Finding Hubs without Materializing Meta-path Graphs Technical Report

Yudong Niu, Yuchen Li
Singapore Management University
{ydniu.2018@phdcs.,yuchenli@}smu.edu.sg

Panagiotis Karras
University of Copenhagen
piekarras@gmail.com

Yanhao Wang
East China Normal University
yhwang@dase.ecnu.edu.cn

ABSTRACT

How can we efficiently find important nodes, or *hubs*, based on implicit connections by meta-paths in a heterogeneous knowledge graph (KG)? Unfortunately, materializing a large meta-path graph incurs a prohibitive cost. In this paper, we propose a novel *sketch propagation* framework that finds hubs without explicitly materializing a meta-path graph. We develop the framework for *degree centrality* and extend it to *h-index*, i.e., the largest value h such that a node has h neighbors each of degree at least h. We also devise efficient pruning techniques to accelerate *personalized* queries, which ask whether a node q is a hub, under both measures. Through extensive experimentation, we confirm the efficacy and efficiency of our solutions, reaching orders of magnitude speedups compared to exact methods while consistently achieving accuracy beyond 90%.

PVLDB Reference Format:
Yudong Niu, Yuchen Li, Panagiotis Karras, and Yanhao Wang. Finding Hubs without Materializing Meta-path Graphs
Technical Report. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/YudongNiu/HUB.

1 INTRODUCTION

Finding hubs in graphs according to specific measures [8, 22] has applications in many domains such as evaluating network robustness [25, 27, 35, 41] and information propagation [7, 9, 24]. Most prior research on hub queries assumes that a materialized data graph is available and accessible with low latency. Still, in real-world scenarios, analysts work with large heterogeneous data graphs, e.g., knowledge graphs (KGs) [18, 19], which capture implicit relationships between entities, often encoded by *meta-paths* [12, 34, 47]. For example, the meta-path "(*author*) ⟶ (*paper*) ⟵ (*author*)" in an academic KG like DBLP reveals a coauthorship graph that links authors who have collaborated on at least one paper.

In this paper, we investigate the fundamental HUB problem, which finds hubs based on meta-paths in a KG, i.e., the set of nodes having *degree* or *h-index* centrality no less than a quantile node in the graph derived by a meta-path $\mathcal{M}$ in a KG $\mathcal{G}$.

Applications. Finding hubs in meta-path graphs pertains to various real-world applications, including:

- **Protein Complex Discovery:** Protein-protein interaction (PPI) networks describe interactions between proteins during physiological processes, expressed by meta-paths in the form "(*protein*) ⟶ (*function*) ⟵ (*protein*)"; hubs in a PPI network are pivotal in the discovery of protein complexes [32, 40].
- **Fraud Detection:** Financial KGs record electronic payment transactions, where meta-paths in the form "(*transaction*) ⟶ (*billing*) ⟵ (*transaction*)" establish connections between transactions sharing the same billing accounts; hubs in such a graph indicate potential fraudulent activities [5].
- **Influence Analysis:** Academic KGs record publications and induce collaboration networks via meta-paths in the form "(*author*) ⟶ (*paper*) ⟵ (*author*)" [7, 24]; hubs in such networks denote highly impactful researchers, while meta-paths with constraints [21, 30] reveal those in certain domains; for example, meta-path "(*author*) ⟶ (*paper*, *venue*='DB') ⟵ (*author*)" restricts publication venues to the database community, thus yields impactful database researchers.

Challenges. A naive solution to the HUB problem would enumerate all instances of $\mathcal{M}$ in $\mathcal{G}$ to construct a meta-path graph H, compute the degrees or h-indexes of all nodes in H, and sort them to find the hubs. However, such an explicit meta-path graph materialization is time- and memory-intensive, especially for large KGs [43]. One alternative would be to traverse meta-path instances in the KG and find a node's neighbors in the meta-path graph without explicitly materializing it. Nevertheless, to find hubs exactly, we should access the neighborhood information of all nodes in the meta-path graph and potentially traverse any edge involved in any meta-path instance up to $O(|V_{\mathcal{M}}|)$ times, where $|V_{\mathcal{M}}|$ is the number of nodes in the meta-path graph. Thus, this approach also falters when the meta-path graph, or KG, is large. Furthermore, since the number of meta-paths extracted from a heterogeneous KG using existing methods can be huge (up to several thousand in our experiments), it is inefficient to identify the hubs exactly for every meta-path graph.

Our Contributions. We find that in real-world scenarios, enumerating all hubs exactly is often unnecessary, and it suffices to find an approximate set of hubs accurately yet efficiently. Inspired by cardinality sketches [1, 10, 11, 14, 17], we propose a novel *sketch propagation* framework for approximate HUB queries. We construct a *neighborhood cardinality* sketch for each node to estimate its degree in the meta-path graph by iteratively propagating sketches along edges in matching instances of the given meta-path $\mathcal{M}$. Thereby, we jointly estimate the degrees of all nodes in the meta-path graph in a *single* propagation process and significantly enhance the efficiency of HUB queries under degree centrality.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

We then extend this sketch propagation framework to approximate HUB queries with the *h-index* measure [22]. A key challenge is that the h-index of a node is determined by not only its own degree, but also by the degrees of its neighbors, while cardinality sketches only provide information on a node's own degree. To fill this gap, we propose a *pivot-based* algorithm built on the sketch propagation framework. Rather than directly estimating a node's h-index, this algorithm recursively finds nodes having h-index value no less than that of a random pivot node based on the sketches, as quicksort does. Thereby, it effectively skips nodes that cannot be hubs and quickly answers HUB queries.

Furthermore, we provide efficient methods to answer *personalized* HUB queries, that is, to determine whether a query node is a meta-path hub. We maintain a lower bound on the number of nodes with higher meta-path graph centrality scores than the query node and terminate the sketch propagation early when the lower bound exceeds the desired quantile. This early termination mechanism applies to both personalized HUB queries based on degree centrality and those based on h-index score.

We establish theoretically that the sketch propagation framework provides unbiased and efficient approximations for HUB queries based on both measures. Our experiments demonstrate that the estimations converge much faster than the worst-case bound in terms of the number of propagations, achieving several orders of magnitude speedups compared to exact methods.

Our main contributions are summarized as follows:

- We introduce a novel problem of querying hubs (HUB) over meta-path graphs derived from KGs based on degree centrality and h-index. (Section 2)
- We propose a *sketch propagation* framework for approximate degree-based HUB queries and extend it to personalized HUB queries with early termination. (Section 3)
- We further devise a *pivot-based* algorithm, with early termination, for approximate (personalized) HUB queries based on h-index. (Section 4)
- Through extensive experimentation, we demonstrate that our proposals scale well to large meta-path graphs where exact methods fail to terminate in a reasonable time and achieve speedups of orders of magnitude over exact methods in most cases while yielding accuracy beyond 90% with default parameter settings. (Section 5)

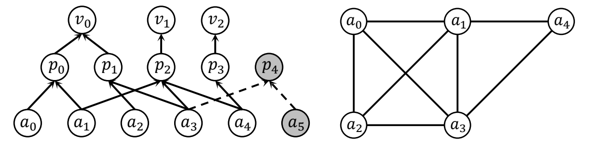
2 PRELIMINARIES

In this section, we introduce the basic notations, formulate the problem of finding hubs (HUB) over meta-path graphs derived from KGs, and present exact algorithms for HUB query processing.

2.1 Notations and Problem Formulations

A knowledge graph (KG) is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the sets of node and edge instances. An edge instance $e \in \mathcal{E}$ connects two node instances $u, v \in \mathcal{V}$. The function $\mathcal{L}$ maps each node or edge instance, v or e, to its type, $\mathcal{L}(v)$ or $\mathcal{L}(e)$. We use $\mathcal{G}$ to refer to the KG for ease of presentation.

Meta-paths are commonly used to extract graphs from KGs [12, 44]. We provide the formal definitions of *meta-path* and its *matching instances* on the KG as follows:



(a) Exemplary knowledge graph $\mathcal{G}$ (b) Meta-path graph H

Figure 1: An exemplary knowledge graph $\mathcal{G}$ and a meta-path graph H derived from $\mathcal{G}$ using meta-path $\mathcal{M}(A,$ 'write', $P,$ 'publish', $V)$. The unshaded nodes and solid edges in (a) denote the matching graph of $\mathcal{M}$.

Definition 2.1 (Meta-path). An L-hop meta-path is a sequence of node and edge types denoted as $\mathcal{M}(x_0, y_0, x_1, \ldots, x_{L-1}, y_{L-1}, x_L)$, where x_i is the type of the i-th node and y_i is the type of the i-th edge. The inverse of $\mathcal{M}$ is denoted as $\mathcal{M}^{-1}\ (x_L, y_{L-1}^{-1}, x_{L-1}, \ldots, x_1, y_0^{-1}, x_0)$, where y_i^{-1} is the inverse type of y_i.

Definition 2.2 (Matching Instance). A matching instance M to a meta-path $\mathcal{M}$ is a sequence of node and edge instances in $\mathcal{G}$ denoted by $M(v_0, e_0, v_1, \ldots, v_{L-1}, e_{L-1}, v_L) \triangleright \mathcal{M}$ satisfying that:

(1) $\forall i \in \{0, \ldots, L\}$, $\mathcal{L}(v_i) = x_i$.
(2) $\forall i \in \{0, \ldots, L-1\}$, $e_i = (v_i, v_{i+1}) \in \mathcal{E}$ and $\mathcal{L}(e_i) = y_i$.

When the context is clear, we use $M(v_0, v_1, \ldots, v_L)$ or simply M to denote a matching instance and use $M(x_i)$ to denote the node v_i in instance M. While our methods support any meta-path $\mathcal{M}$, following the established convention [12, 20, 44], we concatenate $\mathcal{M}$ and its inverse $\mathcal{M}^{-1}$ to induce a homogeneous meta-path graph $H_{\mathcal{M}}$.

Definition 2.3 (Meta-path Graph). Given a KG $\mathcal{G}$ and a meta-path $\mathcal{M}$, $\mathcal{M}$ induces a meta-path graph $H_{\mathcal{M}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ from $\mathcal{G}$ with node set $V_{\mathcal{M}}$ containing all the nodes in $\mathcal{V}$ that match x_0 in any instance of $\mathcal{M}$ and edge set $E_{\mathcal{M}}$ containing an edge $e = (u, v)$ for any two nodes $u, v \in V_{\mathcal{M}}$ such that there are $M \triangleright \mathcal{M}$ and $M' \triangleright \mathcal{M}^{-1}$ in $\mathcal{G}$ with (1) $M(x_0) = u$, (2) $M'(x_0) = v$, and (3) $M(x_L) = M'(x_L)$.

We denote the set of neighbors of u in $H_{\mathcal{M}}$ as $\mathcal{N}(u)$ and the *degree* of u in $H_{\mathcal{M}}$ as $N(u) = |\mathcal{N}(u)|$. Moreover, $n = \max_{u \in H} N(u)$ denotes the maximum degree in $H_{\mathcal{M}}$. When the context is clear, we drop $\mathcal{M}$ and use $H = (V, E)$ to represent a meta-path graph for ease of presentation.

Example 2.4. Figure 1 presents an exemplary academic KG with three node types A ('author'), P ('paper'), and V ('venue') and two edge types $A \rightarrow P$ ('write') and $P \rightarrow V$ ('publish'). A sequence $(A,$ 'write', $P,$ 'publish', $V)$ denotes a meta-path $\mathcal{M}$, which induces the meta-path graph H in Figure 1(b). For instance, two nodes a_0 and a_2 are neighbors in H because there exist an instance $M = (a_0, p_0, v_0)$ of $\mathcal{M}$ and another instance $M' = (v_0, p_1, a_2)$ of the inverse $\mathcal{M}^{-1}$ of $\mathcal{M}$. More intuitively, two authors a_0 and a_2 are connected since they published papers in the same venue v_0.

Problem Statement. In this paper, we study the problem of finding hubs (HUB) over meta-path graphs. Hubs are nodes that rank higher than most other nodes in the network according to a function $c : V \rightarrow \mathbb{R}_{\geq 0}$ that measures node importance. Specifically, we use degree centrality [8], i.e., the number of nodes connected to a node, and h-index [22], i.e., the largest integer h such that a node has h neighbors each of degree at least h, as measures of node importance. The HUB query we consider is formalized as the following:

Problem 1 (HUB Query). *Given a meta-path graph H and $\lambda \in (0,1)$, find every node $u \in V$ such that $c(u) \geq c(v^\lambda)$, where v^λ is the $(1-\lambda)$-quantile node under a centrality measure $c(\cdot)$.*

We also consider *personalized* queries, which inquire whether a query node$q$ is a hub of H.

Problem 2 (Personalized HUB Query). *Given a meta-path graph H, a query node q, and $\lambda \in (0,1)$, check whether $c(q) \geq c(v^\lambda)$.*

Processing HUB queries on large-scale meta-path graphs is computationally expensive. Therefore, we consider approximate HUB queries that can be answered efficiently by randomized algorithms based on the notion of ϵ-separable sets [15].

Definition 2.5 (ϵ-separable set). Given any node $u \in V$ and $\epsilon \in (0,1)$, the two ϵ-separable sets of u, denoted as $S_\epsilon^+(u)$ and $S_\epsilon^-(u)$, are the sets of nodes in H with centrality larger and smaller than $c(u)$ by a relative ratio of ϵ, respectively, i.e., $S_\epsilon^+(u) = \{v \in V | c(v) > (1+\epsilon) \cdot c(u)\}$ and $S_\epsilon^-(u) = \{v \in V | c(v) < (1-\epsilon) \cdot c(u)\}$.

Accordingly, we define the approximate versions of HUB and personalized HUB queries as follows:

Problem 3 (ϵ-Approximate HUB Query). *Given a meta-path graph H and $\epsilon, \lambda \in (0,1)$, return a set S of nodes in H such that $S_\epsilon^+(v^\lambda) \subseteq S$ and $S \cap S_\epsilon^-(v^\lambda) = \varnothing$.*

Problem 4 (ϵ-Approximate Personalized HUB Query). *Given a meta-path graph H, a query node q, and $\epsilon, \lambda \in (0,1)$, return* TRUE *if $q \in S_\epsilon^+(v^\lambda)$ and* FALSE *if $q \in S_\epsilon^-(v^\lambda)$.*

2.2 The Exact Algorithms

Constructing large meta-path graphs is often memory-prohibitive [43]. To address this issue, we propose a memory-efficient baseline solution to exact HUB queries, which computes the exact centrality of each node in H using a generic worst-case optimal join algorithm [2, 13, 26, 37] on the *matching graph* of $\mathcal{M}$ over $\mathcal{G}$. We first introduce the notion of *matching graph* of a meta-path as follows:

Definition 2.6 (Matching Graph). Given a KG $\mathcal{G}$ and a meta-path $\mathcal{M}$, the *matching graph* of $\mathcal{M}$ over $\mathcal{G}$ is a *multi-level* graph $\mathcal{G}^*(\mathcal{V}^*, \mathcal{E}^*)$ with a node set $\mathcal{V}^* = \bigcup_{0 \leq i \leq L} \mathcal{V}_i$ and an edge set $\mathcal{E}^* = \bigcup_{0 \leq i < L} \mathcal{E}_i$, where $\mathcal{V}_i$ is the set of all node instances of type x_i contained in any matching instance of $\mathcal{M}$ and $\mathcal{E}_i$ consists of all edges of type y_i that connects two nodes between $\mathcal{V}_i$ and $\mathcal{V}_{i+1}$.

Definition 2.7 (Successors and Predecessors). For each node $u \in \mathcal{V}_i$, we use $\mathcal{V}^+(u)$ to denote the set of nodes in $\mathcal{V}_{i+1}$ connected with u through edge type y_i, i.e., the *successors* of u in $\mathcal{G}^*$; and use $\mathcal{V}^-(u)$ to denote the set of nodes in $\mathcal{V}_{i-1}$ connected with u through edge type y_{i-1}, i.e., the *predecessors* of u in $\mathcal{G}^*$.

The matching graph can be extracted from $\mathcal{G}$ efficiently by a *forward* and *backward* breadth-first search (BFS). At each level $i \in [0, \ldots, L-1]$, the forward BFS iteratively visits all neighbors of each candidate matching node of x_i via edge type y_i as the candidates for x_{i+1}. Subsequently, the backward BFS visits all neighbors of the candidates for x_{i+1} via edge type y_i^{-1} as the candidates for x_i at each level $i \in [L-1, \ldots, 0]$. Only the candidate nodes and their adjacent edges visited by both forward and backward BFS are identified as the matching nodes and edges. Based on our experimental findings (see Table 2), the cost for extracting the matching graph is negligible compared to the overall computational cost for HUB queries.

Procedure DFS($u, v, \circ$)

Input: The matching graph $\mathcal{G}^*$ and the direction $\circ \in \{+, -\}$

1 Mark $(v, \circ)$ as visited;
2 **if** $v \in \mathcal{V}_L$ *and* $\circ = +$ **then**
3   DFS($u, v, -$);
4 **if** $v \in \mathcal{V}_0$ *and* $\circ = -$ **then**
5   Add v to $\mathcal{N}(u)$;
6 **for** $w \in \mathcal{V}^\circ(v)$ **do**
7   **if** $(w, \circ)$ *is not visited* **then**
8   DFS($u, w, \circ$);

To avoid generating excessive intermediate results, a generic worst-case optimal join is employed on the matching graph to compute the neighbors of each node $u \in V$ in the meta-path graph. This process starts with a depth-first search (DFS) from each node u and traverses the matching instances of $\mathcal{M}$ and $\mathcal{M}^{-1}$. Specifically, DFS($u, u, +$) (Procedure DFS) is executed for each node $u \in V$. Starting from u, it recursively traverses the successors (Lines 6–8) of the current node. Once it visits a node in $\mathcal{V}_L$, the direction of DFS will be switched from traversing the instances of $\mathcal{M}$ to those of $\mathcal{M}^{-1}$ (Lines 2 and 3). Once the traversal reaches $\mathcal{V}_0$ via any instance of $\mathcal{M}^{-1}$, the node v will be added to $\mathcal{N}(u)$ (Lines 4 and 5). Note that after performing a DFS for each node, the set $\mathcal{N}(u)$ is discarded, and only the degree $|\mathcal{N}(u)|$ is kept to ensure a small memory footprint. To compute the h-index values of all nodes without *materializing* the meta-path graph, the worst-case optimal join is performed in two rounds. The first round also computes and stores the degree of each node $v \in V$. Then, the second round computes the h-index value of each node v by combining its neighborhood $\mathcal{N}(v)$ with the degrees kept in the first round.

Although memory consumption is bounded by $O(|\mathcal{V}^*| + |\mathcal{E}^*|)$ and thus not a bottleneck, the exact algorithms are still very inefficient due to repeated edge traversal. Each edge in $\mathcal{E}^*$ is visited up to $O(|V|)$ times to compute the neighborhood of each node in V. Consequently, the time complexity of the exact algorithms is $O(|V| \cdot |\mathcal{E}^*|)$, which can be prohibitively expensive when dealing with large meta-path graphs.

3 SKETCH PROPAGATION FRAMEWORK

In this section, we introduce a novel *sketch propagation* framework to efficiently estimate the degree of each node in H by constructing cardinality sketches for their neighbors. As a result, it effectively answers HUB queries based on degree centrality. Additionally, we propose an early termination technique to improve the efficiency of personalized HUB query processing.

3.1 Sketch Propagation for Approximate HUB

Our basic idea is to approximate the degree of each node in H by constructing KMV sketches for the neighborhood $\mathcal{N}(v)$ of each node $v \in V$ without materializing H. The KMV sketch was initially proposed for distinct-value estimation [4] and defined as follows.

Definition 3.1 (KMV Sketch). Given a collection C of items and an integer $k > 0$, the KMV (k-th Minimum Value) sketch $\mathcal{K}(C)$ is built by independently drawing a number uniformly at random from $(0, 1)$ for each item in C and maintaining the k smallest random numbers. The set of random numbers generated for each item in C is called the *basis* for the KMV sketch. The cardinality of C is estimated by the estimator $|\widetilde{C}| = \mathbb{E}[(k-1)/\zeta]$, where ζ is a random variable by taking the largest random number in $\mathcal{K}(C)$.

There are two key challenges for applying KMV sketches and their corresponding estimators to HUB queries. First, the KMV sketch and its analogs are primarily designed for stream processing [1, 10, 11, 14, 17], where a one-pass scan over the collection C is required. However, for HUB queries, scanning the neighborhood of each node is an expensive operation, as discussed for exact algorithms. Second, prior works only provide asymptotic error bounds for the estimation when $|C| \to \infty$. Although assuming that a large number of elements will appear is reasonable for stream processing, the neighborhood size of each node in H can be relatively small. Thus, the original estimator for the KMV sketch does not provide meaningful error bounds for degree estimations in HUB queries.

To address the above two challenges, we first propose the *sketch propagation* framework that constructs a KMV sketch for $\mathcal{N}(v)$ of each $v \in H$ without explicitly generating and scanning $\mathcal{N}(v)$ and then offers a different estimator with a tight error bound for the case where $\mathcal{N}(v)$ is relatively small. Our framework is based on the notion of *images*.

Definition 3.2 (Images). The forward image of a matching node $u \in \mathcal{V}_i$, denoted as $\mathcal{I}_f(u)$, contains a node $v \in \mathcal{V}_0$ if there exists an instance M of $\mathcal{M}$ including both u and v, i.e.,

$$\mathcal{I}_f(u) = \{v \in \mathcal{V}_0 \mid \exists M \triangleright \mathcal{M}, M(x_0) = v \wedge M(x_i) = u\}.$$

The backward image of $u \in \mathcal{V}_i$, denoted as $\mathcal{I}_b(u)$, contains a node $v \in \mathcal{V}_0$ if there exists a pair of concatenated instances M and M' of $\mathcal{M}$ and $\mathcal{M}^{-1}$ that includes u and v, respectively, i.e.,

$$\begin{aligned}\mathcal{I}_b(u) = \{v \in \mathcal{V}_0 \mid \exists M \triangleright \mathcal{M}, M' \triangleright \mathcal{M}^{-1},\\ M(x_i) = u \wedge M(x_L) = M'(x_L) \wedge M'(x_0) = v\}.\end{aligned}$$

Due to the definition of meta-path graph, it is easy to see that $\mathcal{N}(u) = \mathcal{I}_b(u)$ for each node $u \in V$ in the meta-path graph.

Forward and Backward Propagation. We build a KMV sketch for $\mathcal{N}(u)$ of each $u \in V$ by iteratively maintaining the sketches for the forward and backward images of nodes from $\mathcal{V}_0$ to $\mathcal{V}_L$. First, a random number $r(u)$ is generated for each node $u \in \mathcal{V}_0$. Since $\mathcal{I}_f(u) = \{u\}$, the sketch is initialized as $\mathcal{K}(\mathcal{I}_f(u)) = \{r(u)\}$. The sketches for the forward images of nodes in $\mathcal{V}_i$ are then constructed by propagating the sketches of nodes in $\mathcal{V}_{i-1}$ iteratively. Specifically, each node in $\mathcal{V}_{i-1}$ will propagate its sketch to all its successor nodes in $\mathcal{V}_i$ and a node in $\mathcal{V}_i$ will build its sketch by retaining only the k smallest random numbers among all the sketches it receives. After building the sketches for the forward images of nodes in $\mathcal{V}_L$ (which equals to the sketches w.r.t. their backward images), the algorithm propagates the sketches back from the nodes in $\mathcal{V}_L$ to the nodes in $\mathcal{V}_0$ in the same way as for forward propagation so as to build the sketches for the backward images of nodes in $\mathcal{V}_0$ for estimating the degree of each node in H.

The following lemma ensures the correctness of the *sketch propagation* framework.

Lemma 3.3. *Given a meta-path $\mathcal{M}$, for each node $u \in \mathcal{V}_i$, we have*

$$\mathcal{K}(\mathcal{I}_f(u)) = \oplus_{v\in\mathcal{V}^-(u)}\mathcal{K}(\mathcal{I}_f(v)) \text{ and} \tag{1}$$

$$\mathcal{K}(\mathcal{I}_b(u)) = \oplus_{v\in\mathcal{V}^+(u)}\mathcal{K}(\mathcal{I}_b(v)), \tag{2}$$

where all KMV sketches are constructed from the same basis on $\mathcal{V}_0$, and the operator $\oplus$ extracts the k smallest random numbers from the union of the KMV sketches.

Proof. According to [4, Thm. 5], $\mathcal{K}(A) \oplus \mathcal{K}(B)$ is a KMV sketch for the collection $A \cup B$ for any two collections A and B. We thus only need to prove $\mathcal{I}_f(u) = \bigcup_{v\in\mathcal{V}^-(u)} \mathcal{I}_f(v)$ and $\mathcal{I}_b(u) = \bigcup_{v\in\mathcal{V}^+} \mathcal{I}_b(v)$ for each node $u \in \mathcal{V}_i$.

On the one hand, for any node $u' \in \mathcal{I}_f(u)$, there exists $M \triangleright \mathcal{M}$ such that $M(x_i) = u$ and $M(x_0) = u'$. Suppose that $v = M(x_{i-1})$, we have $u' \in \mathcal{I}_f(v)$ and $v \in \mathcal{V}^-(u)$. Thus, $u' \in \bigcup_{v\in\mathcal{V}^-(u)} \mathcal{I}_f(v)$, i.e., $\mathcal{I}_f(u) \subseteq \bigcup_{v\in\mathcal{V}^-(u)} \mathcal{I}_f(v)$. On the other hand, for any node $u' \in \bigcup_{v\in\mathcal{V}^-(u)} \mathcal{I}_f(v)$, there exists a node $v \in \mathcal{V}^-(u)$ such that $u' \in \mathcal{I}_f(v)$, i.e., there exists an instance $M' \triangleright \mathcal{M}$ such that $M'(x_0) = u'$ and $M'(x_{i-1}) = v$. Moreover, since $v \in \mathcal{V}^-(u)$, there exists an instance $M'' \triangleright \mathcal{M}$ such that $M''(x_{i-1}) = v$ and $M''(x_i) = u$. By concatenating $M'(x_0) = u'$ to $M'(x_{i-1}) = v$ with $M''(x_{i-1}) = v$ to $M''(x_L)$, we construct an instance $M \triangleright \mathcal{M}$ such that $M(x_0) = u'$ and $M(x_i) = u$. Thus, $u' \in \mathcal{I}_f(u)$, i.e., $\bigcup_{v\in\mathcal{V}^-(u)} \mathcal{I}_f(v) \subseteq \mathcal{I}_f(u)$. Therefore, we have $\mathcal{I}_f(u) = \bigcup_{v\in\mathcal{V}^-(u)} \mathcal{I}_f(v)$ for each node $u \in \mathcal{V}_i$. By symmetry, we can also prove that $\mathcal{I}_b(u) = \bigcup_{v\in\mathcal{V}^+} \mathcal{I}_b(v)$. □

Algorithm 1 describes the procedure of the *sketch propagation* framework. It receives a knowledge graph $\mathcal{G}$, a meta-path $\mathcal{M}$, a ratio $\lambda \in (0, 1)$, the number of propagations $\theta \in \mathbb{Z}^+$, and the sketch size $k \in \mathbb{Z}^+$ as input, and returns a set of nodes S for the (approximate) HUB query. It first initializes two arrays $\mathcal{K}_f$ and $\mathcal{K}_b$ to store θ KMV sketches for the respective forward and backward images of each node $u \in \mathcal{V}^*$ (Lines 1–4). Next, it calls the function Propagate, which constructs KMV sketches within $\mathcal{K}_f$ and $\mathcal{K}_b$ (Lines 9–13) and estimates the degree of each node u in H using $\mathcal{K}_b$ (Lines 14–21). Specifically, the function Receive depicts the step for KMV sketch construction when each node receives the sketches propagated from other nodes. Given a node u, the array $\mathcal{K}$ to store sketches, and a parameter $\circ$ indicating the propagation direction, it scans the random numbers in the sketches of nodes in either $\mathcal{V}^+(u)$ or $\mathcal{V}^-(u)$ and maintains the k-smallest numbers in $\mathcal{K}[u][t]$ with a min-heap of size k (Lines 24 and 25). Finally, it returns $\lambda \cdot |V|$ nodes with the largest estimated degrees as S (Lines 6 and 7). The tied values are broken arbitrarily.

Example 3.4. Figure 2 illustrates the sketch propagation framework with $k = 2$ and $\theta = 1$. The forward image $\mathcal{I}_f(u_1)$ is the union of forward images of nodes in its predecessors $\mathcal{V}^-(u_1) = \{v_0, v_1\}$, i.e., the union of nodes in the blue and red boxes at layer x_0. The sketch $\mathcal{K}(\mathcal{I}_f(u_1))$ is constructed by taking the smallest two numbers from the union of sketches of forward images of nodes in $\mathcal{V}^-(u_1)$. Thus, $\mathcal{K}(\mathcal{I}_f(u_1)) = \{0.1, 0.3\} \oplus \{0.2, 0.4\} = \{0.1, 0.2\}$. During the backward propagation, the sketch $\mathcal{K}(\mathcal{I}_b(v_0))$ is constructed similarly but in the reverse direction by taking the smallest two numbers from the union of sketches of backward images of nodes in $\mathcal{V}^+(v_0)$. Thus, $\mathcal{K}(\mathcal{I}_b(v_0)) = \{0.1, 0.15\} \oplus \{0.1, 0.2\} = \{0.1, 0.15\}$.

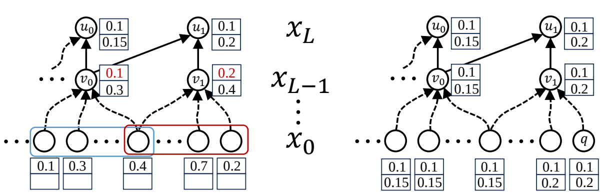
Algorithm 1: Sketch Propagation

Input: Knowledge graph $\mathcal{G}$, meta-path $\mathcal{M}$, quantile parameter λ, number of propagations θ, KMV sketch size k
Output: A set of nodes S for (approximate) HUB query

forall $u \in \mathcal{V}^*$ **do**
  for $t = 1$ **to** θ **do**
    $\mathcal{K}_f[u][t] \leftarrow \varnothing$ and $\mathcal{K}_b[u][t] \leftarrow \varnothing$;
    if $u \in \mathcal{V}_0$ **then** add $r(u) = \mathsf{Rand}(0,1)$ to $\mathcal{K}_f[u][t]$;
$\widetilde{N} \leftarrow \mathsf{Propagate}(\mathcal{K}_f, \mathcal{K}_b)$;
$S \leftarrow$ top-$(\lambda \cdot |V|)$ nodes with the highest degrees in H w.r.t. $\widetilde{N}$;
return S;

Function `Propagate`$(\mathcal{K}_f, \mathcal{K}_b)$:
  for $i = 1$ **to** L **do**
    forall $u \in \mathcal{V}_i$ **do** $\mathsf{Receive}(u, \mathcal{K}_f, -)$;
  forall $u \in \mathcal{V}_L$ **do** $\mathcal{K}_b[u] \leftarrow \mathcal{K}_f[u]$;
  for $i = L-1$ **to** 0 **do**
    forall $u \in \mathcal{V}_i$ **do** $\mathsf{Receive}(u, \mathcal{K}_b, +)$;
  forall $u \in \mathcal{V}_0$ **do**
    $\widetilde{\mu} \leftarrow 0$;
    for $t = 1$ **to** θ **do**
      if $|\mathcal{K}_b[u][t]| = k$ **then**
        $\widetilde{\mu} \leftarrow \widetilde{\mu} + \frac{\max(\mathcal{K}_b[u][t])}{\theta}$;
      else
        $\widetilde{\mu} \leftarrow \widetilde{\mu} + \frac{k}{(|\mathcal{K}_b[u][t]|+1)\cdot\theta}$;
    $\widetilde{N}[u] \leftarrow k/\widetilde{\mu} - 1$;
  return $\widetilde{N}$;

Function `Receive`$(u, \mathcal{K}, \circ)$:
  for $t = 1$ **to** θ **do**
    $\mathcal{K}[u][t] \leftarrow \oplus_{v \in \mathcal{V}^{\circ}(u)} \mathcal{K}[v][t]$;



(a) Forward propagation (b) Backward propagation

Figure 2: Illustration of forward and backward propagation with $k = 2$ and $\theta = 1$. Nodes in the blue and red boxes represent the forward images $\mathcal{I}_f(v_0)$ and $\mathcal{I}_f(v_1)$, respectively.

Theoretical Analysis. Next, we will prove that the *sketch propagation* framework accurately approximates HUB queries with high probability (w.h.p.) (cf. Theorem 3.8). For the proof, we first establish the following two lemmas: (1) Lemma 3.5 indicates the unbiasedness of sketch propagation in estimating the image sizes for each node in the matching graph; (2) Lemma 3.7 demonstrates the concentration properties that guarantee tight approximations around the true values.

For any node $u \in \mathcal{V}^*$, we use $I_f(u)$ and $I_b(u)$ to denote the sizes of $\mathcal{I}_f(u)$ and $\mathcal{I}_b(u)$. Moreover, we use ζ to denote the random variable that takes the maximum random number in the sketch of either u's forward or backward image and μ to denote the expectation of ζ. When the context is clear, we abbreviate $I_b(u)$ and $I_f(u)$ as I. We further use $\widetilde{I}$ and $\widetilde{\mu}$ to denote the estimation of I and μ given by the sketch propagation framework.

Lemma 3.5. *For any $u \in \mathcal{V}^*$, $I = k/\mu - 1$ if $I \geq k$.*

Proof. According to [4], if $I \geq k$, the probability density function of ζ will be $f_\zeta(t) = \frac{t^{k-1}(1-t)^{I-k}}{B(k, I-k+1)}$, where $B(a,b)$ denotes the beta function. Thus, we have

$$\begin{aligned} \mu &= \int_0^1 t f_\zeta(t)\,dt = \int_0^1 \frac{t^k(1-t)^{I-k}}{B(k, I-k+1)}\,dt \\ &= \frac{B(k+1, I-k+1)}{B(k, I-k+1)} = \frac{k \cdot \binom{I}{k}}{(k+1)\cdot\binom{I+1}{k+1}} = \frac{k}{I+1}, \end{aligned} \tag{3}$$

which indicates that $I = k/\mu - 1$. □

Remark. When $I < k$, a KMV sketch directly provides the true value of I with the number of random numbers in the sketch. Further, the original estimator $\mathbb{E}[(k-1)/\zeta]$ in Definition 3.1 is unbiased but with only asymptotic error bounds. In contrast, our proposed estimator in Lemma 3.5 utilizes Bernstein's inequality to establish a tighter confidence bound for arbitrary image sizes.

Theorem 3.6 (Bernstein Inequality [3]). *Let $X_1, X_2, \ldots, X_\theta$ be real-valued i.i.d. random variables such that $X_i \in [0, r]$, $\mathbb{E}[X_i] = \mu$, and $\mathrm{Var}[X_i] = \sigma^2$. Let $\overline{X}_\theta = \frac{1}{\theta}\sum_{i=1}^{\theta} X_i$ denote the empirical mean. With probability at least $1-p$, it holds that*

$$|\overline{X} - \mu| \leq \sqrt{\frac{\sigma^2 \log(2/p)}{\theta}} + \frac{2r\log(1/p)}{\theta}.$$

Lemma 3.7. *For any $\delta, p \in (0,1)$ and $u \in \mathcal{V}^*$, if $\theta = O(\frac{n}{\delta k}\log\frac{1}{p})$, $(1-\delta)\cdot I \leq \widetilde{I} \leq (1+\delta)\cdot I$ will hold with probability at least $1-p$.*

Proof. When $I < k$, the exact value of I will always be obtained, i.e. $\widetilde{I} = I$, and the lemma holds trivially. When $I \geq k$, let $\Delta\mu = |\widetilde{\mu} - \mu|$ and $\Delta I = |\widetilde{I} - I|$. According to Lemma 3.5, $I = \frac{k}{\mu} - 1$ and thus

$$\Delta I = \begin{cases} \frac{k}{\mu} - \frac{k}{\mu+\Delta\mu}, & \text{if } \mu < \widetilde{\mu}; \\ \frac{k}{\mu-\Delta\mu} - \frac{k}{\mu}, & \text{if } \mu \geq \widetilde{\mu}. \end{cases}$$

For any $\delta \in (0,1)$, it follows that $\Delta I \leq \delta I$ if $\Delta\mu \leq \frac{\delta I k}{(I+1)(I+1+\delta I)}$. Furthermore, the variance of ζ is

$$\sigma^2 = \mathbb{E}[\zeta^2] - \mu^2 = \frac{k\cdot(k+1)}{(I+1)(I+2)} - \left(\frac{k}{I+1}\right)^2 = \frac{k(I-k+1)}{(I+1)^2(I+2)}. \tag{4}$$

According to Bernstein's inequality, for any $p \in (0,1)$, we have

$$\Delta\mu \leq \sqrt{\frac{\sigma^2\log(2/p)}{\theta}} + \frac{2\log(1/p)}{\theta}$$

with probability at least $1-p$. Thus, we ensure $\Delta I \leq \delta I$ with probability at least $1-p$ by setting θ such that

$$\sqrt{\frac{\sigma^2\log(2/p)}{\theta}} + \frac{2\log(1/p)}{\theta} \leq \frac{\delta I k}{(I+1)(I+1+\delta I)}.$$

By solving the above inequality as a quadratic inequality in terms of $\sqrt{\theta}$, we have $\Delta I \leq \delta I$ with probability at least $1-p$ if

$$\theta \geq \left(\sqrt{\tfrac{I-k+1}{I+2}\log\tfrac{2}{p}} + \sqrt{\tfrac{I-k+1}{I+2}\log\tfrac{2}{p} + 8\tfrac{\delta I(I+1)}{I+1+\delta I}\log\tfrac{1}{p}}\right)^2 \cdot \frac{(I+1+\delta I)^2}{4\delta^2 I^2 k}.$$

By simplifying the above inequality, we obtain $\theta = O(\frac{I}{\delta k} \log \frac{1}{p})$. Since $I \leq n$ for any image, by setting $\theta = O(\frac{n}{\delta k} \log \frac{1}{p})$, $\Delta I \leq \delta \cdot I$ holds with probability at least $1 - p$. □

Theorem 3.8. *Let $\theta = O(\frac{n}{\epsilon k} \log \frac{|V|}{p})$ for any $\epsilon \in (0, 1)$, the sketch propagation framework answers an ϵ-approximate HUB query under degree centrality correctly with probability at least $1 - p$.*

Proof. Let $\widetilde{N}(u)$ denote the estimated degree of node v through sketch propagation. Given any nodes $u \in S_0^+(v^\lambda)$ and $v \in S_\epsilon^-(v^\lambda)$,

$$\widetilde{N}(v) \leq (1+\delta) \cdot N(v) \leq (1+\delta)(1-\epsilon) \cdot N(v^\lambda)$$
$$\leq (1+\delta)(1-\epsilon) \cdot N(u) \leq \frac{(1+\delta)(1-\epsilon)}{1-\delta} \cdot \widetilde{N}(u)$$

hold due to Lemma 3.7 and the definitions of $S_\epsilon^-(v^\lambda)$ and $S_0^+(v^\lambda)$. By setting $\delta < \frac{\epsilon}{2-\epsilon}$, we have $\widetilde{N}(v) < \widetilde{N}(u)$, i.e., the sketches rank v lower than u, with probability at least $1-p$. Taking the union bound over all nodes in $S_0^+(v^\lambda)$ and setting $\theta = O(\frac{n}{\epsilon k} \log \frac{|V|}{p})$, the sketches rank all nodes in $S_0^+(v^\lambda)$ higher than $v \in S_\epsilon^-(v^\lambda)$. Since there are at least $\lambda|V|$ nodes in $S_0^+(v^\lambda)$, v will not be included in the result S. Similarly, given any nodes $u \in S_0^-(v^\lambda)$ and $v \in S_\epsilon^+(v^\lambda)$, we have $\widetilde{N}(v) \geq (1-\delta) \cdot N(v) \geq (1-\delta)(1+\epsilon) \cdot N(v^\lambda) \geq (1-\delta)(1+\epsilon) \cdot N(u) \geq \frac{(1-\delta)(1+\epsilon)}{1+\delta} \cdot \widetilde{N}(u)$. By setting $\delta < \frac{\epsilon}{2+\epsilon}$, it holds that $\widetilde{N}(v) > \widetilde{N}(u)$, i.e. the sketches rank v higher than u, with probability at least $1-p$. Also taking the union bound over all nodes in $S_0^-(v^\lambda)$ and setting $\theta = O(\frac{n}{\epsilon k} \log \frac{|V|}{p})$, the sketches rank all nodes in $S_0^-(v^\lambda)$ lower than $v \in S_\epsilon^+(v^\lambda)$. Since there are at least $(1-\lambda)|V|$ nodes in $S_0^-(v^\lambda)$, v will be included in S. Finally, by taking the union bound over all nodes in $S_\epsilon^+(v^\lambda)$ and $S_\epsilon^-(v^\lambda)$ and setting $\theta = O(\frac{n}{\epsilon k} \log \frac{|V|^2}{p}) = O(\frac{n}{\epsilon k} \log \frac{|V|}{p})$, all nodes in $S_\epsilon^+(v^\lambda)$ are included in S, whereas all nodes in $S_\epsilon^-(v^\lambda)$ are excluded from S, with probability at least $1-p$. □

Finally, the time complexity of the *sketch propagation* framework is $O(\frac{n|\mathcal{E}^*|}{\epsilon} \log \frac{|V|}{p})$. The bottleneck of the *sketch propagation* framework lies in constructing the KMV sketches, which propagates θ KMV sketches of size k over the matching graph with $|\mathcal{E}^*|$ edges.

3.2 Early Termination for Approximate Personalized HUB

Given a query node q, a personalized HUB query can be answered directly using the estimated node degrees through *sketch propagation*. Specifically, if q is ranked higher than the $(1-\lambda)$-quantile node in terms of estimated degree, it returns TRUE for the personalized query, and FALSE otherwise. The following corollary is a direct extension of Theorem 3.8 for personalized queries.

Corollary 3.9. *Let $\theta = O(\frac{n}{\epsilon k} \log \frac{|V|}{p})$ for any $\epsilon \in (0, 1)$. Sketch propagation answers personalized ϵ-approximate HUB queries under degree centrality correctly with probability at least $1 - p$.*

Nevertheless, we can further optimize personalized HUB query processing by terminating the *sketch propagation* early once we have determined that q is not a hub with high confidence. We first give the following lemma to inspire the design of the early termination optimization.

Lemma 3.10. *Given a query node q, if there exists a node $u \in \mathcal{V}^*$ such that $I_f(u) \geq \lambda|V|$ and $I_b(u) > N(q)$, then the answer to the personalized HUB query is* FALSE.

Proof. For any node $u \in \mathcal{V}^*$, it is easy to see that any $v_1 \in \mathcal{I}_f(u)$ and $v_2 \in \mathcal{I}_b(u)$ must be neighbors in H. Hence, each node in $\mathcal{I}_f(u)$ has at least $I_b(u)$ neighbors. Since $I_b(u) > N(q)$ and $I_f(u) \geq \lambda|V|$, there are at least $\lambda|V|$ nodes with higher degrees than $N(q)$. □

Based on Lemma 3.10, we can terminate *sketch propagation* when the estimations $\widetilde{I}_f(u)$ and $\widetilde{I}_b(u)$ of $I_f(u)$ and $I_b(u)$ for a node $u \in \mathcal{V}^*$ by KMV sketches satisfy that $\widetilde{I}_f(u) \geq \lambda|V|$ and $\widetilde{I}_b(u) > N(q)$. However, this may potentially cause a false negative result when $I_f(u)$ and $I_b(u)$ are overestimated. To derive a theoretically robust setting, we terminate sketch propagation when a node u satisfies $\widetilde{I}_f(u) \geq (1+\beta) \cdot \lambda|V|$ and $\widetilde{I}_b(u) > (1+\beta) \cdot N(q)$. By selecting an appropriate value of β, we can ensure that the probability of a false negative is small enough.

Algorithm 2 presents the procedure of OptimizedReceive, an optimized version of Receive at Line 23 in Algorithm 1 with early termination of sketch propagation. Specifically, OptimizedReceive returns whether sketch propagation should be terminated at node u. It first calls Receive at Line 23 of Algorithm 1 to construct the KMV sketch for the backward image of u (Line 1) and then estimates $I_f(u)$ and $I_b(u)$ accordingly (Lines 2–12). Finally, it returns whether the estimated $\widetilde{I}_f(u)$ and $\widetilde{I}_b(u)$ have satisfied the early termination conditions (Lines 13–15).

Algorithm 2: OptimizedReceive

Input: Node u, arrays of KMV sketches $\mathcal{K}_f, \mathcal{K}_b$
Output: If sketch propagation can be terminated

1 Receive$(u, \mathcal{K}_b, +)$;
2 $\widetilde{I}_f \leftarrow 0$ and $\widetilde{I}_b \leftarrow 0$;
3 **for** $t = 1$ **to** θ **do**
4   **if** $|\mathcal{K}_f[u][t]| = k$ **then**
5     $\widetilde{I}_f \leftarrow \widetilde{I}_f + \frac{\max(\mathcal{K}_f[u][t])}{\theta}$;
6   **else**
7     $\widetilde{I}_f \leftarrow \widetilde{I}_f + \frac{k}{(|\mathcal{K}_f[u][t]|+1)\cdot\theta}$;
8   **if** $|\mathcal{K}_b[u][t]| = k$ **then**
9     $\widetilde{I}_b \leftarrow \widetilde{I}_b + \frac{\max(\mathcal{K}_b[u][t])}{\theta}$;
10   **else**
11     $\widetilde{I}_b \leftarrow \widetilde{I}_b + \frac{k}{(|\mathcal{K}_b[u][t]|+1)\cdot\theta}$;
12 $\widetilde{I}_f \leftarrow k/\widetilde{I}_f - 1$ and $\widetilde{I}_b \leftarrow k/\widetilde{I}_b - 1$;
13 **if** $\widetilde{I}_f \geq (1+\beta) \cdot \lambda|V|$ and $\widetilde{I}_b \geq (1+\beta) \cdot N(q)$ **then**
14   **return** TRUE;
15 **return** FALSE;

Example 3.11. Continuing with Example 3.4 where $\mathcal{K}(\mathcal{I}_b(v_0)) = \{0.1, 0.15\}$, the algorithm estimates $\widetilde{I}_b(v_0) = \frac{2}{0.15} - 1 \approx 12.3$. The size $\widetilde{I}_f(v_0) = \frac{2}{0.3} - 1 \approx 5.7$ is estimated from the sketch $\mathcal{K}(\mathcal{I}_f(v_0))$ in Figure 2(a). Assuming $\lambda = 0.01$, $|V| = 500$, $N(q) = 9$ and setting $\beta = 0$, we have $\widetilde{I}_f(u_1) > 0.01 \cdot 500$ and $\widetilde{I}_b(u_1) > 9$, and thus sketch propagation can be early terminated.

Choosing Appropriate β. The following theorem provides an upper bound for β so that the probability that early termination yields a false negative result is at most p_e for any $p_e \in (0, 1)$.

THEOREM 3.12. *If $\theta = O(\frac{n}{\epsilon k} \log \frac{|V|}{p})$ and $\beta = \frac{\log p_e - \log |\mathcal{V}^*|}{\log p - \log |V|} \cdot \epsilon$, the probability that a personalized HUB query whose answer is* TRUE *is incorrectly answered with* FALSE *is at most p_e.*

PROOF. Early termination occurs only when there exists a node $u \in \mathcal{V}^*$ such that $\widetilde{I}_f(u) \geq (1+\beta) \cdot \lambda |V|$ and $\widetilde{I}_b(u) > (1+\beta) \cdot N(q)$. Thus, sketch propagation will not be early terminated incorrectly if

$$\widetilde{I}_f(u) < (1+\beta) \cdot \lambda |V| \text{ or } \widetilde{I}_b(u) \leq (1+\beta) \cdot N(q), \ \forall u \in \mathcal{V}^*. \quad (5)$$

For any given $\delta' > 0$, we have

$$O(\frac{n}{\epsilon k} \log \frac{|V|}{p}) = O(\frac{n}{(\frac{\log \delta' p}{\log (p/|V|)})\epsilon k} \log \frac{1}{\delta' p}) \quad (6)$$

Combining Theorem 3.7 and Eqn. 6, if $\theta = O(\frac{n}{\epsilon k} \log \frac{|V|}{p})$, we have

$$\widetilde{I}_f(u) \leq (1+\frac{\log \delta' p}{\log(p/|V|)} \cdot \epsilon) \cdot I_f(u); \ \widetilde{I}_b(u) < (1+\frac{\log \delta' p}{\log(p/|V|)} \cdot \epsilon) \cdot I_b(u).$$

Each holds with probability at least $1 - \delta' p$ for a given node u. Let $\delta' = \frac{p_e}{p|\mathcal{V}^*|}$ and $\beta = \frac{\log p_e - \log |\mathcal{V}^*|}{\log p - \log |V|} \cdot \epsilon$. Taking the union bound,

$$\widetilde{I}_f(u) \leq (1+\beta) \cdot I_f(u) \text{ or } \widetilde{I}_b(u) < (1+\beta) \cdot I_b(u), \ \forall u \in \mathcal{V}^* \quad (7)$$

holds with probability at least $1 - p_e$.

According to Lemma 3.10, since the answer to a personalized HUB query is TRUE, $I_f(u) < \lambda |V|$ or $I_b(u) \leq N(q), \forall u \in \mathcal{V}^*$. Combined with Eqn. 7, Eqn. 5 holds with probability $1 - p_e$. □

4 EXTENSION TO H-INDEX

In this section, we extend the sketch propagation framework to process approximate HUB queries based on the h-index measure. To compute the h-indexes of the nodes in H, we need to compute the degrees of their neighbors. However, the KMV sketches can only estimate the neighborhood sizes of each node but fail to provide the degree estimations of their neighbors. Therefore, we propose a novel pivot-based algorithm that can estimate the set of hubs in terms of h-index without explicitly calculating the h-index of every node in H. Before diving into the algorithm, we first review the definition of *h-index* [22].

Definition 4.1 (H-index). Given a node $u \in H$, the h-index of u is the largest value $h(u)$ such that u has no fewer than $h(u)$ neighbors of degrees at least $h(u)$.

4.1 Pivot-based Method for Approximate HUB

The basic idea of the pivot-based algorithm is to choose a random pivot node $u \in V$ and iteratively partition the nodes in H into two disjoint sets, Q_u^+ and Q_u^-, based on their h-index values in comparison with $h(u)$, i.e., $Q_u^+ = \{v \in V \mid h(v) \geq h(u)\}$ and $Q_u^- = \{v \in V \mid h(v) < h(u)\}$. As such, we can decide that $Q_u^+ \subseteq S_0^+(v_\lambda)$ if $u \in S_0^+(v_\lambda)$ since all nodes in Q_u^+ have h-index values greater than or equal to $h(u)$ and thus are ranked higher than u, or $Q_u^- \cap S_0^+(v_\lambda) = \varnothing$ if $u \in S_0^-(v_\lambda)$ similarly. Such a partitioning strategy allows for efficient bisection when identifying hubs, i.e., $S_0^+(v^\lambda)$. If $u \in S_0^+(v_\lambda)$, we can add all nodes in Q_u^+ to the result set and exclude them from consideration in the subsequent iteration. On the contrary, if $u \in S_0^-(v^\lambda)$, the nodes in Q_u^+ may contain those in both $S_0^+(v^\lambda)$ and $S_0^-(v^\lambda)$, which require further partitioning in next iterations. Meanwhile, all nodes in Q_u^- must not be in $S_0^+(v^\lambda)$ and are excluded from consideration. The above iterative process proceeds until all nodes have been decided. Then, all the nodes in $S_0^+(v^\lambda)$ have been added to the result set.

Pivot-based Partitioning Method. The key challenge lies in partitioning the nodes in H without explicitly computing the h-index of each node. In fact, we can compare the h-indexes of two nodes using only one of them. By the definition of h-index, for any two nodes $u, v \in V$, we have $h(v) \geq h(u)$ if and only if v has at least $h(u)$ neighbors in H with degrees larger than or equal to $h(u)$. Based on the above observation, we introduce a two-stage pivotal partitioning method for our pivot-based algorithm.

Stage (I): Estimating $h(u)$. We first employ *sketch propagation* on the matching graph to estimate the degrees of all nodes in H. Subsequently, at each iteration, we scan through the set of neighbors $\mathcal{N}(u)$ of a randomly chosen node u (obtained by a single DFS on the matching graph starting from u). Based on the degree estimations, we compute an approximate h-index $\widetilde{h}(u)$ of node u.

Stage (II): Estimating Q_u^+ and Q_u^-. Once we have the approximate h-index $\widetilde{h}(u)$, we proceed to eliminate nodes in H with degrees smaller than $\widetilde{h}(u)$. Next, we perform *sketch propagation* on the subgraph of the matching graph that only includes the remaining nodes. The resulting KMV sketches estimate the degrees of nodes in the subgraph of H induced by the set of nodes with degrees greater than or equal to $\widetilde{h}(u)$ in the original graph H, denoted as $\widetilde{H}(u)$. Finally, we obtain Q_u^+ as the set of nodes with estimated degrees greater than $\widetilde{h}(u)$ in the induced subgraph of $\widetilde{H}(u)$, while the remaining nodes are added to Q_u^-.

Algorithm 3 depicts the procedure of our pivot-based algorithm. It also receives a knowledge graph $\mathcal{G}$, a meta-path $\mathcal{M}$, the quantile parameter λ, the number of propagations θ, and the KMV sketch size k as input, and returns a set of nodes S to answer the (approximate) HUB query based on h-index. It first calls the same Propagate function as Algorithm 1 over the arrays $\mathcal{K}_f$ and $\mathcal{K}_b$ of KMV sketches to estimate the degree of each node in H (Lines 1–5). Note that the estimated degrees will be reused across all iterations for partitioning. The iterative process proceeds until Q, which contains the nodes to be partitioned in each iteration, is empty (Lines 7–26). In each iteration, it first randomly chooses a pivot node u from Q and estimates its h-index $\widetilde{h}(u)$ (Lines 8-12). Then, it re-initializes arrays $\mathcal{K}_f$ and $\mathcal{K}_b$ of KMV sketches for the nodes with degrees greater than $\widetilde{h}(u)$ (Lines 13–17). After that, the Propagate function is performed over $\mathcal{K}_f$ and $\mathcal{K}_b$ to estimate the number of neighbors with degrees larger than $\widetilde{h}(u)$ for each node in Q (Line 18). Accordingly, it partitions Q into Q_u^+ and Q_u^- according to $\widetilde{h}$ (Lines 19–21) and updates the result set S and the candidate node set Q based on Q_u^+ and Q_u^- (Lines 22–26). Finally, when Q is empty, the result set S is returned (Line 27). The tied values are broken arbitrarily.

Theoretical Analysis. Next, we prove that the pivot-based algorithm returns the correct answer to an approximate HUB query

Algorithm 3: Pivot-based Algorithm

Input: Knowledge graph $\mathcal{G}$, meta-path $\mathcal{M}$, quantile parameter λ, number of propagations θ, KMV sketch size k

Output: The set of nodes S for (approximate) HUB query

forall $u \in \mathcal{V}^*$ **do**
  for $t = 1$ **to** θ **do**
    $\mathcal{K}_f[u][t] \leftarrow \emptyset$ and $\mathcal{K}_b[u][t] \leftarrow \emptyset$;
    if $u \in \mathcal{V}_0$ **do** add $r(u) = \mathsf{Rand}(0,1)$ to $\mathcal{K}_f[u][t]$;
$\widetilde{N} \leftarrow \mathsf{Propagate}(\mathcal{K}_f, \mathcal{K}_b)$;
$Q \leftarrow V$ and $S \leftarrow \emptyset$;
while $|Q| > 0$ **do**
  Sample a random node from Q as the pivot node u;
  Sort the neighbors $\mathcal{N}(u)$ of u in descending order by $\widetilde{N}$;
  Initialize $\widetilde{h}(u) \leftarrow 0$;
  forall $v \in \mathcal{N}(u)$ **do**
    if $\widetilde{N}(v) \geq \widetilde{h}(u)$ **then** $\widetilde{h}(u) \leftarrow \widetilde{h}(u) + 1$;
  forall $v \in \mathcal{V}^*$ **do**
    for $t = 1$ **to** θ **do**
      $\mathcal{K}_f[v][t] \leftarrow \emptyset$ and $\mathcal{K}_b[v][t] \leftarrow \emptyset$;
      if $v \in \mathcal{V}_0$ *and* $\widetilde{N}(v) \geq \widetilde{h}(u)$ **then**
        Add $r(v) = \mathsf{Rand}(0,1)$ to $\mathcal{K}_f[v][t]$;
  $\widetilde{h} \leftarrow \mathsf{Propagate}(\mathcal{K}_f, \mathcal{K}_b)$;
  Initialize $Q_u^+ \leftarrow \emptyset$ and $Q_u^- \leftarrow \emptyset$;
  forall $v \in Q$ **do**
    if $\widetilde{h}(v) \geq \widetilde{h}(u)$ **then** add v to Q_u^+ **else** add v to Q_u^-;
  if $|Q_u^+| + |S| \leq \lambda|V|$ **then**
    $Q \leftarrow Q_u^-$ and $S \leftarrow S \cup Q_u^+$;
    if $|S| \leq \lambda|V|$ **then** $S \leftarrow S \cup \{u\}$;
  else
    $Q \leftarrow Q_u^+$;
return S;

based on h-index w.h.p. We first give the following lemma, which states that Algorithm 3 provides a concentrated estimation of $h(u)$ for the randomly chosen pivot node u in each iteration.

Lemma 4.2. *For any $\delta, p \in (0,1)$, if $\theta = O(\frac{n}{\delta k} \log \frac{|V|}{p})$, we have $(1-\delta) \cdot h(u) \leq \widetilde{h}(u) \leq (1+\delta) \cdot h(u)$ with probability at least $1-p$ for the pivot node u in each iteration.*

Proof. Let us order the nodes in $\mathcal{N}(u)$ according to their degrees in H descendingly and denote v_i as the i-th neighbor of u. Then, we have $N(v_i) \geq h(u)$ for $0 \leq i \leq h(u)-1$ and $N(v_i) \leq h(u)$ for $h(u) \leq i \leq N(u)-1$.

By combining Lemma 3.7 with the union bound over V, when $\theta = O(\frac{n}{\delta k} \log \frac{|V|}{p})$ in the first stage, we have $\widetilde{N}(v_i) \geq (1-\delta)N(v_i) \geq (1-\delta)h(u)$ for $0 \leq i \leq h(u)-1$ with probability at least $1-p$. Thus, node u has $h(u) \geq (1-\delta)h(u)$ neighbors with estimated degrees larger than or equal to $(1-\delta)h(u)$. Thus, $\widetilde{h}(u) \geq (1-\delta)h(u)$ with probability at least $1-p$. Similarly, we have $\widetilde{N}(v_i) \leq (1+\delta)N(v_i) \leq (1+\delta)h(u)$ for $i \geq h(u)$ with probability at least $1-p$. Thus, node u has no more than $h(u)$ neighbors with estimated degrees larger than $(1+\delta)h(u)$, and we have $\widetilde{h}(u) \leq (1+\delta)h(u)$ with probability at least $1-p$. □

Then, the following lemma indicates that the pivot-based partitioning method can divide candidate nodes into ϵ-separable sets.

Lemma 4.3. *For any $\delta' \in (0,1)$ and pivot node u, by setting $\theta = O(\frac{n}{\delta' k} \log \frac{|V|}{p})$, we have $S^-_{\delta'}(u) \subseteq Q_u^-$ and $S^+_{\delta'}(u) \subseteq Q_u^+$ hold with probability at least $1-p$.*

Proof. For any node $v \in S^-_{\delta'}(u)$, let $\mathcal{N}_u(v)$ denote the neighbors of v in $\widetilde{H}(u)$ and $N_u(v) = |\mathcal{N}_u(v)|$. Since $h(v) < (1-\delta')h(u)$, v has at most $(1-\delta')h(u)$ neighbors with degrees larger than or equal to $(1-\delta')h(u)$. For each neighbor v' of v with $N(v') < (1-\delta')h(u)$, when $\theta = O(\frac{n}{\delta k} \log \frac{|V|}{p})$, $\widetilde{N}(v') < (1+\delta)N(v') < (1+\delta)(1-\delta')h(u) < \frac{(1+\delta)(1-\delta')}{1-\delta}\widetilde{h}(u)$. By setting $\delta < \frac{\delta'}{2-\delta'}$, we have $\widetilde{N}(v') < \widetilde{h}(u)$, i.e., v' is filtered out in *Stage (I)*, with probability at least $1-p$. Thus, we have $N_u(v) < (1-\delta')h(u)$ with probability at least $1-p$. Let $\widetilde{N}_u(v)$ denote the estimation of $N_u(v)$ in *Stage (II)* of pivot-based partitioning. We have $\widetilde{N}_u(v) \leq (1+\delta)N_u(u)$ with probability at least $1-p$, and thus $\widetilde{N}_u(v) \leq (1+\delta)N_u(v) < (1+\delta)(1-\delta')h(u) \leq \frac{(1+\delta)(1-\delta')}{1-\delta}\widetilde{h}(u) \leq \widetilde{h}(u)$, i.e., v is added into Q_u^- with probability at least $1-p$. Thus, by setting $\theta = O(\frac{n}{\frac{\delta'}{2-\delta'}k} \log \frac{|V|}{p}) = O(\frac{n}{\delta' k} \log \frac{|V|}{p})$, we have $S^-_{\delta'}(u) \subseteq Q_u^-$. Similarly, we also get $S^+_{\delta'}(u) \subseteq Q_u^+$ with probability at least $1-p$ when $\theta = O(\frac{n}{\delta' k} \log \frac{|V|}{p})$. □

Subsequently, we can formally analyze the correctness of the pivot-based algorithm for processing ϵ-approximate HUB queries based on the h-index in the following theorem.

Theorem 4.4. *Let $\theta = O(\frac{n}{\epsilon k} \log \frac{|V|}{p})$ for any $\epsilon \in (0,1)$, the pivot-based algorithm returns the answer to an ϵ-approximate HUB query based on the h-index correctly with probability at least $1-p$.*

Proof. We prove the theorem by examining three different cases of the ranges of the h-index $h(u)$ for the pivot node u. For each case, we show the correctness of the first iteration of the pivot-based algorithm, while subsequent iterations are proven by induction. By selecting $\delta' < \frac{\epsilon}{2}$ as Lemma 4.3, we establish that:

- **Case 1 ($h(u) > (1+\frac{\epsilon}{2})h(v^\lambda)$):** Each v with $h(v) \leq h(v^\lambda)$ will be added to Q_u^-. Thus, $|Q_u^-| \geq (1-\lambda)|V|$. Therefore, Q_u^- will be used as the candidate set Q in the next iteration, and Q^+, which does not contain any node in $S^-_\epsilon(v^\lambda)$, will be added to S.
- **Case 2 ($h(u) < (1-\frac{\epsilon}{2})h(v^\lambda)$):** Each v with $h(v) \geq h(v^\lambda)$ will be added to Q_u^+. Thus, $|Q_u^+| \geq \lambda|V|$. Therefore, Q^+, which contains all nodes in $S^+_\epsilon(v^\lambda)$, will be used as the candidate set Q in the next iteration.
- **Case 3 ($(1-\frac{\epsilon}{2})h(v^\lambda) < h(u) < (1+\frac{\epsilon}{2})h(v_\lambda)$):** Any v with $h(v) \geq (1+\epsilon)h(v^\lambda)$ will be added to Q_u^+. And any v' with $h(v') \leq (1-\epsilon)h(v^\lambda)$ will be added to Q_u^-. Thus, Q_u^+, which contains all nodes in $S^+_\epsilon(v^\lambda)$, will be either added to S or used as the candidate set Q in the next iteration; whereas Q_u^-, which contains all nodes in $S^-_\epsilon(v^\lambda)$, will be either eliminated or used as the candidate set Q for the next iteration.

Considering all the above cases, it follows that, for any randomly chosen pivot u, none of the nodes in $S^-_\epsilon(v^\lambda)$ is added to the result set S. In contrast, all nodes in $S^+_\epsilon(v^\lambda)$ have a probability of at least $1-p$ to be either added to S or retained for the subsequent iteration.

Consequently, we prove the theorem by applying the union bound over all iterations. ◻

Finally, the amortized time complexity of the pivot-based algorithm is $O(\frac{n|\mathcal{E}^*|}{\epsilon}\log^2\frac{|V|}{p})$ since it invokes the sketch propagation framework $O(\log|V|)$ times in amortization.

4.2 Early Termination for Approximate Personalized HUB

Given a query node q, the personalized HUB query based on the h-index can also be answered by directly using the query node q as the pivot node and performing the pivot-based partitioning method to obtain Q_u^+. Then, it simply returns FALSE if $|Q_u^+| > \lambda|V|$ or TRUE otherwise. Following Lemma 4.3, the following corollary indicates that the above algorithm provides correct answers for approximate personalized HUB queries w.h.p.

Corollary 4.5. *For any $\epsilon \in (0,1)$, by setting $\theta = O(\frac{n}{\epsilon k}\log\frac{|V|}{p})$, the pivot-based algorithm answers an ϵ-approximate personalized HUB query based on h-index with probability at least $1-p$.*

Early termination. An early termination optimization can also accelerate personalized HUB queries based on h-index.

Lemma 4.6. *Given a query node q and the quantile parameter λ, if there exists a node $u \in \mathcal{V}^*$ such that $I_f(u) > h(q)$ and $I_b(u) \geq \max(h(q), \lambda|V|)$, then the answer to the personalized HUB query must be* FALSE.

Proof. Note that each pair of nodes in $\mathcal{I}_b(u)$ and $\mathcal{I}_f(u)$ are a neighbor of each other in H according to the proof of Lemma 3.10. Thus, each node $v_1 \in \mathcal{I}_b(u)$ has at least $I_f(u)$ neighbors with degrees larger than or equal to $I_b(u)$. Since $I_f(u) > h(q)$ and $I_b(u) \geq \max(h(q), \lambda\cdot|V|)$, there exists at least $I_b(u) \geq \lambda|V|$ nodes with h-indexes larger than or equal to $h(q)$. Therefore, q is not a hub in terms of h-index. ◻

Based on Lemma 4.6, sketch propagation can be early terminated during *Stage (I)* of the pivot-based partitioning method. Specifically, we use the KMV sketches to estimate $I_f(u)$ and $I_b(u)$ for each $u \in \mathcal{V}^*$. If there exists a node $u \in \mathcal{V}^*$ such that $\widetilde{I}_f(u) > (1+\beta)N(q)$ and $\widetilde{I}_b(u) \geq (1+\beta)\max(N(q), \lambda|V|)$, the algorithm can be early terminated since $N(q)$ is an upper bound of $h(q)$.

Once we obtain $\widetilde{h}(q)$ after *Stage (I)*, the algorithm can also be terminated without entering *Stage (II)* if there exists a node $u \in \mathcal{V}^*$ such that $\widetilde{I}_f(u) > (1+\beta)\widetilde{h}(q)$ and $\widetilde{I}_b(u) \geq (1+\beta)\max(\widetilde{h}(q), \lambda|V|)$. The following theorem provides an upper bound of β that assures the probability of false negatives caused by early termination is at most $p_e \in (0,1)$.

Theorem 4.7. *If $\theta = O(\frac{n}{\epsilon k}\log\frac{|V|}{p})$ and $\beta = \frac{\log p_e - \log|\mathcal{V}^*|}{\log p - \log|V|}\cdot\epsilon$, the pivot-based algorithm with early termination answers a personalized HUB query with probability at least $1-p_e$.*

Proof. Early termination occurs during *Stage (I)* of the pivot-based partitioning method only when there exists a node $u \in \mathcal{V}^*$ such that $\widetilde{I}_f(u) \geq (1+\beta)\cdot N(q)$ and $\widetilde{I}_b(u) \geq (1+\beta)\cdot N(q)$ and $\widetilde{I}_b(u) \geq (1+\beta)\cdot\lambda|V|$. Since we have $\widetilde{I}_b(u) \geq \widetilde{I}_f(u)$, the pivot-based partitioning will not be early terminated incorrectly if

$$\widetilde{I}_f(u) < (1+\beta)\cdot N(q) \text{ or } \widetilde{I}_b(u) < (1+\beta)\cdot\lambda|V|,\ \forall u \in \mathcal{V}^* \quad (8)$$

Based on Eqn. (8), the theorem can be proved in the exactly same way for proving Theorem 3.12.

When the early termination takes place after the first stage of pivot-based partitioning based on $\widetilde{h}(q)$, the theorem can be proved by combining Lemma 4.2 with Eqn. (6). ◻

Table 1: Statistics of KGs used in the experiments, where $|\mathcal{L}(\mathcal{V})|$ and $|\mathcal{L}(\mathcal{E})|$ denote the numbers of node and edge types, and $|\mathbb{M}|$ is the number of evaluated meta-paths.

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{L}(\mathcal{V})\rvert$	$\lvert\mathcal{L}(\mathcal{E})\rvert$	$\lvert\mathbb{M}\rvert$
IMDB	21.4K	86.6K	4	6	679
ACM	10.9K	548K	4	8	20
DBLP	26.1K	239.6K	4	6	48
PubMed	63.1K	236.5K	4	10	172
FreeBase	180K	1.06M	8	36	151
Yelp	82.5K	29.9M	4	4	2230

5 EXPERIMENTS

5.1 Experimental Setup

Datasets. We conduct our experiments on six real-world KGs. The statistics of those KGs are reported in Table 1. IMDB is a KG about actors and directors of movies. ACM and DBLP are two academic KGs denoting the co-authorships between researchers through papers and publishing venues. PubMed is a biomedical KG representing the relationships between genes, chemicals, and diseases of different species. FreeBase is extracted from a general-purpose KG developed by Google[1]. Yelp is a business KG recording the user ratings and comments on businesses at different locations.

Query Generation. For each dataset, we use AnyBURL [23] to extract a set of candidate meta-paths $\mathbb{M}$. We set the confidence threshold to 0.1 for AnyBURL and take the snapshot at ten seconds. Note that AnyBURL can extract meta-paths with extra node constraints, resulting in a large number of potential meta-paths for validation. This makes the efficient processing of HUB queries even more required. $|\mathbb{M}|$ in Table 1 shows the number of meta-paths found on each dataset. For personalized HUB queries, we randomly sample 10 nodes from $\mathcal{V}_0$ as the query nodes for each meta-path.

Compared Methods. To the best of our knowledge, there has not yet been any prior study on the HUB problem. Therefore, we only compare our sketch propagation-based algorithms with the exact algorithm based on the generic worst-case optimal join.

- ExactD and ExactH use the exact algorithms in Section 2.2 to compute the degrees and h-indexes of all nodes in H, thereby identifying the hubs by sorting the nodes in descending order of degree and h-index.
- GloD and PerD apply sketch propagation in Section 3.1 to estimate the degrees of all nodes in H. GloD returns the set of hubs based on estimated degrees, whereas PerD decides if a query node q is a hub.

[1] https://developers.google.com/freebase

- PerD$^+$ optimizes PerD with early termination in Section 3.2.
- GloH and PerH apply the pivot-based algorithm in Section 4.1 to find the hubs based on estimated h-indexes. GloH recursively partitions the nodes until all hubs are found, whereas PerH uses a query node q as the pivot to check whether it is a hub.
- PerH$^+$ optimizes PerH with early termination in Section 4.2.

Parameter Settings. By default, we set $\lambda = 0.05$ to find the hubs as the top 5% of nodes with the highest degrees or h-indexes in hidden networks. Our results in Section 5.3 show that h-index-based HUB queries are well approximated using smaller values of θ and k compared to degree-based HUB queries. Therefore, for algorithms targeting degree-based HUB queries (GloD, PerD, PerD$^+$), we set $\theta = 8$ and $k = 32$ by default, while for algorithms targeting h-index-based HUB queries (GloH, PerH, PerD$^+$), we set $\theta = 8$ and $k = 4$ by default. We use $\beta = 0$ as the default setting since it produces correct early terminations for personalized queries across different datasets, as shown in Section 5.3. These settings also demonstrate that our proposed methods can yield accurate results in practice with significantly less stringency than those outlined in the worst-case analysis.

Evaluation Metrics. We evaluate the quality of hubs each algorithm returns for global HUB queries (GloD and GloH) with the F1-score. Our algorithms return a node set S containing $\lambda \cdot |V|$ nodes. However, there might be more than $\lambda \cdot |V|$ nodes satisfying the criteria of Problem 1 due to tied values with v^λ. As such, we define *precision* as $|S \cap S^*|/|S|$ and *recall* as $|S \cap S^+|/|S^+|$ for F1-score computation. Here, S^* represents the set of nodes with centralities at least the same as that of v^λ, and S^+ represents the set of nodes with strictly higher centralities than v^λ. In effect, we exclude the nodes with tied centralities to v^λ in the recall calculation. To assess the effectiveness of PerD, PerD$^+$, PerH, and PerH$^+$ for personalized HUB queries, we record their accuracy in determining whether a query node q has equal or greater centrality compared to v^λ. We report the average F1 and accuracy scores across all meta-paths in Table 1. For efficiency evaluation, we report the average running time of each algorithm to process one HUB query.

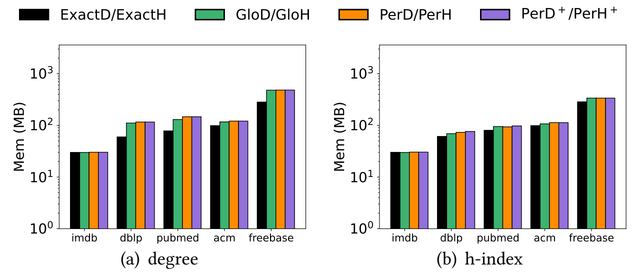
Environment. All experiments were conducted on a Linux Server with AMD EPYC 7643 CPU and 256 GB memory running Ubuntu 20.04. All algorithms were implemented in C++ with `-O3` optimization and executed in a single thread. We release the source code in an anonymous repository[2].

5.2 Efficiency Evaluation

In Table 2, we present the execution time per query for each method, along with its speedup ratios compared to the corresponding exact method across five datasets. We also report the average time to construct the matching graph, which is nearly negligible compared with the running time of each algorithm. Based on these results, we make the following observations.

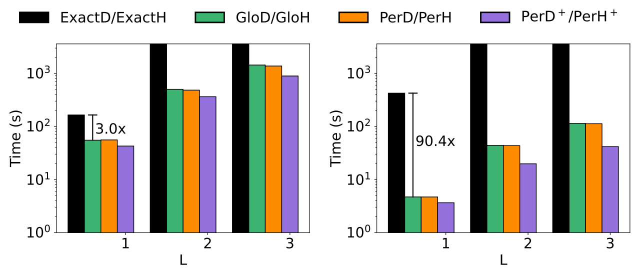
First, our sketch propagation algorithms are consistently more efficient than the exact baselines across all datasets except IMDB and provide more than 10x speedups for degree-based global queries (on FreeBase, GloD achieves a 29x speedup over ExactD) and upto two-orders of magnitude speedups for h-index-based global queries

[2] https://anonymous.4open.science/r/HUD-0B7A/source/readme.md



(a) degree (b) h-index

Figure 3: Memory consumption of different methods across datasets.



(a) degree (b) h-index

Figure 4: Running times of different algorithms on Yelp for meta-paths with varying length L.

(on PubMed, GloH achieves a 135.7x speedup over ExactH). The speedup over degree-based global queries is smaller than h-index-based ones since h-index-based queries are approximated accurately with smaller k and θ as discussed in Section 5.3. GloD and GloH are slightly slower than ExactD and ExactH for degree-based queries on IMDB since IMDB is a very small dataset, where the worst-case optimal join can be performed quickly over the matching graph, whereas our methods require additional costs for sketch construction and degree estimation. Nevertheless, GloD answers HUB queries on IMDB in 1.3ms.

Second, the algorithms for personalized queries, i.e., PerD$^+$ and PerH$^+$, benefit from early termination optimization and achieve more than 2x additional speedups over PerD and PerH. For degree-based personalized queries, backward propagation, which can be early terminated, is more time-consuming than forward propagation since forward sketches usually contain fewer random numbers than backward sketches.

We then report the memory consumption of different methods in Figure 3. We can observe from Figure 3(a) that GloD, PerD and PerD$^+$ take relatively more memory than the exact algorithm ExactD in order to maintain KMV sketches for images. Nevertheless, our methods take at most 1.93 times memory than ExactD (on DBLP). On the other hand, the gap between memory consumption for GloH, PerH, PerH$^+$ and the memory consumption of ExactH is marginal since h-index based queries require less KMV sketches for accurate estimation. Specifically, our methods take at most 1.21 times memory than ExactH (on PubMed).

Scalability. On the largest dataset Yelp (with 29.9M edges and more than 2000 candidate meta-paths), ExactD and ExactH cannot be finished within 24 hours. To evaluate the efficiencies of different

Table 2: Overall results for the efficiencies of different algorithms. For ExactD and ExactH, the running time on each dataset is reported. For GloD,PerD$^+$,GloH, and PerH$^+$, the running time, as well as the speedup ratios over ExactD and ExactH, are reported. The average time to construct the matching graph $\mathcal{G}^*$ is reported as $T(\mathcal{G}^*)$.

Centrality	Degree							H-index							$T(\mathcal{G}^*)$
Method	ExactD	GloD		PerD		PerD$^+$		ExactH	GloH		PerH		PerH$^+$		
IMDB	0.0009s	0.0013s	0.66x	0.0016s	0.55x	0.0017s	0.53x	0.0013s	0.001s	1.23x	0.0015s	0.82x	0.0015s	0.81x	0.88ms
ACM	4.77s	1.77s	2.7x	2.04s	2.34x	0.81s	5.91x	8.01s	0.19s	41.7x	0.25s	32x	0.076s	105x	4.18ms
DBLP	7.95s	0.63s	12.59x	0.67s	11.79x	0.27s	29x	14.59s	0.23s	64.32x	0.1s	148x	0.049s	298x	3.58ms
PubMed	5.96s	0.45s	13.24x	0.35s	17.05x	0.13s	45x	12.11s	0.09s	135.7x	0.056s	215x	0.032s	381x	8ms
FreeBase	25.15s	0.87s	29x	0.92s	27.4x	0.41s	60.9x	50.5s	0.26s	191x	0.19s	268x	0.13s	379x	26ms

algorithms, we sampled ten meta-paths of different lengths ($L = 1, 2, 3$) and ran each method within a one-hour limit per meta-path. Figure 4 shows that our proposed methods process all queries within the time limit, while the exact algorithms exceed the time limit for meta-paths of length $L = 2, 3$.

5.3 Effectiveness Evaluation

Results for Degree-based HUB Queries. Figures 5(a)–5(e) illustrate the effectiveness of GloD, PerD, and PerD$^+$ on each dataset with varying the parameter λ from 0.01 to 0.05. We first observe that GloD achieves F1-scores of more than 0.85 across all datasets for global HUB queries, and PerD and PerD$^+$ consistently achieve at least 0.98 accuracy for personalized HUB queries. Moreover, GloD provides more accurate answers with increasing λ and achieves over 0.9 F1-score when $\lambda = 0.05$. The reason is that, for a larger λ, the $(1-\lambda)$-quantile node v^λ has a smaller degree, and thus the HUB queries apply less strict requirements to hubs, which are more easily approximated.

We also experiment with greater values of λ ranging from 0.1 to 0.5, reported in Figures 6(f)-6(j). We can observe that the gap between effectiveness of PerD and PerD$^+$ increases with the increase of λ. Nevertheless, PerD and PerD$^+$ still consistently achieve accuracy above 0.9 for personalized HUB queries.

Figures 5(k)–5(o) show the effectiveness of GloD, PerD, and PerD$^+$ on each dataset with varying the parameter k. First, GloD, PerD, and PerD$^+$ all provide better results with increasing k. Larger KMV sketches allow the sketch propagation-based methods to approximate HUB queries with higher F1-scores or accuracy owing to more accurate degree estimations. Second, personalized HUB queries are accurately approximated by PerD and PerD$^+$, even with relatively smaller values of k. This is because personalized HUB queries only require comparing the degrees of q and v^λ. For the default setting of a relatively small $\lambda = 0.05$, most nodes in G have significantly different degrees from the degree of v^λ, facilitating the comparison.

Third, the gap between the accuracy of PerD$^+$ and PerD is marginal, even when $k = 2$ (with a maximum gap of 0.044 on DBLP), and narrows further with increasing k. This is because a larger k leads to more accurate estimations of image sizes, thus reducing the chance of false early termination.

Figures 5(p)–5(t) demonstrate the effectiveness of GloD, PerD, and PerD$^+$ on each dataset with varying parameter θ. First, we still observe that GloD returns monotonically better results with increasing θ since HUB queries are better approximated with more KMV sketches. We also observe that GloD is less sensitive to θ than k. This is attributed to the default setting of $k = 32$, which already provides a reasonably accurate approximation. For personalized HUB queries, PerD and PerD$^+$ consistently achieve an accuracy of at least 0.95 across all datasets, regardless of the changes in θ.

Results for HUB Queries based on H-index. The effectiveness results of GloH, PerH, and PerH$^+$ are presented in Figure 6. Similar observations can be made for the methods for h-index-based HUB queries as those for degree-based HUB queries. Furthermore, we note that GloH achieves a higher F1-score for h-index-based HUB queries compared to GloD under the same parameter settings. This is because the range of h-index values is much smaller than that of degrees, resulting in a larger number of tied h-index values. By excluding the nodes with tied values to v^λ from the recall evaluation, h-index-based HUB queries become much easier to approximate.

Summary. GloD, PerD, and PerD$^+$ achieve F1-scores or accuracy of above 0.9 across all datasets when $k = 32$ and $\theta = 8$ for degree-based HUB queries with $\lambda = 0.05$. Similarly, for h-index-based HUB queries, GloH, PerH, and PerH$^+$ achieve F1 scores or accuracy of above 0.9 across all datasets when $k = 4$ and $\theta = 8$. All these results establish the effectiveness of our proposed methods in the default parameter settings.

6 RELATED WORK

Hubs in Hidden Networks. Previous work has extensively explored the problem of finding hubs in hidden networks, where the existence of any edge can only be decided via *probe operations*. Tao et al. [36] proposed two instance-optimal exact algorithms over two kinds of probing strategies of hidden networks. Wang et al. [39] extended this problem to include group testing, whereby probes can verify edge connections between multiple nodes. Sheng et al. [28] proposed a sampling algorithm for hub queries, which Cao et al. [6] subsequently improved. Strouthopoulos and Papadopoulos [33] studied the discovery of k-cores in hidden networks. However, all these algorithms face two notable challenges when applied to HUB queries. First, they focus on bipartite networks and potentially take $O(|V|^2)$ probe operations when handling meta-path graphs. Second, the probe operations are time-consuming for our problem because they involve many breadth-first search (BFS) operations on the matching graph.

Xirogiannopoulos et al. [43] construct compact representations of hidden networks extracted by *path joins* in relational databases, which are analogous to meta-paths in KGs. However, this work is orthogonal to ours, while our sketch propagation framework is applicable to the compact structure of [43]. Besides, our exact

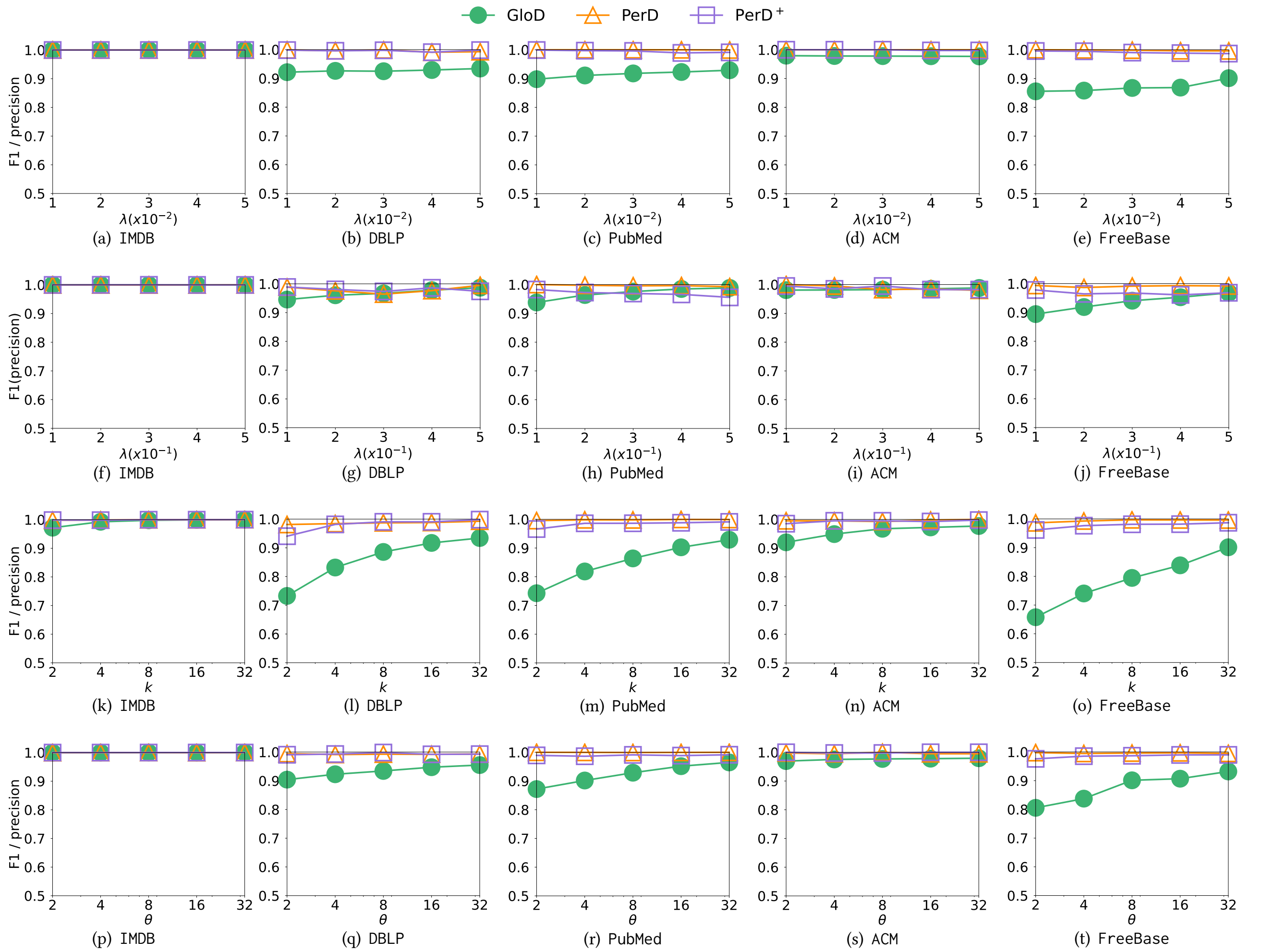


Figure 5: Effectiveness of GloD, PerD, and PerD$^+$ for HUB queries based on degree centrality with varying parameters. Subfigures (a)-(e): varying parameter λ. Subfigures (f)-(j): varying parameter k. Subfigures (k)-(o): varying parameter θ.

algorithms that leverage worst-case-optimal joins to compute neighbors efficiently are aligned with the future direction to improve the efficiency of neighbor computation suggested in [43].

Meta-paths in KGs. Meta-paths constitute a prominent approach for extracting information from knowledge graphs (KGs) and lend a fundamental concept to the design of node similarity measures. Measures such as Pathsim [34], Joinsim [42], RelSim [38], and Hetesim [29], leverage meta-paths to quantify node similarity and play a crucial role in various KG analysis tasks. For instance, Pathsim evaluates the similarity between nodes in a heterogeneous network as the number of matching meta-path instances connecting them. Joinsim is a variant of Pathsim that satisfies the triangle inequality, facilitating the efficient discovery of top-k similarity node pairs. Meta-paths also find applications in community search within KGs [12, 20, 44, 48]. Fang et al. [12] proposed the $(k, \mathcal{M})$-core model while Yang et al. [44] introduced the $(k, \mathcal{M})$-Btruss and $(k, \mathcal{M})$-Ctruss models for community search in KGs, which correspond to a k-core and a k-truss in the meta-path graph derived using $\mathcal{M}$, respectively. Additionally, meta-paths have been used in KG-based recommender systems [16, 31, 45, 46].

Nevertheless, even though meta-path graphs have been used in various scenarios, the fundamental problem of finding hub has received little attention. To the best of our knowledge, ours is the first systematic investigation of efficient hub queries over meta-path graphs derived from KGs.

7 CONCLUSION

We introduced and investigated the problem of finding hubs (HUB) over meta-path graphs derived from KGs, based on the degree or h-index scores. We proposed a sketch propagation framework for approximate degree-based HUB queries and a pivot-based algorithm that extends sketch propagation to h-index-based HUB queries. We studied personalized HUB queries based on both centrality measures and enhanced corresponding algorithms with early termination. Theoretically, sketch propagation and pivot-based algorithms correctly answer approximate HUB queries with high

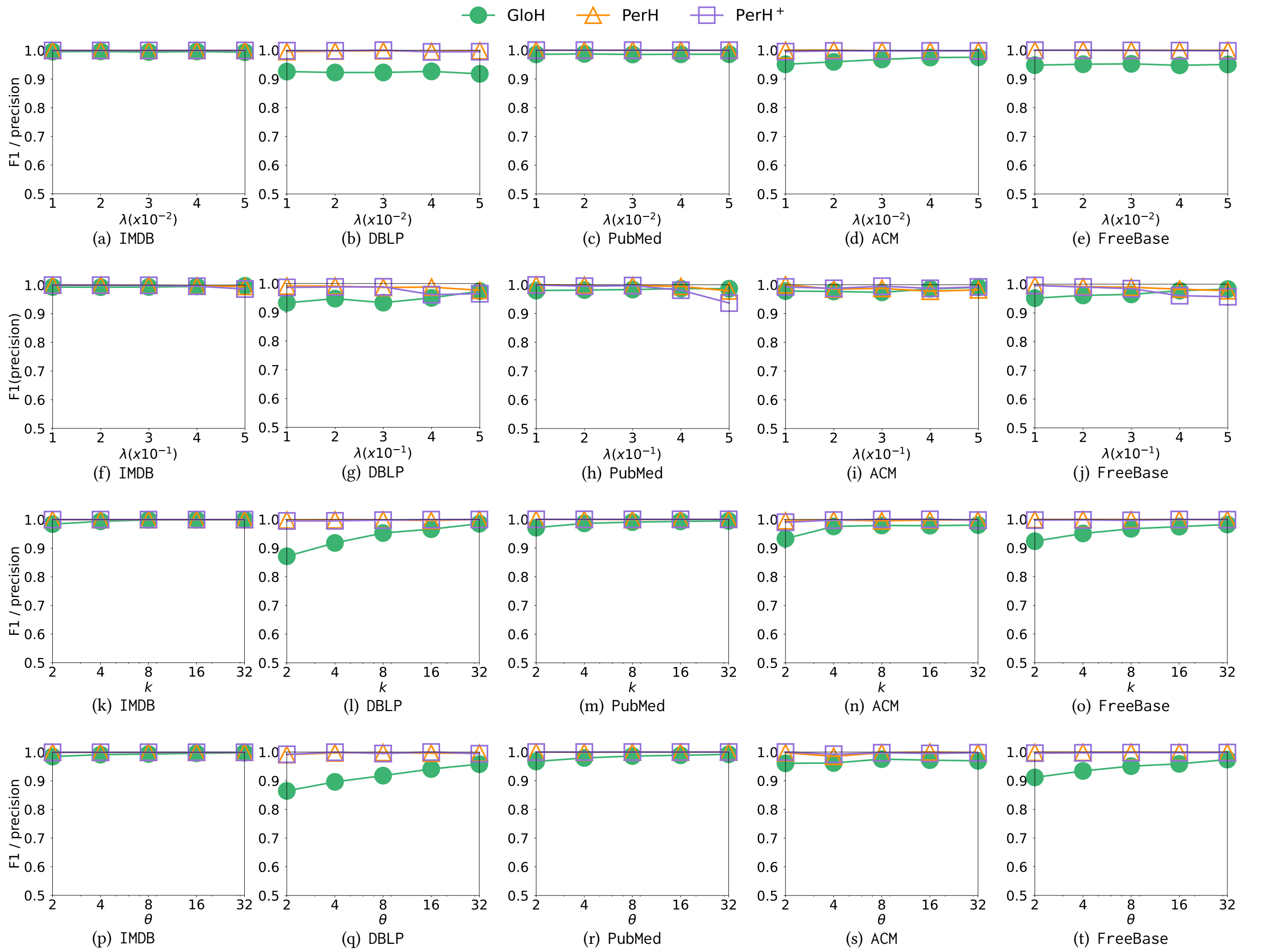


Figure 6: Effectiveness of GloH, PerH, and PerH$^+$ for HUB queries based on h-index with varying parameters. Subfigures (a)-(e): varying parameter λ. Subfigures (f)-(j): varying parameter k. Subfigures (k)-(o): varying parameter θ.

probability. Through extensive experimentation, we verified the effectiveness and efficiency of our proposals on real-world KGs.

REFERENCES

[1] Noga Alon, Yossi Matias, and Mario Szegedy. 1999. The Space Complexity of Approximating the Frequency Moments. *J. Comput. Syst. Sci.* 58, 1 (1999), 137–147.

[2] Diego Arroyuelo, Aidan Hogan, Gonzalo Navarro, Juan L. Reutter, Javiel Rojas-Ledesma, and Adrián Soto. 2021. Worst-Case Optimal Graph Joins in Almost No Space. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 102–114.

[3] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. 2007. Tuning Bandit Algorithms in Stochastic Environments. In *Algorithmic Learning Theory - 18th International Conference, ALT 2007, Sendai, Japan, October 1-4, 2007, Proceedings.* Springer, Berlin, Heidelberg, 150–165.

[4] Kevin S. Beyer, Peter J. Haas, Berthold Reinwald, Yannis Sismanis, and Rainer Gemulla. 2007. On Synopses for Distinct-Value Estimation under Multiset Operations. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD '07)*. Association for Computing Machinery, New York, NY, USA, 199–210.

[5] Bokai Cao, Mia Mao, Siim Viidu, and S Yu Philip. 2017. HitFraud: a broad learning approach for collective fraud detection in heterogeneous information networks. In *2017 IEEE international conference on data mining (ICDM)*. IEEE, 769–774.

[6] Wei Cao, Jian Li, Yufei Tao, and Zhize Li. 2015. On Top-k Selection in Multi-Armed Bandits and Hidden Bipartite Graphs. *Advances in Neural Information Processing Systems* 28 (2015), 1036–1044.

[7] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient Influence Maximization in Social Networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*. Association for Computing Machinery, New York, NY, USA, 199–208.

[8] Kousik Das, Sovan Samanta, and Madhumangal Pal. 2018. Study on centrality measures in social networks: a survey. *Soc. Netw. Anal. Min.* 8, 13 (2018), 1–11.

[9] Benjamin Doerr, Mahmoud Fouz, and Tobias Friedrich. 2012. Why rumors spread so quickly in social networks. *Commun. ACM* 55, 6 (2012), 70–75.

[10] Marianne Durand and Philippe Flajolet. 2003. Loglog Counting of Large Cardinalities. In *Algorithms - ESA 2003, 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003, Proceedings.* Springer, Berlin, Heidelberg, 605–617.

[11] Cristian Estan, George Varghese, and Michael E. Fisk. 2006. Bitmap algorithms for counting active flows on high-speed links. *IEEE/ACM Trans. Netw.* 14, 5 (2006), 925–937.

[12] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. 2020. Effective and Efficient Community Search over Large Heterogeneous Information Networks. *Proc. VLDB Endow.* 13, 6 (2020), 854–867.

[13] Michael J. Freitag, Maximilian Bandle, Tobias Schmidt, Alfons Kemper, and Thomas Neumann. 2020. Adopting Worst-Case Optimal Joins in Relational Database Systems. *Proc. VLDB Endow.* 13, 11 (2020), 1891–1904.

[14] Phillip B. Gibbons. 2001. Distinct Sampling for Highly-Accurate Answers to Distinct Values Queries and Event Reports. In *Proceedings of 27th International*

Conference on Very Large Data Bases (VLDB '01). Morgan Kaufmann, 541–550.
[15] Michael Greenwald and Sanjeev Khanna. 2001. Space-Efficient Online Computation of Quantile Summaries. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data (SIGMOD '01)*. Association for Computing Machinery, New York, NY, USA, 58–66.
[16] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2022. A Survey on Knowledge Graph-Based Recommender Systems. *IEEE Trans. Knowl. Data Eng.* 34, 8 (2022), 3549–3568.
[17] Stefan Heule, Marc Nunkesser, and Alexander Hall. 2013. HyperLogLog in Practice: Algorithmic Engineering of a State of the Art Cardinality Estimation Algorithm. In *Proceedings of the 16th International Conference on Extending Database Technology (EDBT '13)*. Association for Computing Machinery, New York, NY, USA, 683–692.
[18] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. 2021. Knowledge graphs. *ACM Computing Surveys (Csur)* 54, 4 (2021), 1–37.
[19] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems* 33, 2 (2021), 494–514.
[20] Yangqin Jiang, Yixiang Fang, Chenhao Ma, Xin Cao, and Chunshan Li. 2022. Effective Community Search over Large Star-Schema Heterogeneous Information Networks. *Proc. VLDB Endow.* 15, 11 (2022), 2307–2320.
[21] Yitong Li, Chuan Shi, Philip S Yu, and Qing Chen. 2014. Hrank: a path based ranking method in heterogeneous information network. In *Web-Age Information Management: 15th International Conference, WAIM 2014, Macau, China, June 16-18, 2014. Proceedings 15*. Springer, 553–565.
[22] Linyuan Lü, Tao Zhou, Qian-Ming Zhang, and H. Eugene Stanley. 2016. The H-index of a network node and its relation to degree and coreness. *Nat. Commun.* 7, 1 (2016), 10168.
[23] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. 2019. Anytime Bottom-Up Rule Learning for Knowledge Graph Completion. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 3137–3143.
[24] Shodai Mihara, Sho Tsugawa, and Hiroyuki Ohsaki. 2015. Influence Maximization Problem for Unknown Social Networks. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015 (ASONAM '15)*. Association for Computing Machinery, New York, NY, USA, 1539–1546.
[25] Mark E. J. Newman. 2002. Assortative mixing in networks. *Phys. Rev. Lett.* 89, 20 (2002), 208701.
[26] Hung Q. Ngo, Ely Porat, Christopher Ré, and Atri Rudra. 2018. Worst-case Optimal Join Algorithms. *J. ACM* 65, 3, Article 16 (2018), 40 pages.
[27] Gerald Paul, Sameet Sreenivasan, Shlomo Havlin, and H. Eugene Stanley. 2006. Optimization of network robustness to random breakdowns. *Phys. A: Stat. Mech. Appl.* 370, 2 (2006), 854–862.
[28] Cheng Sheng, Yufei Tao, and Jianzhong Li. 2012. Exact and approximate algorithms for the most connected vertex problem. *ACM Trans. Database Syst.* 37, 2, Article 12 (2012), 39 pages.
[29] Chuan Shi, Xiangnan Kong, Yue Huang, Philip S. Yu, and Bin Wu. 2014. HeteSim: A General Framework for Relevance Measure in Heterogeneous Networks. *IEEE Trans. Knowl. Data Eng.* 26, 10 (2014), 2479–2492.
[30] Chuan Shi, Yitong Li, Philip S Yu, and Bin Wu. 2016. Constrained-meta-path-based ranking in heterogeneous information network. *Knowledge and Information Systems* 49 (2016), 719–747.
[31] Chuan Shi, Zhiqiang Zhang, Ping Luo, Philip S. Yu, Yading Yue, and Bin Wu. 2015. Semantic Path based Personalized Recommendation on Weighted Heterogeneous Information Networks. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM '15)*. Association for Computing Machinery, New York, NY, USA, 453–462.
[32] Sriganesh Srihari and Hon Wai Leong. 2013. A survey of computational methods for protein complex prediction from protein interaction networks. *Journal of bioinformatics and computational biology* 11, 02 (2013), 1230002.
[33] Panagiotis Strouthopoulos and Apostolos N. Papadopoulos. 2019. Core discovery in hidden networks. *Data Knowl. Eng.* 120 (2019), 45–59.
[34] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *Proc. VLDB Endow.* 4, 11 (2011), 992–1003.
[35] Toshihiro Tanizawa, Shlomo Havlin, and H. Eugene Stanley. 2012. Robustness of onionlike correlated networks against targeted attacks. *Phys. Rev. E* 85, 4 (2012), 046109.
[36] Yufei Tao, Cheng Sheng, and Jianzhong Li. 2010. Finding Maximum Degrees in Hidden Bipartite Graphs. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD '10)*. Association for Computing Machinery, New York, NY, USA, 891–902.
[37] Todd L. Veldhuizen. 2014. Triejoin: A Simple, Worst-Case Optimal Join Algorithm. In *Proceedings of the 17th International Conference on Database Theory (ICDT)*. OpenProceedings.org, 96–106.
[38] Chenguang Wang, Yizhou Sun, Yanglei Song, Jiawei Han, Yangqiu Song, Lidan Wang, and Ming Zhang. 2016. RelSim: Relation Similarity Search in Schema-Rich Heterogeneous Information Networks. In *Proceedings of the 2016 SIAM International Conference on Data Mining (SDM)*. SIAM, 621–629.
[39] Jianguo Wang, Eric Lo, and Man Lung Yiu. 2013. Identifying the Most Connected Vertices in Hidden Bipartite Graphs Using Group Testing. *IEEE Trans. Knowl. Data Eng.* 25, 10 (2013), 2245–2256.
[40] Min Wu, Xiaoli Li, Chee-Keong Kwoh, and See-Kiong Ng. 2009. A core-attachment based method to detect protein complexes in PPI networks. *BMC bioinformatics* 10 (2009), 1–16.
[41] Zhi-Xi Wu and Petter Holme. 2011. Onion structure and network robustness. *Phys. Rev. E* 84, 2 (2011), 026106.
[42] Yun Xiong, Yangyong Zhu, and Philip S. Yu. 2015. Top-k Similarity Join in Heterogeneous Information Networks. *IEEE Trans. Knowl. Data Eng.* 27, 6 (2015), 1710–1723.
[43] Konstantinos Xirogiannopoulos and Amol Deshpande. 2017. Extracting and Analyzing Hidden Graphs from Relational Databases. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17)*. Association for Computing Machinery, New York, NY, USA, 897–912.
[44] Yixing Yang, Yixiang Fang, Xuemin Lin, and Wenjie Zhang. 2020. Effective and Efficient Truss Computation over Large Heterogeneous Information Networks. In *36th IEEE International Conference on Data Engineering (ICDE)*. IEEE, 901–912.
[45] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized Entity Recommendation: A Heterogeneous Information Network Approach. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM '14)*. Association for Computing Machinery, New York, NY, USA, 283–292.
[46] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*. Association for Computing Machinery, New York, NY, USA, 635–644.
[47] Yuyan Zheng, Chuan Shi, Xiaohuan Cao, Xiaoli Li, and Bin Wu. 2017. Entity Set Expansion with Meta Path in Knowledge Graph. In *Advances in Knowledge Discovery and Data Mining - 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part I*. Springer, Cham, 317–329.
[48] Yingli Zhou, Yixiang Fang, Wensheng Luo, and Yunming Ye. 2023. Influential Community Search over Large Heterogeneous Information Networks. *Proc. VLDB Endow.* 16, 8 (2023), 2047–2060.