ij} r_{S,j} = \sum_{u \in V_D(i)} \sum_{v \in V_D} A_{uv} r_v$. Therefore, we have the following equation:

$$\text{Eq. (3)} = \sum_{u \in V_D(i)} \left\{ \alpha \sum_{v \in V_D} A_{uv} r_v + (1-\alpha) q_u \right\}. \tag{4}$$

Thus, $r_{S,i} = \sum_{u \in V_D(i)} r_u$ always holds. □

Lemma 1 implies that Definition 2 effectively estimates the distribution of node-type level importance obtained by ObjectRank. Additionally, from Lemma 1, $r_u \leq r_{S,i}$ holds for any $u \in V_D(i)$. That is, nodes with a node-type i should be unpromising in ObjectRank if $r_{S,i}$ yields a low importance. By following this property, SchemaRank prunes such nodes in subsequent steps.

3.3 Fine-Grained RWR

Next, SchemaRank detects the top-k important nodes from the importance vector $\mathbf{r}_S$ estimated in the previous step. SchemaRank iterates (1) *upper and lower bounds estimation* and (2) *incremental pruning* until convergence.

Upper and Lower Bounds Estimation: In this step, SchemaRank estimates the bounds of node-importance in the data-graph by using the query vector $\mathbf{q}$ and the importance vector $\mathbf{r}_S$ estimated in the previous step.

Suppose that node $i \in V_D$ has a node-type $u \in V_S$. As we proved in Lemma 1, the importance r_i should be smaller than $r_{S,u}$. That is, $r_{S,u}$ plays a good upper bound of the importance r_i. To leverage this property, SchemaRank first constructs a vector $\mathbf{b}^{(0)} \in \mathbb{R}^{|V_D|}$ from $\mathbf{r}_S$, whose i-th element is initialized as $b_i^{(0)} = r_{S,u} / \|\mathbf{b}^{(0)}\|$. Afterward, it estimates the following upper and lower bounds of node-importance from $\mathbf{b}^{(0)}$ and $\mathbf{q}$, respectively.

Definition 3 (Upper and lower bounds). *In the t-th iteration, the lower bound* $\underline{r}_i^{(t)}$ *and the upper bound* $\overline{r}_i^{(t)}$ *of node i are defined as*

$$\underline{r}_i^{(t)} = \begin{cases} (1-\alpha)q_i & (t=0) \\ \underline{r}_i^{(t-1)} + (1-\alpha)\alpha^t p_i^{(t)} & (t>0) \end{cases}, \tag{5}$$

$$\overline{r}_i^{(t)} = \begin{cases} b_i^{(0)} + \frac{\alpha}{1-\alpha}\overline{A}_i & (t=0) \\ \underline{r}_i^{(t-1)} + \alpha^t b_i^{(t)} + \frac{\alpha^{t+1}}{1-\alpha}\Delta^{(t)}\overline{A}_i & (t>0) \end{cases}, \tag{6}$$

where $\overline{A}_i = \max\{A_{ij} | j \in V_D\}$, $\Delta^{(t)} = \sum_{u \in V_D} \max\{b_u^{(t)} - b_u^{(t-1)}, 0\}$, *and* $p_i^{(t)}$ *and* $b_i^{(t)}$ *are the i-the elements of* $\boldsymbol{p}^{(t)} = \boldsymbol{A}^t\boldsymbol{q}$ *and* $\boldsymbol{b}^{(t)} = \boldsymbol{A}^t\boldsymbol{b}^{(0)}$, *respectively.*

Definition 3 estimates the importance of each node after convergence. Based on the following properties, Definition 3 can more precisely estimate the importance of each node by recursively updating $\overline{r}_i^{(t)}$ and $\underline{r}_i^{(t)}$.

Lemma 2. *By Definition 3,* $\underline{r}_i^{(t)} \leq r_i \leq \overline{r}_i^{(t)}$ *holds.*

Proof: We first prove $\underline{r}_i^{(t)} \leq r_i$. From Eq. (1),

$$\mathbf{r}^{(t)} = \alpha^t \mathbf{A}^t \mathbf{r}^{(0)} + (1-\alpha)\sum_{j=0}^{t-1} \alpha^j \mathbf{A}^j \mathbf{q}. \tag{7}$$

If $t \to \infty$, $\mathbf{r} = \mathbf{r}^{(\infty)}$. Thus, assuming the number of iterations t goes to ∞, the following condition can be derived

$$r_i = (1-\alpha)\sum_{j=0}^{\infty} \alpha^j p_i^{(j)} \geq (1-\alpha)\sum_{j=0}^{t} \alpha^j p_i^{(j)} = \underline{r}_i^{(t)}. \tag{8}$$

We then prove $r_i \leq \overline{r}_i^{(t)}$. If $t \to \infty$, a Markov chain converges to the stationary distribution regardless of where it begins [28]. That is, $p_i^{(t)} \approx b_i^{(t)}$ holds for a large t. Thus, from Eq. (1), we have the following condition

$$\begin{aligned} r_i \approx \quad & (1-\alpha)\sum_{j=0}^{t} \alpha^j b_i^{(j)} + (1-\alpha)\sum_{j=1}^{\infty} \alpha^{t+j} b_i^{(t+j)} \\ \leq\ & (1-\alpha)\sum_{j=0}^{t} \alpha^j b_i^{(j)} + \alpha^t b_i^{(t)} + \frac{\alpha^{t+1}}{1-\alpha}\Delta^{(t)}\overline{A}_i = \overline{r}_i^{(t)}. \end{aligned} \tag{9}$$

Therefore, from Eqs. (8) and (9), $\underline{r}_i^{(t)} \leq r_i \leq \overline{r}_i^{(t)}$ holds. □

Lemma 3. *If* $t = \infty$, $\underline{r}_i^{(\infty)} = \overline{r}_i^{(\infty)} = r_i$ *always holds.*

Proof: From Eqs. (8) and (9), $\underline{r}_i^{(t)} \to r_i$ and $\overline{r}_i^{(t)} \to r_i$ if $t \to \infty$ since $\alpha^\infty = 0$ and $\frac{\alpha^\infty}{1-\alpha} = 0$. Thus, from Lemma 2, $\underline{r}_i^{(\infty)} = \overline{r}_i^{(\infty)} = r_i$ holds. □

Lemmas 2 and 3 indicate that the exact importance r_i can be obtained by continuously updating $\overline{r}_i^{(t)}$ and $\underline{r}_i^{(t)}$. Based on this property, SchemaRank incrementally prunes unpromising nodes in the following procedure.

Algorithm 1. Proposed method: SchemaRank

Input: G_S: schema-graph, G_V: data-graph, $\mathbf{q}$: query vector, and k: # of results
Output: T: a set of top-k nodes

1: $t \leftarrow 0$;
2: Construct $\mathbf{q}_S$ by Definition 1;
3: **while** $\mathbf{r}_S^{(t)}$ does not converge **do**
4: $\quad \mathbf{r}_S^{(t+1)} \leftarrow \alpha \mathbf{S}\mathbf{r}_S^{(t)} + (1-\alpha)\mathbf{q}_S$;
5: $\quad t \leftarrow t+1$;
6: Construct $\mathbf{b}^{(0)}$, $t \leftarrow 0$, and $T \leftarrow V_D$;
7: **while** $\underline{r}_i^{(t)}$ and $\overline{r}_i^{(t)}$ do not converge for $\forall i \in T$ **do**
8: $\quad$ **for each** $i \in T$ **do**
9: $\quad\quad$ Update $\underline{r}_i^{(t)}$ and $\overline{r}_i^{(t)}$ by Definition 3;
10: $\quad$ **If** $T > k$ **then** $T \leftarrow V_D^{(t)}$ by Definition 4;
11: $\quad t \leftarrow t+1$;
12: **return** T;

Incremental Node Pruning: Using the upper and lower bounds, SchemaRank incrementally prunes unpromising nodes that cannot become the top-k important nodes. Let $\epsilon^{(t-1)}$ be the k-th highest lower bound obtained in $(t-1)$-th iteration. In each iteration, SchemaRank constructs the following subset by incrementally pruning unpromising nodes.

Definition 4 (Incremental pruning). *In the t-th iteration, SchemaRank constructs a subset of V_D defined as* $V_D^{(t)} = \{i \in V_D^{(t-1)} | \overline{r}_i^{(t-1)} \geq \epsilon^{(t-1)}\}$.

Definition 4 indicates that SchemaRank incrementally prunes a node if its upper bound $\overline{r}_i^{(t-1)}$ is smaller than $\epsilon^{(t-1)}$. This leads the following property.

Lemma 4. *If $i \notin V_D^{(t)}$, node i never becomes a top-k node.*

Proof: Suppose node i has been pruned in the t-th iteration, *i.e.*, $\overline{r}_i^{(t)} < \epsilon^{(t)}$. From Lemmas 2 and 3, the bounds become tighter as the number of iteration t increases. That is, we hold $r_i = \overline{r}_i^{(\infty)} \leq \cdots \leq \overline{r}_i^{(t+1)} \leq \overline{r}_i^{(t)}$, and $\epsilon^{(t)} \leq \epsilon^{(t+1)} \leq \cdots \leq \epsilon^{(\infty)}$. Thus, once $i \notin V_D^{(t)}$ holds, $r_i < \epsilon^{(\infty)}$, which completes the proof. □

From Lemma 4, SchemaRank can safely remove unpromising nodes during the iterative computations.

3.4 Algorithm

Algorithm 1 gives a full description of SchemaRank. First, SchemaRank performs the coarse-grained RWR (Sect. 3.2) on the schema-graph G_S (lines 2–5). It iterates RWR on G_S until the importance vector $\mathbf{r}_S$ converges. Afterward, SchemaRank starts the fine-grained RWR (Sect. 3.3) on the data-graph G_D (lines 6–12). To estimate the importance of nodes, it constructs vector $\mathbf{b}^{(0)}$ from

$\mathbf{r}_S$ obtained in the previous step (line 6). Then it estimates the upper and the lower bounds of each node from Definition 3 (lines 8–9), and it prunes unpromising nodes by following Definition 4 (line 10). SchemaRank iterates (1) the recursive bounds updated by Definition 3 and (2) the incremental node pruning by Definition 4 until T reaches k. Once the top-k importance nodes are specified, SchemaRank performs only the bound updates (lines 8–9) to obtain converged node importance by following Lemma 3 (line 7).

Unlike existing methods, SchemaRank does not require pre-computations. Hence, it is a simple solution to find the same top-k nodes as ObjectRank.

Next we discuss the theoretical aspects of SchemaRank. Let c and d be the average sizes of $V_D^{(t)}$ and degree, respectively. We have the following properties:

Theorem 1. *SchemaRank requires* $O((|V_S|+|E_S|)t_S+(cd+\log c \log k)t_D)$ *time, where* t_S *and* t_D *are the numbers of iterations on* G_S *and* G_D*, respectively.*

Proof: From Definition 2, the coarse-grained RWR incurs $O((|V_S| + |E_S|)t_S)$ time. In the fine-grained RWR, it explores the top-k nodes by performing the incremental pruning. In each iteration, SchemaRank estimates the upper and the lower bounds for all nodes in $V_D^{(t)}$. This procedure incurs $O(cd)$ time because $b_i^{(t)}$ and the bounds are obtained in $O(d)$ time and $O(1)$ time, respectively. SchemaRank then updates $V_D^{(t)}$ by Definition 4. The k-th highest lower bound $\epsilon^{(t)}$ can be found in $O(\log c \log k)$ time using Fibonacci heaps. Thus, SchemaRank needs $O((|V_S| + |E_S|)t_S + (cd + \log c \log k)t_D)$ time. □

In practice, $|V_S| \ll c \leq |V_D|$, and $|E_S| \ll cd \leq |E_D|$ because the schema-graph consists of only the node-types. Hence, the practical time complexity is $O((|V_S| + |E_S|)t_S + (cd + \log c \log k)t_D) \approx O((cd + \log c \log k)t_D)$. Consequently, the running costs of SchemaRank is dramatically lower than ObjectRank, which requires $O((|V_D| + |E_D|)t_D)$ time until convergence.

Theorem 2. *SchemaRank outputs the same top-k nodes as ObjectRank.*

Proof: From Lemma 3, we have $\underline{r}_i^{(\infty)} = \overline{r}_i^{(\infty)} = r_i$. Thus, SchemaRank can detect the same k-th highest important nodes as ObjectRank by increasing the number of iterations. From Lemma 4, once node $i \notin V_D^{(t)}$, it cannot become the top-k important nodes. Thus, Theorem 2 holds. □

Theorem 2 indicates that SchemaRank efficiently detects the top-k nodes without sacrificing the accuracy of ObjectRank.

4 Evaluation

Here, we experimentally discuss the efficiency and the exactness of SchemaRank.
Methods: We compared SchemaRank with the following algorithms.

– **ObjectRank:** The baseline method [9].

Table 2. Statistics of real-world datasets.

Name	$\|V_D\|$	$\|E_D\|$	Source
ACM (small)	629 K	632 K	Citation-network V1
DBLP (small)	1.51 M	2.08 M	DBLP-Citation-network V4
ACM (large)	2.38 M	10.4 M	ACM-Citation-network V8
DBLP (large)	4.10 M	36.6 M	DBLP-Citation-network V11

- **BinRank:** The indexing algorithm for ObjectRank [10]. It runs ObjectRank for several clustered queries in advance. Then it indexes nodes whose importance is higher than a threshold. Given a query, BinRank performs RWR on the indexed nodes.
- **LORank:** The pruning method for ObjectRank [19]. It prunes nodes whose importance is lower than the user-specified threshold.
- **SimMat:** The top-k pruning algorithm for RWR [7]. SimMat estimates low important nodes through SVD-based pre-computation. Given a query, it incrementally prunes nodes with a low importance.
- **FORank:** The state-of-the-art pruning-based top-k algorithm for ObjectRank [20]. This is an extension of fast RWR algorithms, F-Rank [6] and Castanet [5]. It incrementally estimates the importance, and it prunes unpromising nodes by using a user-specified threshold.

We set $\alpha = 0.85$, which is the same as [9], and the maximum number of iterations is $10,000$. For all competitors, we used the parameter settings recommended by their papers. We conducted all experiments on a server with Intel Xeon CPU 2.60 GHz and 128 GiB RAM. We implemented all methods in C++ as a single-threaded program with the entire graph held in the main memory.

Datasets: We used four public real-world graphs published by AMiner [25].[2] Table 2 shows their statistics. The graphs are extracted from ACM digital library or DBLP bibliographic database. They are composed of four node-types: authors, papers, conferences, and years.

Schema-Graph: In the experimental evaluations, we tested the following schema-graphs with different edge-weight distributions.

- **Skewed:** The schema-graph shown in Fig. 1 (a), which has skewed edge-weights as same as [10].
- **Uniform:** A schema-graph that assigns uniform edge-weights to the graph in Fig. 1 (a). We assigned an edge-weight 0.5 for all edges in Fig. 1 (a).

Query Nodes: For each dataset, we randomly selected 100 nodes from the data-graph, and tested each algorithm using the query nodes. We reported the average results with the standard deviation when testing each query 50 times.

[2] All datasets are publicly available at https://aminer.org/citation.

Table 3. Running time (± standard deviation) on real-world datasets.

(a) ACM (small)			
Methods	Skewed	Uniform	Pre-comp.
SchemaRank ($k=10^2$)	**0.82** (±0.002) **s**	**1.21** (±0.002) **s**	—
SchemaRank ($k=10^3$)	**1.01** (±0.003) **s**	**1.45** (±0.001) **s**	—
ObjectRank	4.51 (±0.007) s	4.41 (±0.009) s	—
BinRank	3.33 (±0.005) s	3.05 (±0.001) s	10.1 h
LORank	2.59 (±0.004) s	3.42 (±0.005) s	—
SimMat ($k=10^2$)	2.21 (±0.009) s	2.34 (±0.007) s	14.4 h
SimMat ($k=10^3$)	3.12 (±0.009) s	3.22 (±0.008) s	14.4 h
FORank ($k=10^2$)	1.75 (±0.003) s	1.94 (±0.001) s	—
FORank ($k=10^3$)	2.25 (±0.002) s	2.33 (±0.002) s	—
(b) DBLP (small)			
Methods	Skewed	Uniform	Pre-comp.
SchemaRank ($k=10^2$)	**1.91** (±0.003) **s**	**2.54** (±0.003) **s**	—
SchemaRank ($k=10^3$)	**2.08** (±0.002) **s**	**2.63** (±0.003) **s**	—
ObjectRank	22.8 (±0.011) s	22.7 (±0.009) s	—
BinRank	6.42 (±0.009) s	7.22 (±0.011) s	17.9 h
LORank	16.4 (±0.008) s	17.2 (±0.008) s	—
SimMat ($k=10^2$)	N/A	N/A	>24 h
SimMat ($k=10^3$)	N/A	N/A	>24 h
FORank ($k=10^2$)	6.77 (±0.002) s	7.45 (±0.003) s	—
FORank ($k=10^3$)	8.68 (±0.003) s	9.03 (±0.004) s	—
(c) ACM (large)			
Methods	Skewed	Uniform	Pre-comp.
SchemaRank ($k=10^2$)	**41.1** (±0.056) **s**	**55.3** (±0.064) **s**	—
SchemaRank ($k=10^3$)	**56.3** (±0.066) **s**	**68.6** (±0.072) **s**	—
ObjectRank	24,122 (±1.41) s	24,306 (±1.67) s	—
BinRank	N/A	N/A	>24 h
LORank	22,068 (±4.43) s	23,328 (±5.09) s	—
SimMat ($k=10^2$)	N/A	N/A	>24 h
SimMat ($k=10^3$)	N/A	N/A	>24 h
FORank ($k=10^2$)	320 (±0.224) s	347 (±0.206) s	—
FORank ($k=10^3$)	381 (±0.258) s	423 (±0.211) s	—
(d) DBLP (large)			
Methods	Skewed	Uniform	Pre-comp.
SchemaRank ($k=10^2$)	**134** (±0.195) **s**	**169** (±0.106) **s**	—
SchemaRank ($k=10^3$)	**168** (±0.173) **s**	**241** (±0.122) **s**	—
ObjectRank	>24 h	>24 h	—
BinRank	N/A	N/A	>24 h
LORank	>24 h	>24 h	—
SimMat ($k=10^2$)	N/A	N/A	>24 h
SimMat ($k=10^3$)	N/A	N/A	>24 h
FORank ($k=10^2$)	2,209 (±1.908) s	2,677 (±1.966) s	—
FORank ($k=10^3$)	2,431 (±1.912) s	3,137 (±1.903) s	—

4.1 Efficiency

We evaluated the running time of each algorithm on real-world graphs. Table 3 shows the results for the two types of schema-graphs. For the top-k algorithms (*i.e.,* SchemaRank, SimMat, and FORank), we varied the size of k as 10^2 and 10^3. Since BinRank and SimMat require pre-computations, we also assessed their pre-computation time.

Overall Results: Table 3 demonstrates that SchemaRank outperforms the other algorithms. SchemaRank is up to 644.7 times and 16.5 times faster than ObjectRank and the state-of-the-art algorithm FORank, respectively. As described in Sect. 2.2, ObjectRank incurs $O((|V_D| + |E_D|)t)$ time since it iteratively computes all nodes and edges. By contrast, SchemaRank computes only the essential nodes and edges through the coarse-grained RWR and the fine-grained RWR. As shown in Fig. 1 (a), the schema-graphs are very small ($|V_S| = 4$ and $|E_S| = 8$). Thus, the computation cost for the coarse-grained RWR should be low. For instance, in this evaluation, the coarse-grained RWR consumed 10 microseconds at most. Hence, as we proved in Theorem 1, SchemaRank practically shows $O((|V_S| + |E_S|)t_S + (cd + \log c \log k)t_D) \approx O(cd + \log c \log k)$ time. Because this is a nearly linear time against c, which is the average size of the nodes computed in each iteration, our approach is faster than the other methods.

Skewed v.s. Uniform Schema-Graph: We then assessed the impact of the schema-graph types: Skewed and Uniform. Table 3 shows that Skewed requires a shorter running time than Uniform. These results imply that SchemaRank effectively handles the skewness of node-importance in the real-world datasets [24]. The coarse-grained RWR finds many node-types with a low importance if the edge-weights are highly skewed. Thus, in the Skewed schema-graph, it can prune a large portion of nodes earlier.

Pre-computation Time: Although BinRank and SimMat successfully reduced the running time on small graphs, they require long pre-computation time as shown in Table 3. As a result, they cannot compute large graphs, *i.e.,* ACM (large) and DBLP (large), since the pre-computation is not complete within 24 h. In contrast, SchemaRank does not require the pre-computation. Thus, our proposal can find the top-k nodes on the fly.

4.2 Effectiveness

Here, we discuss the effectiveness of the coarse-grained RWR. In Fig. 2, we compared the running time of SchemaRank with its variant, which excludes the coarse-grained RWR, and two baselines (FORank and ObjectRank). SchemaRank-w/o-CR represents SchemaRank without the coarse-grained RWR that replaces $\mathbf{b}^{(t)}$ with $\mathbf{p}^{(t)}$ in Eq. (6). Figure 2 demonstrates that SchemaRank is up to 12.5 times faster than SchemaRank-w/o-CR. By contrast, SchemaRank-w/o-CR is competitive or slightly faster than the state-of-the-art algorithm FORank. Thus, our coarse-grained RWR improves the efficiency without high

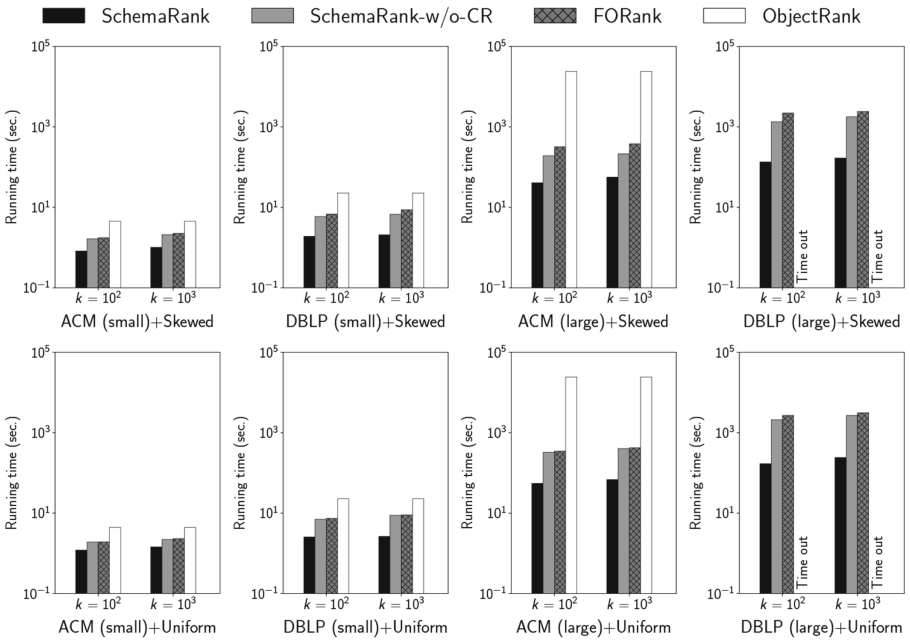


Fig. 2. Effectiveness of coarse-grained RWR

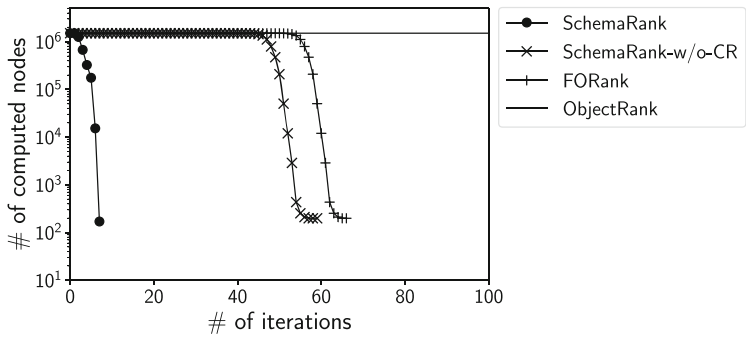


Fig. 3. # of computed nodes on DBLP (small).

pre-computation costs. Because it reveals the importance distribution at the node-type level, SchemaRank can quickly exclude unpromising nodes with a low importance.

To further discuss how the coarse-grained RWR effectively works, Fig. 3 plots the number of nodes computed by each algorithm in each iteration. In this evaluation, we used DBLP (small) with the schema-graph (Skewed), and we set $k = 10^2$. SchemaRank removes more than 97.9% nodes of the graph after a few iterations, whereas the other ones compute more significant numbers of nodes

and iterations. These results demonstrate that the coarse-grained RWR successfully reduces the computation costs and terminates the iterations earlier. In real-world graphs, the vast majority of nodes should have a low importance. By capturing this skewness through the coarse-grained RWR, SchemaRank successfully reduces the size of computed nodes c within a few iterations.

4.3 Exactness

A major advantage of SchemaRank is that it finds the same top-k important nodes as ObjectRank, while dynamically pruning unpromising nodes. To verify the exactness of SchemaRank, we compared the top-k nodes obtained by each algorithm against those obtained by ObjectRank. We used *average precision* [18] to measure the accuracy of top-k ranking importance compared with ObjectRank. The average precision is the mean score of precision@n [17] ($1 \leq n \leq k$). If an average precision is one, the top-k ranking importance is identical to ObjectRank. We tested $k = 10^2$ and $k = 10^3$ on all datasets except for DBLP (large).

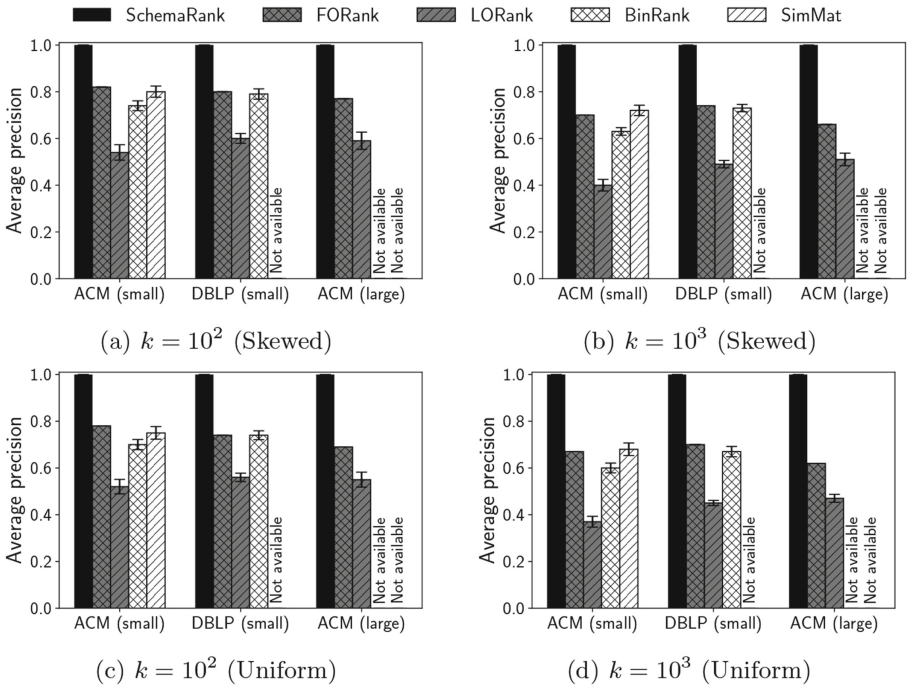


Fig. 4. Average precision of top-k ranking.

Figure 4 shows the average precision of each algorithm. SchemaRank is always one, whereas FORank, LORank, BinRank, and SimMat plateau at lower average

precision values. Hence, SchemaRank always outputs the same top-k results and ranking as ObjectRank. On the other hand, the other methods fail to reproduce the results of ObjectRank, although they reduce the runtime. As we proved in Theorem 2, SchemaRank guarantees that the same top-k nodes as ObjectRank are found. In addition, as discussed in Lemma 3, the upper and lower bounds converge to the same node importance as ObjectRank as the number of iterations increases. Therefore, SchemaRank can inherit the importance evaluation quality of ObjectRank, although it drastically reduces the running time.

5 Conclusion

We proposed SchemaRank, which is an efficient algorithm that detects exact top-k important nodes from massive heterogeneous graphs (*e.g.,* knowledge graphs). SchemaRank estimates the node-importance distribution at the node-type level, and it prunes unpromising nodes by a two-step RWR approach. In the experiments, SchemaRank offers an improved efficiency for massive graphs without sacrificing the accuracy of ObjectRank. Currently ObjectRank is an essential tool for many applications in the Semantic Web community. SchemaRank will help to improve the quality of current and future Semantic Web applications.

Acknowledgement. This work was partially supported by JSPS KAKENHI Early-Career Scientists (Grant Number JP18K18057) and JST PRESTO JPMJPR2033, Japan. I thank to Hiroyuki Kitagawa, Toshiyuki Amagasa, and Tomoki Sato for their helps and insightful discussions.

References

1. Chakrabarti, S.: Dynamic personalized pagerank in entity-relation graphs. In: Proceedings of the 16th International Conference on World Wide Web (WWW), pp. 571–580 (2007)
2. Fang, H., Wu, F., Zhao, Z., Duan, X., Zhuang, Y.: Community-based question answering via heterogeneous social network learning. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI), pp. 122–128 (2016)
3. Fazzinga, B., Gianforme, G., Gottlob, G., Lukasiewicz, T.: Semantic Web Search based on Ontological Conjunctive Queries. Journal of Web Semantics **9**(4), 453–473 (2011)
4. Fu, T.Y., Lee, W.C., Lei, Z.: HIN2Vec: explore meta-paths in heterogeneous information networks for representation learning. In: Proceedings of the 26th ACM International Conference on Information and Knowledge Management (CIKM), pp. 1797–1806 (2017)
5. Fujiwara, Y., Nakatsuji, M., Shiokawa, H., Mishima, T., Onizuka, M.: Efficient ad-hoc search for personalized PageRank. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD), pp. 445–456 (2013)
6. Fujiwara, Y., Nakatsuji, M., Shiokawa, H., Mishima, T., Onizuka, M.: Fast and exact top-k algorithm for PageRank. In: Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI 2013), pp. 1106–1112 (2013)

7. Fujiwara, Y., Nakatsuji, M., Shiokawa, H., Onizuka, M.: Efficient search algorithm for SimRank. In: Proceedings of the 29th IEEE International Conference on Data Engineering (ICDE), pp. 589–600 (2013)
8. Golub, G., Van Loan, C.: Matrix Computations, 4th edn. Johns Hopkins University Press, Baltimore (2012)
9. Hristidis, V., Hwang, H., Papakonstantinou, Y.: Authority-based keyword search in databases. ACM Trans. Database Syst. **33**(1) (2008)
10. Hwang, H., Balmin, A., Reinwald, B., Nijkamp, E.: BinRank: scaling dynamic authority-based search using materialized subgraphs. IEEE Trans. Knowl. Data Eng. **22**(8), 1176–1190 (2010)
11. Jeh, G., Widom, J.: SimRank: a measure of structural-context similarity. In: Proceedings of the 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pp. 538–543 (2002)
12. Jiang, Z., Liu, H., Fu, B., Wu, Z., Zhang, T.: Recommendation in heterogeneous information networks based on generalized random walk model and Bayesian personalized ranking. In: Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM), pp. 288–296 (2018)
13. Komamizu, T.: Learning interpretable entity representation in linked data. In: Hartmann, S., Ma, H., Hameurlain, A., Pernul, G., Wagner, R.R. (eds.) DEXA 2018. LNCS, vol. 11029, pp. 153–168. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98809-2_10
14. Langville, A.N., Meyer, C.D.: Google's PageRank and Beyond: The Science of Search Engine Rankings. Princeton University Press (2012)
15. Lehmann, J., et al.: DBpedia-a large-scale, multilingual knowledge base extracted from Wikipedia. Semantic Web **6**(2), 167–195 (2015)
16. Li, B., King, I.: Routing questions to appropriate answerers in community question answering services. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM), pp. 1585–1588 (2010)
17. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, New York (2008)
18. Robertson, S.: A New Interpretation of Average Precision. In: Proceedings of the 31st Annual International ACM SIGIR Conference (SIGIR), pp. 689–690 (2008)
19. Sakakura, Y., Yamaguchi, Y., Amagasa, T., Kitagawa, H.: A local method for ObjectRank estimation. In: Proceedings of the 15th International Conference on Information Integration and Web-based Applications and Services (iiWAS), pp. 92:92–92:101 (2013)
20. Sato, T., Shiokawa, H., Yamaguchi, Y., Kitagawa, H.: FORank: fast ObjectRank for large heterogeneous graphs. In: Companion Proceedings of The Web Conference (WWW), pp. 103–104 (2018)
21. Shiokawa, H.: Scalable affinity propagation for massive datasets. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2021), vol. 35, no. (11), pp. 9639–9646, May 2021
22. Shiokawa, H., Amagasa, T., Kitagawa, H.: Scaling fine-grained modularity clustering for massive graphs. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI), pp. 4597–4604 (2019)
23. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: a core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web (WWW), pp. 697–706 (2007)
24. Sun, J., Qu, H., Chakrabarti, D., Faloutsos, C.: Neighborhood formation and anomaly detection in bipartite graphs. In: Proceedings of the 5th IEEE International Conference on Data Mining (ICDM), pp. 418–425 (2005)

25. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: ArnetMiner: extraction and mining of academic social networks. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 990–998 (2008)
26. Tong, H., Faloutsos, C.: Center-piece subgraphs: problem definition and fast solutions. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 404–413 (2006)
27. Tsitsulin, A., Mottin, D., Karras, P., Müller, E.: VERSE: versatile graph embeddings from similarity measures. In: Proceedings of the 2018 World Wide Web Conference (WWW), pp. 539–548 (2018)
28. Wan, L., Lou, W., Abner, E., Kryscio, R.J.: A comparison of time-homogeneous Markov chain and Markov process multi-state models. Commun. Stat. Case Stud. Data Anal. Appl. **2**(3–4), 92–100 (2016)
29. Yu, D.L., Ma, Y.L., Yu, Z.G.: Inferring MicroRNA-disease association by hybrid recommendation algorithm and unbalanced bi-random walk on heterogeneous network. Sci. Rep. **9**(2474) (2019)
30. Zhao, Z., Lu, H., Cai, D., He, X., Zhuang, Y.: Microblog sentiment classification via recurrent random walk network learning. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI), pp. 3532–3538 (2017)