A Sketch Propagation Framework for Hub Queries on Unmaterialized Relational Graphs

Yudong Niu[1], Yuchen Li[1], Panagiotis Karras[2], Yanhao Wang[3]
[1]*School of Computing and Information Systems, Singapore Management University, Singapore*
[2]*Department of Computer Science, University of Copenhagen, Copenhagen, Denmark*
[3]*School of Data Science and Engineering, East China Normal University, Shanghai, China*
ydniu.2018@phdcs.smu.edu.sg, yuchenli@smu.edu.sg, piekarras@gmail.com, yhwang@dase.ecnu.edu.cn

***Abstract*—Relational graphs encapsulate nontrivial inherent interactions among entities in heterogeneous data sources. Identifying *hubs* in relational graphs is vital in various applications such as fraud detection, influence analysis, and protein complex discovery. However, building relational graphs induced by meta-paths on heterogeneous data entails substantial costs, thus hindering efficient hub discovery. In this paper, we propose a novel sketch propagation framework for approximate hub queries in induced relational graphs that avoids explicitly materializing those graphs. Our framework specifically supports hub queries that ask for all nodes whose centrality scores, based on *degree* or *h-index*, are in the top quantile with provable guarantees under the notion of ϵ-separable sets. In addition, we devise pruning techniques that efficiently process *personalized* hub queries asking whether a given node is a hub. Extensive experiments on real-world and synthetic data confirm the efficacy and efficiency of our proposals, which achieve orders of magnitude speedups over exact methods while consistently attaining accuracy beyond 90%.**

I. Introduction

The graph model has been widely used to denote relationships between different entities and to support data mining tasks across domains [1]–[4]. Application needs often necessitate *inducing* composite relationships from heterogeneous data using meta-paths [5]–[7] and representing them via *relational graphs*. For example, on e-commerce platforms, market segmentation relies on community analysis in such *induced* relational graphs that link customers with shared shopping preferences via meta-paths such as "(*customer*) $\rightarrow$ (*product*) $\leftarrow$ (*customer*)". Nevertheless, building such induced graphs is resource-intensive due to the cost of "join" operations on heterogeneous data [8]. Moreover, although several studies have focused on accelerating relational graph materialization [5], [9]–[11], these methods are unsuitable for online business applications that require real-time on-demand analysis over time spans, rendering full materialization infeasible.

To address these challenges, we propose the analysis of induced relational graphs without full materialization. As a pioneer effort in this direction, we focus on *hub queries* (HUB), which find important nodes in a relational graph according to a given *centrality measure*.

Applications. HUB queries in relational graphs find applications in a variety of real-world scenarios, including:

- *Fraud Detection:* In e-commence platforms, by identifying hubs in relational graphs induced by "(*account*) $\rightarrow$ (*transaction*) $\leftarrow$ (*account*)" from transactional data, we can uncover financial accounts with an abnormally large number of interactions with other accounts, potentially indicating the existence of money laundering [12].
- *Influence Analysis:* Hubs in a co-author network induced by "(*author*) $\rightarrow$ (*paper*) $\leftarrow$ (*author*)" [13] denote influential researchers who are active in general domains. Moreover, hubs in a venue-filtered co-author network induced by meta-paths, e.g., "(*author*) $\rightarrow$ (*paper*, *venue*='DB') $\leftarrow$ (*author*)", reveal influencers in specific research areas [11], [14].
- *Protein Complex Discovery:* Protein-protein interaction (PPI) networks induced by meta-paths such as "(*protein*) $\rightarrow$ (*pathway*) $\leftarrow$ (*protein*)" from protein databases such as STRING (see https://string-db.org/) and Reactome (see https://reactome.org/) capture collective interactions among proteins in physiological pathways. Hubs in a PPI network are vital for the discovery of protein complexes [15], [16].

Our Contributions. We first consider HUB queries based on degree centrality. Inspired by cardinality sketches [17]–[21], we propose a novel sketch propagation framework for HUB queries on relational graphs with provable approximation guarantees. Specifically, we construct a *neighborhood cardinality* sketch for each node to estimate its degree in the relational graph by iteratively propagating sketches along edges in matching instances of the given meta-path $\mathcal{M}$. Therefore, instead of computing the degree of each node after materialization, we collectively estimate the degrees of all nodes through a *single* propagation process, which substantially improves the efficiency of processing HUB queries under degree centrality.

Then, we extend the sketch framework to approximate HUB queries based on the h-index [22] which requires higher-order neighborhood information. Specifically, a key challenge is that the h-index of a node depends, in addition to its own degree, on the degrees of its neighbors, while cardinality sketches can only provide information on a node's own degree. To address this issue, we propose a *pivot-based* algorithm built on the sketch propagation process. Instead of directly estimating h-indexes, we recursively find the nodes whose h-indexes reach or exceed that of a randomly chosen pivot node using sketches, in a quicksort-like fashion. As such, we can skip the nodes that cannot be hubs so as to process h-index-based HUB queries more efficiently. Note that we consider the h-index as a representative of centrality measures based on higher-order neighborhoods. By employing similar techniques, our framework can also be adapted to support other higher-order centrality measures, such as the i10-index and g-index [23]. Furthermore, our framework can be extended to support closeness [24], [25] centrality by propagating cardinality sketches over the matching instances of $\mathcal{M}$ in multiple iterations to estimate the number of nodes at different distances from each node in the relational graph.

Furthermore, we devise efficient pruning methods to answer *personalized* HUB queries, that is, to determine whether a given node is a hub. We maintain a lower bound on the number of nodes with centrality scores higher than the query node and terminate the sketch propagation process when that lower bound exceeds a given threshold. This early termination mechanism can accelerate personalized HUB queries based on both degree and h-index centrality measures.

Under the notion of ϵ-separable sets, we show that the sketch propagation framework provides *unbiased and efficient* approximations for HUB queries based on both centrality measures. The experiments reveal that the estimations converge much more quickly than the worst-case bound on the number of propagation iterations, achieving orders of magnitude speedups over exact methods.

Our main contributions are summarized as follows.

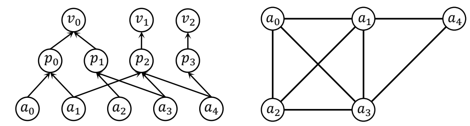
- We introduce the novel problem of hub queries (HUB) based on the degree and h-index centrality measures over relational graphs induced by meta-paths (Sect. II).
- We propose a *sketch propagation* framework for approximate degree-based HUB queries and extend it to personalized HUB queries with early termination (Sect. III).
- We devise a *pivot-based* algorithm, also with early termination, for approximate (personalized) HUB queries using the h-index measure (Sect. IV).
- We show that our methods scale well to massive graphs in which state-of-the-art exact methods fail to terminate in a reasonable time and mostly achieve orders of magnitude speedups over them while yielding accuracy beyond 90%.
- We conduct case studies to indicate that our methods serve as seeding strategies comparable to state-of-the-art methods for influence analysis on relational graphs (Sect. V).

II. Preliminaries

A. Notation and Problem Formulation

We model a heterogeneous data source as a knowledge graph (KG). However, our formulation and methodology are independent of the data format and can be easily adapted to other data sources such as RDBMS.

We denote a KG as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L})$ with $\mathcal{V}$ and $\mathcal{E}$ representing the sets of node and edge instances. An edge instance $e = (u, v) \in \mathcal{E}$ connects two nodes $u, v \in \mathcal{V}$. The type function $\mathcal{L}$ maps each node or edge instance, v or e, to its type, $\mathcal{L}(v)$ or $\mathcal{L}(e)$. For simplicity of presentation, we assume that $\mathcal{L}(e)$ is determined by the type of its end nodes. Nevertheless, our methods can also handle more complex scenarios.



(a) KG $\mathcal{G}$ (b) Relational graph H

Fig. 1. An exemplary KG $\mathcal{G}$ and a relational graph H induced from $\mathcal{G}$ using a meta-path $\mathcal{M}(A, P, V, P, A)$.

Meta-paths are commonly used to induce relational graphs from heterogeneous data [6], [26]. We provide the definitions of *meta-path* and its *matching instances* on the KG as follows:

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

Although our methods support any meta-path $\mathcal{M}$, following the established convention [6], [26], [27], we only consider symmetric meta-paths to induce homogeneous relational graphs. A meta-path $\mathcal{M}$ is symmetric if $\mathcal{M} = \mathcal{M}^{-1}$.

Definition 2 (Matching Instance)**.** *A matching instance M to a meta-path $\mathcal{M}$ is a sequence of node instances in $\mathcal{G}$ denoted by $M = (v_0, v_1, \ldots, v_{L-1}, v_L) \triangleright \mathcal{M}$ satisfying:*

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

We use $M(x_i)$ to denote the node v_i in M.

Definition 3 (Relational Graph)**.** *For a KG $\mathcal{G}$, a (symmetric) meta-path $\mathcal{M}$ induces a relational graph $H_{\mathcal{M}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$, where $V_{\mathcal{M}}$ contains all nodes in $\mathcal{V}$ that match x_0 in any instance of $\mathcal{M}$ and $E_{\mathcal{M}}$ consists of all edges (u, v) for any two nodes $u, v \in V_{\mathcal{M}}$ such that there is an instance $M \triangleright \mathcal{M}$ in $\mathcal{G}$ with $M(x_0) = u$ and $M(x_L) = v$.*

We denote the neighborhood of u in $H_{\mathcal{M}}$ as $\mathcal{N}(u)$ and *degree* of u in $H_{\mathcal{M}}$ as $N(u) = |\mathcal{N}(u)|$. Furthermore, $\Lambda = \max_{u \in H} N(u)$ represents the maximum degree among all nodes in $H_{\mathcal{M}}$. When the context is clear, we will drop $\mathcal{M}$ and use $H = (V, E)$ to represent a relational graph.

Example 1. *Fig. 1(a) presents an academic KG with three node types A ('author'), P ('paper'), and V ('venue') and two edge types $A \to P$ ('write') and $P \to V$ ('publish'). A sequence (A, P, V, P, A) denotes a meta-path $\mathcal{M}$, which induces the relational graph H in Fig. 1(b), where a_0 and a_2 are neighbors because there exist an instance $M = (a_0, p_0, v_0, p_1, a_2)$ of $\mathcal{M}$ in $\mathcal{G}$. Intuitively, the two authors a_0 and a_2 are connected since they publish papers in the same venue v_0.*

<u>Problem Statement.</u> We study the problem of hub queries in relational graphs. Hubs are nodes ranking high according to a function $c : V \to \mathbb{R}_{\geq 0}$ that measures centrality. We use *degree* [28], 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 centrality measures. The HUB query we consider is formalized as follows:

Problem 1 (HUB Query)**.** *Given a relational graph H and a parameter $\lambda \in (0,1)$, 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* HUB queries to ask whether a given node q is a hub of H.

Problem 2 (Personalized HUB Query)**.** *Given a relational graph H, a node q, and $\lambda \in (0,1)$, decide if $c(q) \geq c(v^\lambda)$.*

Processing HUB queries on large-scale relational graphs is computationally expensive. Specifically, computing the exact centrality such as the degree of each node requires traversing the matching graph with $|\mathcal{E}^*|$ edges (as defined in Sect. II-C) from the corresponding node, leading to a time complexity of $O(|V| \cdot |\mathcal{E}^*|)$ in total, which is prohibitive for large KGs. Thus, we consider how to answer HUB queries approximately and efficiently using randomized algorithms under the notion of ϵ-separable sets.

Definition 4 (ϵ-Separable Set [29])**.** *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 scores 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 $S^-_\epsilon(u) = \{v \in V | c(v) < (1-\epsilon) \cdot c(u)\}$.*

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

Problem 3 (ϵ-Approximate HUB Query)**.** *Given a relational 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 relational graph H, a node q, and $\epsilon, \lambda \in (0,1)$, return* `TRUE` *if $q \in S^+_\epsilon(v^\lambda)$ and* `FALSE` *if $q \in S^-_\epsilon(v^\lambda)$.*

Before presenting our proposed algorithms for HUB queries, we summarize the frequently used symbols in Table I.

TABLE I
FREQUENTLY USED SYMBOLS.

Symbol	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L})$	A knowledge graph
$\mathcal{M}, \mathcal{M}^{-1}$	A meta-path and its inverse path
$M \triangleright \mathcal{M}$	An instance of the meta-path $\mathcal{M}$
$\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$	The matching graph
$H = (V, E)$	The relational graph induced by a meta-path
$c(u), \tilde{c}(u)$	The centrality score of u and its estimation
$\mathcal{N}(v)$	The set of neighbors of v in H
Λ	The maximum degree among all nodes in H
$\mathcal{K}(C)$	The KMV sketch of a collection C
$\mathcal{I}(u)$, $I(u)$	The image of u and its size
v^λ	The node ranked at the $(1-\lambda)$-percentile
S	The set of result nodes for a HUB query
$S^+_\epsilon(u), S^-_\epsilon(u)$	The set of nodes $\{v \in V \| c(v) > (1+\epsilon)c(u)\}$ and $\{v \in V \| c(v) < (1-\epsilon)c(u)\}$, respectively

B. Related Work

Relational Graph Analysis. There have been extensive studies on materializing and analyzing relational graphs induced by meta-paths from heterogeneous data. Chatzopoulos et al. [9] accelerated the enumeration of instances for multiple meta-paths by sharing workloads. This method can be extended to relational graph materialization. Guo et al. [10] first proposed BoolAP, an algorithm for relational graph construction through boolean matrix multiplication. They also proposed BoolAP$^+$, an optimized algorithm for graphs with locally dense regions. Although these methods improve the efficiency of relational graph materialization over naïve methods, they are still prohibitively expensive for large relational graphs, as indicated in our experiments (see Sect. V-B).

Then, great effort [6], [26], [27], [30] has been devoted to community search over relational graphs. Fang et al. [6] and Yang et al. [26] proposed $(k, \mathcal{M})$-core and $(k, \mathcal{M})$-truss community models in relational graphs induced by meta-paths. Jiang et al. [27] and Zhou et al. [30] extended the $(k, \mathcal{M})$-core model by incorporating star schemas and influences. We note that these methods introduced edge- and vertex-disjoint meta-paths to induce relational graphs. Despite their effectiveness for community definition, they take $O(|V|^2 \cdot |\mathcal{E}^*|)$ time to extract a relational graph since they incur expensive max-flow computations to find the neighbors of each node on meta-path networks. Since the HUB problem is already computationally challenging for non-disjoint meta-paths, we leave the study of disjoint meta-paths for future work.

Xirogiannopoulos et al. [8] proposed to construct compact representations of relational graphs induced by *path joins* in databases for memory-efficient query processing. Although they focus on a different problem from ours, the sketch we build can serve as the compact structure in [8]. In addition, the exact algorithm in Sect. II-C that leverages the worst-case optimal join for neighborhood computation aligns with the future direction for efficiency improvements suggested in [8].

Meta-paths were also used to rank entities based on centrality measures [31]–[33] and assess the similarity between entities [5], [34]–[36] within heterogeneous data. These methods can operate directly on the original data and do not involve the materialization of relational graphs.

Generally, although relational graphs were widely used in different applications, the fundamental problem of hub queries has received little attention. To the best of our knowledge, this is the first systematic investigation of efficient hub queries over relational graphs induced from heterogeneous data.

Sketch-based Graph Queries. Several works have been devoted to accelerating graph queries using cardinality sketches. Cohen [37] introduced the use of cardinality sketches to estimate the size of transitive closure of each node, which was subsequently extended to influence queries [38] by estimating the combined reachability size of a query node over sampled graphs under the independent cascade (IC) model. Liu et al. [39] extended the structural graph clustering to consider the multi-hop neighborhood for similarity computation, employing KMV sketches to accelerate the estimation of multi-hop Jaccard similarities. Furthermore, Zhang et al. [40] proposed an efficient approach for maintaining neighborhood

KMV sketches following graph updates, thereby extending the sketch-based structural graph clustering to dynamic graphs. Chierichetti et al. [41] utilized cardinality sketches for efficient closeness centrality and reachability tree size estimation within public-private networks from the point of view of each node.

Nevertheless, existing studies that apply cardinality sketches to graph problems primarily focus on accelerating graph queries over materialized graphs by eliminating computationally expensive intermediate results. In contrast, we adopt cardinality sketches to avoid the materialization of relational graphs when performing graph queries.

Hubs in Hidden Networks. Another line of literature has studied the problem of finding hubs in hidden networks, where the existence of any edge can only be decided via *probe operations*. Tao et al. [42] proposed two instance-optimal exact algorithms over two kinds of probing strategies of hidden networks. Wang et al. [43] extended this problem to include group testing, whereby probes can verify edge connections between multiple nodes. Sheng et al. [44] proposed a sampling algorithm for hub queries, upon which Cao et al. [45] subsequently improved. Strouthopoulos et al. [46] studied the discovery of k-cores in hidden networks. However, all of the above algorithms face two challenges when applied to HUB queries. First, they focus on bipartite networks and potentially take $O(|V|^2)$ probe operations for HUB queries on relational graphs. Second, the probe operations are time-consuming for HUB queries because they involve many breadth-first search (BFS) operations on the matching graph.

C. Memory-Efficient Exact Method for HUB

Since the materialization of large relational graphs can often be memory-prohibitive [8], we propose a memory-efficient baseline method for exact HUB queries, which computes the exact centrality of each node in H using a generic worst-case optimal join algorithm [47]–[50] on the *matching graph* of $\mathcal{M}$ over $\mathcal{G}$ and avoid full materialization. We first introduce the notion of *matching graph* of a meta-path.

Definition 5 (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 node set $\mathcal{V}^* = \bigcup_{0\le i\le L} \mathcal{V}_i$ and edge set $\mathcal{E}^* = \bigcup_{0\le i<L} \mathcal{E}_i$, 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 that connects two nodes between $\mathcal{V}_i$ and $\mathcal{V}_{i+1}$. For each node $u \in \mathcal{V}_i$, we use $\mathcal{V}^+(u)$ to denote the nodes in $\mathcal{V}_{i+1}$ connected with u, i.e.,* successors *of u in $\mathcal{G}^*$; and use $\mathcal{V}^-(u)$ to denote the nodes in $\mathcal{V}_{i-1}$ connected with u, i.e., predecessors of u in $\mathcal{G}^*$.*

For each node $u \in \mathcal{V}_i$, we use $\bar{u}$ to denote the node in $\mathcal{V}_{L-i}$ corresponding to the same node in $\mathcal{G}$.

Example 2. *Fig. 2(a) illustrates the matching graph $\mathcal{G}^*$ of a meta-path $\mathcal{M}(A, P, V, P, A)$. The node set $\mathcal{V}_0 = \{a_0, a_1, \dots, a_4\}$ consists of nodes of type x_0, i.e., 'authors'. Then, the set of successors of v_0 is $\mathcal{V}^+(v_0) = \{\bar{p}_0, \bar{p}_1\}$, which contains the neighbors of v_0 in $\mathcal{V}_3$, and the set of predecessors of v_0 is $\mathcal{V}^-(v_0) = \{p_0, p_1\}$, which contains the neighbors of v_0 in $\mathcal{V}_1$.*

Procedure DFS(u, v)

Input: The matching graph $\mathcal{G}^*$

1 Mark v as visited;
2 **if** $v \in \mathcal{V}_L$ **then** Add v to $\mathcal{N}(u)$;
3 **for** $w \in \mathcal{V}^+(v)$ **do**
4   **if** w is not visited **then** DFS(u, w);

The matching graph $\mathcal{G}^*$ can be extracted from $\mathcal{G}$ level by level efficiently through the breadth-first search (BFS). Given the matching nodes in $\mathcal{V}_i$, we iteratively visits all neighbors in $\mathcal{G}$ of each node in $\mathcal{V}_i$ and add its neighbors with type x_{i+1} as successors (in $\mathcal{V}_{i+1}$) of the matching node in the matching graph. Based on our experimental findings (see Table IV), the cost of extracting $\mathcal{G}^*$ from $\mathcal{G}$ is negligible compared to the total computational cost of HUB queries.

To avoid generating excessive intermediate results, we employ a worst-case optimal join on the matching graph to compute the neighbors of each node $u \in V$ in the relational graph. Starting with a depth-first search (DFS) from each node u, we traverse the matching instances of $\mathcal{M}$. Specifically, DFS(u, u) (Procedure DFS) runs for each node $u \in V$ and recursively traverses the current node's successors (Lines 3–4). Once we visit a node v in $\mathcal{V}_L$, we add it to $\mathcal{N}(u)$ (Line 2). After performing DFS for each node, we discard $\mathcal{N}(u)$ and keep only the degree $|\mathcal{N}(u)|$ to ensure a small memory footprint. To compute the h-indexes of all nodes without *materializing* the relational graph, we perform the worst-case optimal join in two rounds. The first round computes and stores the degree of each node $u \in V$; the second round computes the h-index of each node u by combining its neighborhood $\mathcal{N}(u)$ with the degrees kept in the first round.

Although the memory consumption is bounded by $O(|\mathcal{V}^*| + |\mathcal{E}^*|)$ and thus is not a bottleneck, the exact algorithm is still inefficient due to repeated edge traversals. 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 processing large relational graphs.

III. 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. In addition, we propose an early termination technique to further accelerate personalized HUB query processing.

A. Sketch Propagation for 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 [51] and defined as follows.

Definition 6 (KMV Sketch)**.** *Given a collection $C = \{o_0, o_1, \ldots, o_{|C|}\}$ 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 as $|\widetilde{C}| = \frac{k}{\mathbb{E}[\zeta]} - 1$, where ζ is a random variable by taking the largest number in $\mathcal{K}(C)$.*

By constructing multiple (specifically θ) KMV sketches and applying the Bernstein inequality [52], an estimation of the collection size with high probability can be obtained.

Lemma 1. *For any $\delta, p \in (0, 1)$, if $\theta = \Theta(\frac{|C|}{\delta k} \log \frac{1}{p})$, we have $(1-\delta)|C| \le |\widetilde{C}| \le (1+\delta)|C|$ with probability at least $1-p$.*

Note that the KMV sketch is designed primarily for stream processing [17]–[21], where a one-pass scan over the collection C is required. However, for HUB queries, scanning the neighborhood of each node is quite expensive, as discussed in the description of exact algorithms. To address this challenge, we propose the *sketch propagation* framework that constructs a KMV sketch for $\mathcal{N}(v)$ of each $v \in V$ without explicitly generating and scanning $\mathcal{N}(v)$ and offers a different estimator with a tight error bound for the case where $\mathcal{N}(v)$ is relatively small based on the notion of *images*.

Definition 7 (Images)**.** *The image of a matching node $u \in \mathcal{V}_i$, denoted as $\mathcal{I}(u)$, contains a node $v \in \mathcal{V}_0$ if there exists an instance M of $\mathcal{M}$ including both u and v, i.e.,*

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

Example 3. *In Fig. 2(a), the image $\mathcal{I}(p_1)$ contains two nodes a_2 and a_3, as they are connected with p_1 via $M(a_2, p_1, v_0, \bar{p}_0, \bar{a}_0)$ and $M(a_3, p_1, v_0, \bar{p}_0, \bar{a}_0)$, respectively.*

An important observation is that $\mathcal{N}(u) = \mathcal{I}(\bar{u})$ for each node $u \in V = \mathcal{V}_0$ in the relational graph due to its definition.

Sketch Propagation. We build a KMV sketch for $\mathcal{N}(u)$ of each $u \in V$ by iteratively maintaining the sketches for the 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 $\mathcal{I}(u) = \{u\}$, the sketch is initialized as $\mathcal{K}(\mathcal{I}(u)) = \{r(u)\}$. The sketches for the images of the nodes in $\mathcal{V}_i$ are then constructed by iteratively propagating the sketches of the nodes in $\mathcal{V}_{i-1}$. Specifically, each node in $\mathcal{V}_{i-1}$ propagates its sketch to all its successor nodes in $\mathcal{V}_i$ and a node in $\mathcal{V}_i$ builds its sketch by retaining the k smallest random numbers among all sketches it receives. The following lemma ensures the correctness of the *sketch propagation* framework.

Lemma 2. *Given a meta-path $\mathcal{M}$, for each node $u \in \mathcal{V}_i$*

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

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

Proof. According to [51, Thm. 5], $\mathcal{K}(A) \oplus \mathcal{K}(B)$ is a KMV sketch for the collection $A \cup B$ for two collections A and B. Therefore, we only need to prove $\mathcal{I}(u) = \bigcup_{v \in \mathcal{V}^-(u)} \mathcal{I}(v)$ for each $u \in \mathcal{V}_i$.

On the one hand, for any node $u' \in \mathcal{I}(u)$, 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 $u' \in \mathcal{I}(v)$ and $v \in \mathcal{V}^-(u)$. Thus, $u' \in \bigcup_{v \in \mathcal{V}^-(u)} \mathcal{I}(v)$, i.e., $\mathcal{I}(u) \subseteq \bigcup_{v \in \mathcal{V}^-(u)} \mathcal{I}(v)$.

On the other hand, for any node $u' \in \bigcup_{v \in \mathcal{V}^-(u)} \mathcal{I}(v)$, there exists a node $v \in \mathcal{V}^-(u)$ such that $u' \in \mathcal{I}(v)$, 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, $u' \in \mathcal{I}(u)$, i.e., $\bigcup_{v \in \mathcal{V}^-(u)} \mathcal{I}(v) \subseteq \mathcal{I}(u)$.

Thus, we have $\mathcal{I}(u) = \bigcup_{v \in \mathcal{V}^-(u)} \mathcal{I}(v)$ for each $u \in \mathcal{V}_i$. $\square$

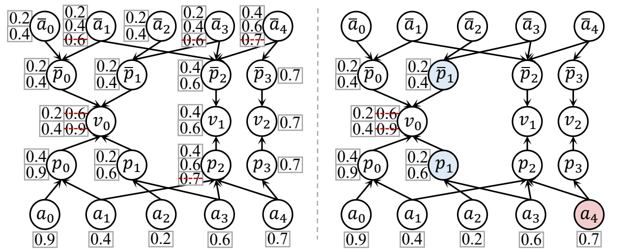
Algorithm 1: Sketch Propagation

Input: KG $\mathcal{G}$, meta-path $\mathcal{M}$, quantile parameter λ, number of propagations θ, KMV sketch size k
Output: A set of hubs S

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

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

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



(a) Sketch propagation (b) Early termination

Fig. 2. Running example of sketch propagation framework with $k = 2$ and $\theta = 1$. Subfigure (a) illustrates the complete sketch propagation process for a HUB query; Subfigure (b) demonstrates the early terminated sketch propagation for a personalized HUB query with a_4 as the query node. The redlined elements are discarded due to the k-size limitation.

Algorithm 1 details the *sketch propagation* framework. It receives a KG $\mathcal{G}$, a meta-path $\mathcal{M}$, a parameter $\lambda \in (0, 1)$, 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 an array $\mathcal{K}$ to store θ KMV sketches for images of each node $u \in \mathcal{V}^*$ (Lines 1–4). Next, it calls the function `Propagate`, which constructs KMV sketches within $\mathcal{K}$ (Lines 9–10) and estimates the degree of each node u in H using $\mathcal{K}$ (Lines 11–18). 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, it scans the random numbers in the sketches of all nodes in $\mathcal{V}^-(u)$ and keeps the k smallest numbers in $\mathcal{K}[u][t]$ with a min-heap of size k (Lines 21 and 22). Finally, it returns top-$(\lambda \cdot |V|)$ nodes with the highest estimated degrees as S (Lines 6 & 7). The ties are broken arbitrarily.

Example 4. *Fig. 2(a) illustrates the sketch propagation process with $k = 2$ and $\theta = 1$. To begin with, we generate a random number for each node in $\mathcal{V}_0$ and then propagate random numbers along the matching graph to $\mathcal{V}_4$. The sketch $\mathcal{K}(\mathcal{I}(v_0))$ is constructed by taking the two smallest numbers from the union of sketches of the images of the nodes in $\mathcal{V}^-(v_0)$. Thus, $\mathcal{K}(\mathcal{I}(v_0)) = \{0.4, 0.9\} \oplus \{0.2, 0.6\} = \{0.2, 0.4\}$ and the cardinality of $\mathcal{I}(v_0)$ is estimated as $\widetilde{I}(v_0) = \frac{2}{0.4} - 1 = 4$.*

Theoretical Analysis. Next, we will prove that *sketch propagation* approximates HUB queries with high probability (w.h.p.) (cf. Theorem 1). Hereafter, we use $I(u)$ to denote the sizes of $\mathcal{I}(u)$ for any node $u \in \mathcal{V}^*$. We also use $\widetilde{I}(u)$ to denote the estimate of $I(u)$ provided by sketch propagation.

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

Proof. Let $\widetilde{N}(u)$ denote the estimated degree of node u via sketch propagation. Given any $u \in S_0^+(v^\lambda)$ 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 by applying Lemma 1 over $\mathcal{K}(\mathcal{I}(v))$ 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 $\theta = \Theta(\frac{\Lambda}{\epsilon k} \log \frac{|V|}{p})$, 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 $\theta = \Theta(\frac{\Lambda}{\epsilon k} \log \frac{|V|}{p})$, 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 $\theta = \Theta(\frac{\Lambda}{\epsilon k} \log \frac{|V|^2}{p}) = \Theta(\frac{\Lambda}{\epsilon k} \log \frac{|V|}{p})$, 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 $1 - p$. $\square$

The time complexity of Algorithm 1 is $O(\frac{\Lambda|\mathcal{E}^*|}{\epsilon} \log \frac{|V|}{p})$. Its bottleneck lies in the KMV sketch construction process, which propagates θ sketches of sizes at most k over the matching graph with $|\mathcal{E}^*|$ edges. Its space complexity is $O(k\theta|\mathcal{V}^*| + |\mathcal{E}^*|)$. Compared to the exact algorithm, the additional space is used to store KMV sketches.

B. Early Termination for 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 1 for personalized queries.

Corollary 1. *When $\theta = \Theta(\frac{\Lambda}{\epsilon k} \log \frac{|V|}{p})$, the sketch propagation correctly answers any personalized ϵ-approximate HUB query under degree centrality with probability at least $1 - p$.*

Nevertheless, we can optimize personalized HUB query processing by terminating the *sketch propagation* process once we can determine that q is not a hub with high confidence. First, the following lemma inspires the design of early termination optimization.

Lemma 3. *Given a query node q, if there exists a node $u \in \mathcal{V}^*$ such that $I(\bar{u}) \geq \lambda|V|$ and $I(u) > N(q)$, then the answer to the personalized HUB query is `FALSE`.*

Proof. For any node $u \in \mathcal{V}^*$, any $v_1 \in \mathcal{I}(\bar{u})$ and $v_2 \in \mathcal{I}(u)$ must be neighbors in H. Hence, each node in $\mathcal{I}(\bar{u})$ has at least $I(u)$ neighbors. Since $I(u) > N(q)$ and $I(\bar{u}) \geq \lambda|V|$, there are at least $\lambda|V|$ nodes with degrees higher than $N(q)$. $\square$

Based on Lemma 3, we can terminate the *sketch propagation* process when the estimations $\widetilde{I}(\bar{u})$ and $\widetilde{I}(u)$ of $I(\bar{u})$ and $I(u)$ for $u \in \mathcal{V}^*$ by KMV sketches satisfy $\widetilde{I}(\bar{u}) \geq \lambda|V|$

Algorithm 2: `OptimizedReceive`

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

1 $\texttt{Receive}(u, \mathcal{K})$;
2 $\widetilde{I} \leftarrow 0$ and $\widetilde{I}' \leftarrow 0$;
3 **for** $t = 1$ **to** θ **do**
4   **if** $|\mathcal{K}[\bar{u}][t]| = k$ **then** $\widetilde{I} \leftarrow \widetilde{I} + \frac{\max(\mathcal{K}[\bar{u}][t])}{\theta}$;
5   **else** $\widetilde{I} \leftarrow \widetilde{I} + \frac{k}{(|\mathcal{K}[\bar{u}][t]|+1)\cdot\theta}$;
6   **if** $|\mathcal{K}[u][t]| = k$ **then** $\widetilde{I}' \leftarrow \widetilde{I}' + \frac{\max(\mathcal{K}[u][t])}{\theta}$;
7   **else** $\widetilde{I}' \leftarrow \widetilde{I}' + \frac{k}{(|\mathcal{K}[u][t]|+1)\cdot\theta}$;
8 $\widetilde{I} \leftarrow k/\widetilde{I} - 1$ and $\widetilde{I}' \leftarrow k/\widetilde{I}' - 1$;
9 **if** $\widetilde{I} \geq \lambda|V|$ and $\widetilde{I}' \geq N(q)$ **then return** *TRUE*;
10 **return** *FALSE*;

and $\widetilde{I}(u) > N(q)$. Algorithm 2 presents the procedure of `OptimizedReceive`, an optimized version of `Receive` at Line 20 in Algorithm 1 with early termination. Specifically, `OptimizedReceive` returns whether the sketch propagation should end at node u. It calls `Receive` to construct the KMV sketch for the image of u (Line 1) and estimates $I(\bar{u})$ and $I(u)$ accordingly (Lines 2–8). Finally, it returns whether the estimated $\widetilde{I}(\bar{u})$ and $\widetilde{I}(u)$ have satisfied the early termination conditions (Lines 9–10).

Example 5. *Continuing with Example 4, Fig. 2(b) demonstrates the sketch propagation process for a personalized* HUB *query with* $q = a_4$ *and* $\lambda = 0.4$. *Once we obtain the sketch* $\mathcal{K}(\mathcal{I}(\bar{p}_1)) = \{0.2, 0.4\}$*, we have* $\widetilde{I}(\bar{p}_1) = \frac{2}{0.4} - 1 = 4$ *and* $\widetilde{I}(p_1) = \frac{2}{0.6} - 1 = 2.3$. *Thus, there are* $\widetilde{I}(\bar{p}_1) > \lambda|V|$ *nodes with an estimated degree larger than* $N(a_4) = 2$. *Therefore, the sketch propagation process can be early terminated with a decision* `FALSE`*.*

The following theorem bounds the probability that early termination yields false negative results.

Theorem 2. *Let* $\theta = O(\frac{\Lambda}{\epsilon k}\log\frac{|V|}{p})$*, the probability that a personalized* HUB *query with answer* `TRUE` *is answered with* `FALSE` *due to early termination at node* u *is at most* $2 \cdot p^{(\beta-1)/\epsilon}$*, where* $\beta = \min(\frac{\widetilde{I}(\bar{u})}{\lambda|V|}, \frac{\widetilde{I}(u)}{N(q)})$*.*

Proof. We first observe that for any $\delta > 0$, we have

$$O(\frac{\Lambda}{\epsilon k}\log\frac{|V|}{p}) = O(\frac{\Lambda}{\delta k}\log\frac{1}{p^{\delta/\epsilon}}) \tag{2}$$

By Lemma 1 and Eqn. 2, if $\theta = O(\frac{\Lambda}{\epsilon k}\log\frac{|V|}{p})$, we have

$$\widetilde{I}(\bar{u}) \leq (1+\delta)\cdot I(\bar{u});\ \widetilde{I}(u) \leq (1+\delta)\cdot I(u). \tag{3}$$

Each holds with probability at least $1 - p^{\delta/\epsilon}$. According to Eqn. 3, for any $\delta < \beta - 1$, we have

$$I(\bar{u}) \geq \frac{\widetilde{I}(\bar{u})}{1+\delta} \geq \frac{\beta\lambda|V|}{1+\delta} > \lambda|V| \tag{4}$$

and

$$I(u) \geq \frac{\widetilde{I}(u)}{1+\delta} \geq \frac{\beta N(q)}{1+\delta} > N(q) \tag{5}$$

holds simultaneously with probability $1 - 2\cdot p^{(\beta-1)/\epsilon}$.

Note that the early termination occurring at node u leads to a false negative answer only if $\widetilde{I}(\bar{u}) \geq \lambda|V|$ and $\widetilde{I}(u) > N(q)$, but $I(\bar{u}) < \lambda|V|$ or $I(u) \leq N(q)$. Thus, a personalized HUB query with answer `TRUE` is incorrectly answered due to an early termination at u with probability at most $2\cdot p^{(\beta-1)/\epsilon}$. $\square$

IV. Extension to H-index

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

Definition 8 (h-index)**.** *Given a node* $u \in H$*, the h-index of* u *is the largest integer* $h(u)$ *such that* u *has no fewer than* $h(u)$ *neighbors of degrees at least* $h(u)$*.*

A. Pivot-based Algorithm for 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., $Q_u^+ = \{v \in V \,|\, h(v) \geq h(u)\}$ and $Q_u^- = \{v \in V \,|\, h(v) < h(u)\}$. 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 no less than $h(u)$ and thus are ranked higher than u, or $Q_u^- \cap S_0^+(v_\lambda) = \varnothing$ if $u \in S_0^-(v_\lambda)$ similarly. This 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 subsequently. In contrast, if $u \in S_0^-(v^\lambda)$, the nodes in Q_u^+ may contain those in $S_0^+(v^\lambda)$ and $S_0^-(v^\lambda)$, which require further partitioning in subsequent iterations. Meanwhile, all nodes in Q_u^- must not be in $S_0^+(v^\lambda)$ and are excluded from consideration. The iterative process above proceeds until all nodes have been decided. Then, all nodes in $S_0^+(v^\lambda)$ are added to the result.

Pivot-based Partitioning. 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 $u, v \in V$, we have $h(v) \geq h(u)$ iff 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): Estimate $h(u)$. We first employ *sketch propagation* on the matching graph to estimate the degrees of all nodes in H. Subsequently, at each iteration, we scan 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.

Algorithm 3: Pivot-based Algorithm

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

Output: The set of hubs S

1 **forall** $u \in \mathcal{V}^*$ **do**
2   **for** $t = 1$ **to** θ **do**
3     $\mathcal{K}[u][t] \leftarrow \varnothing$;
4   **if** $u \in \mathcal{V}_0$ **do** add $r(u) = \texttt{Rand}(0, 1)$ to $\mathcal{K}[u][t]$;
5 $\widetilde{N} \leftarrow \texttt{Propagate}(\mathcal{K})$;
6 $Q \leftarrow V$ and $S \leftarrow \varnothing$;
7 **while** $|Q| > 0$ **do**
8   Sample a random node from Q as the pivot node u;
9   Sort nodes in $\mathcal{N}(u)$ in descending order by $\widetilde{N}$;
10   Initialize $\widetilde{h}(u) \leftarrow 0$;
11   **forall** $v \in \mathcal{N}(u)$ *and* $\widetilde{N}(v) \geq \widetilde{h}(u)$ **do**
12     $\widetilde{h}(u) \leftarrow \widetilde{h}(u) + 1$;
13   **forall** $v \in \mathcal{V}^*$ **do**
14   **for** $t = 1$ **to** θ **do**
15       $\mathcal{K}[v][t] \leftarrow \varnothing$;
16   **if** $v \in \mathcal{V}_0$ *and* $\widetilde{N}(v) \geq \widetilde{h}(u)$ **then**
17   Add $r(v) = \texttt{Rand}(0, 1)$ to $\mathcal{K}[v][t]$;
18   $\widetilde{h} \leftarrow \texttt{Propagate}(\mathcal{K})$;
19   Initialize $Q_u^+ \leftarrow \varnothing$ and $Q_u^- \leftarrow \varnothing$;
20   **forall** $v \in Q$ **do**
21   **if** $\widetilde{h}(v) \geq \widetilde{h}(u)$ **do** add v to Q_u^+;
22   **else** add v to Q_u^-;
23   **if** $|Q_u^+| + |S| \leq \lambda|V|$ **then**
24     $Q \leftarrow Q_u^-$ and $S \leftarrow S \cup Q_u^+$;
25   **if** $|S| \leq \lambda|V|$ **then** $S \leftarrow S \cup \{u\}$;
26   **else** $Q \leftarrow Q_u^+$;
27 **return** S;

Stage (II): Estimate Q_u^+ and Q_u^-. Once we have the approximate h-index $\widetilde{h}(u)$, we proceed to eliminate the 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 KG $\mathcal{G}$, 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 array $\mathcal{K}$ 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 $\mathcal{K}$ of KMV sketches for the nodes with degrees greater than $\widetilde{h}(u)$ (Lines 13–17). After that, the Propagate function is performed over $\mathcal{K}$ to estimate the number of neighbors with degrees greater than $\widetilde{h}(u)$ for each node in Q (Line 18). Accordingly, it divides Q into Q_u^+ and Q_u^- according to $\widetilde{h}$ (Lines 19–22) and updates the result set S and the candidate node set Q based on Q_u^+ and Q_u^- (Lines 23–26). Finally, when Q is empty, the result set S is returned (Line 27). The ties are broken arbitrarily.

Theoretical Analysis. Algorithm 3 returns the correct answer for any h-index-based HUB query 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. *For any $\delta, p \in (0, 1)$, if $\theta = \Theta(\frac{\Lambda}{\delta k} \log \frac{|V|}{p})$, 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 1 with the union bound over V, when $\theta = \Theta(\frac{\Lambda}{\delta k} \log \frac{|V|}{p})$ 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 of at least $(1-\delta)h(u)$. 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 greater than $(1+\delta)h(u)$, and $\widetilde{h}(u) \leq (1 + \delta)h(u)$ with probability at least $1 - p$. □

The following lemma indicates that the pivot-based partitioning method divides candidate nodes into ϵ-separable sets.

Lemma 5. *For any $\delta' \in (0, 1)$ and pivot node u, by setting $\theta = \Theta(\frac{\Lambda}{\delta' k} \log \frac{|V|}{p})$, we have $S_{\delta'}^-(u) \subseteq Q_u^-$ and $S_{\delta'}^+(u) \subseteq Q_u^+$ 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 $\theta = \Theta(\frac{\Lambda}{\delta k} \log \frac{|V|}{p})$, $\widetilde{N}(v') < (1 + \delta)N(v') < (1 + \delta)(1 - \delta')h(u) < \frac{(1+\delta)(1-\delta')}{1-\delta}\widetilde{h}(u)$. 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 $\widetilde{N}_u(v) \le (1+\delta)N_u(u)$ with probability at least $1-p$, and thus $\widetilde{N}_u(v) \le (1+\delta)N_u(v) < (1+\delta)(1-\delta')h(u) \le \frac{(1+\delta)(1-\delta')}{1-\delta}\widetilde{h}(u) \le \widetilde{h}(u)$, i.e., v is added to Q_u^- with probability at least $1-p$. Thus, by setting $\theta = \Theta(\frac{\Lambda}{\frac{\delta'}{2-\delta'}k}\log\frac{|V|}{p}) = \Theta(\frac{\Lambda}{\delta' k}\log\frac{|V|}{p})$, 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 $\theta = \Theta(\frac{\Lambda}{\delta' k}\log\frac{|V|}{p})$. □

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 3. *Let $\theta = \Theta(\frac{\Lambda}{\epsilon k}\log\frac{|V|}{p})$. Algorithm 3 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 for the ranges of the h-index $h(u)$ for the pivot node u. For each case, we show the correctness of the first iteration in Algorithm 3, while subsequent iterations are proven by induction. By selecting $\delta' < \frac{\epsilon}{2}$ as Lemma 5, we have:

- **Case 1** ($h(u) > (1+\frac{\epsilon}{2})h(v^\lambda)$)**:** Each v with $h(v) \le h(v^\lambda)$ will be added to Q_u^-. Thus, $|Q_u^-| \ge (1-\lambda)|V|$. 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 $h(v) \ge h(v^\lambda)$ will be added to Q_u^+. Thus, $|Q_u^+| \ge \lambda|V|$. 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 $h(v) \ge (1+\epsilon)h(v^\lambda)$ will be added to Q_u^+. And any v' with $h(v') \le (1-\epsilon)h(v^\lambda)$ will be added to Q_u^-. Thus, Q_u^+, which contains all nodes in $S_\epsilon^+(v^\lambda)$, will be 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 eliminated or used as the candidate set Q for the next iteration.

Considering all of the above cases, it follows that, for any randomly chosen pivot u, none of the nodes in $S_\epsilon^-(v^\lambda)$ is added to 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. Thus, we prove the theorem by applying the union bound over all iterations. □

Finally, the amortized time complexity of Algorithm 3 is $O(\frac{\Lambda|\mathcal{E}^*|}{\epsilon}\cdot\log^2\frac{|V|}{p})$ because it invokes *sketch propagation* $O(\log|V|)$ times in amortization. Its space complexity is $O(k\theta|\mathcal{V}^*| + |\mathcal{E}^*|)$, which is equal to that of Algorithm 1.

B. Early Termination for Personalized HUB

Given a query node q, the personalized HUB query based on the h-index can also be answered by performing the pivot-based partitioning method with q as the pivot node to obtain Q_u^+ and returning `FALSE` if $|Q_u^+| > \lambda|V|$ or `TRUE` otherwise. Following Lemma 5, the following corollary indicates that the above algorithm provides correct answers for approximate personalized HUB queries w.h.p.

Corollary 2. *Let $\theta = \Theta(\frac{\Lambda}{\epsilon k}\log\frac{|V|}{p})$ for any $\epsilon \in (0,1)$. Algorithm 3 correctly answers an ϵ-approximate personalized HUB query based on the h-index with probability at least $1-p$.*

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

Lemma 6. *Given a query node q and the quantile parameter λ, if there exists a node $u \in \mathcal{V}^*$ such that $I(\bar{u}) > h(q)$ and $I(u) \ge \max(h(q), \lambda|V|)$, then the personalized HUB query returns `FALSE`.*

Proof. Each pair of nodes in $\mathcal{I}(u)$ and $\mathcal{I}(\bar{u})$ are neighbors of each other in H according to the proof of Lemma 3. Thus, each node $v_1 \in \mathcal{I}(u)$ has at least $I(\bar{u})$ neighbors with degrees larger than or equal to $I(u)$. Since $I(\bar{u}) > h(q)$ and $I(u) \ge \max(h(q), \lambda\cdot|V|)$, there exist at least $I(u) \ge \lambda|V|$ nodes with h-indexes greater than or equal to $h(q)$. Therefore, q is not a hub based on the h-index. □

Based on Lemma 6, the sketch propagation can be 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 $\widetilde{I}_f(u) > N(q)$ and $\widetilde{I}_b(u) \ge \max(N(q), \lambda|V|)$, the algorithm can be terminated early since $N(q)$ is an upper bound of $h(q)$. once we obtain $\widetilde{h}(q)$ after *Stage (I)*, the algorithm can be terminated without entering *Stage (II)* if there exists a node $u \in \mathcal{V}^*$ such that $\widetilde{I}(\bar{u}) > \widetilde{h}(q)$ and $\widetilde{I}(u) \ge \max(\widetilde{h}(q), \lambda|V|)$.

The following theorem bounds the probability that an early termination after *Stage (I)* yields a false negative results for personalized HUB queries based on h-index.

Theorem 4. *Let $\theta = O(\frac{\Lambda}{\epsilon k}\log\frac{|V|}{p})$, the probability that a personalized HUB query based on h-index with answer `TRUE` is incorrectly answered with `FALSE` due to early termination after Stage (I) at node u is at most $(p + 2p^{((1-\epsilon)\beta-1)/\epsilon})$, where $\beta = \min(\frac{\widetilde{I}(\bar{u})}{\widetilde{h}(q)}, \frac{\widetilde{I}(u)}{\lambda|V|})$.*

Proof. By Lemma 4 and Eqn. 2, for any $\delta < (1-\epsilon)\beta - 1$,

$$I(\bar{u}) \ge \frac{\widetilde{I}(\bar{u})}{1+\delta} \ge \frac{\beta\widetilde{h}(q)}{1+\delta} \ge \frac{\beta(1-\epsilon)h(q)}{1+\delta} > h(q) \tag{6}$$

and

$$\begin{aligned} I(u) \ge \frac{\widetilde{I}(u)}{1+\delta} &\ge \frac{\beta\cdot\max(\widetilde{h}(q), \lambda|V|)}{1+\delta} \\ &\ge \frac{\beta\cdot\max((1-\epsilon)h(q), \lambda|V|)}{1+\delta} > \max(h(q), \lambda|V|) \end{aligned} \tag{7}$$

holds with probability at least $1 - 2p^{((1-\epsilon)\beta-1)/\epsilon} - p$.

Note that the early termination occurs at node u leads to a false negative answer only if $\widetilde{I}(\bar{u}) \ge h(q)$ and $\widetilde{I}(u) > \max(h(q), \lambda|V|)$. However, we have $I(\bar{u}) < h(q)$ or $I(u) \le \max(h(q), \lambda|V|)$. Thus, a personalized HUB query with answer `TRUE` is incorrectly answered due to an early termination at u with probability at most $p + 2p^{((1-\epsilon)\beta-1)/\epsilon}$. □

TABLE II

STATISTICS OF DATASETS USED IN THE EXPERIMENTS, WHERE $|\mathcal{L}(\mathcal{V})|$ AND $|\mathcal{L}(\mathcal{E})|$ ARE THE NUMBERS OF NODE AND EDGE TYPES AND $|\mathbb{M}|$ IS THE NUMBER OF EVALUATED META-PATHS.

Dataset	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	$\|\mathcal{L}(\mathcal{V})\|$	$\|\mathcal{L}(\mathcal{E})\|$	$\|\mathbb{M}\|$
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	2,230
BPM	0.6M∼19.2M	60M∼1.92B	2	1	1

V. EXPERIMENTS

A. Experimental Setup

Datasets. The statistics of the datasets used in the experiments are reported in Table II. `IMDB` is a KG describing actors and directors of movies. `ACM` and `DBLP` are academic KGs denoting researchers and their publications with corresponding venues. `PubMed` is a biomedical KG representing the relationships between genes, chemicals, and diseases of different species. `FreeBase` is a general-purpose KG developed by Google. `Yelp` is a KG built from user ratings and comments on businesses at different locations. `BPM` represents a series of synthetic datasets generated by the Bipartite Preferential Model [8], [53], [54] for scalability tests (§ V-E). Please refer to our technical report [55] for the generation of `BPM`.

Query Generation. For each dataset, we use AnyBURL [56] 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 efficient processing of HUB queries even more necessary. $|\mathbb{M}|$ in Table II shows the number of meta-paths found on each dataset. For personalized HUB queries, we randomly sample 10 nodes from $\mathcal{V}_0$ as query nodes for each meta-path.

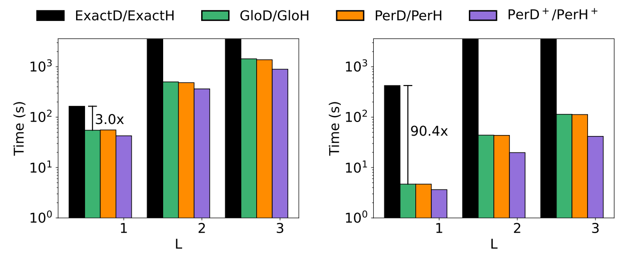
Compared Methods. To the best of our knowledge, there have been no prior studies on the HUB problem. Therefore, we compare our proposed methods with exact algorithms based on the state-of-the-art materialization methods. The compared methods are listed as follows:

- ExactD and ExactH find exact hub nodes by computing the respective degree and h-indexes of each node using the generic worst-case optimal join [47]–[50] (§ II-C).
- BoolAP [10] returns exact hubs based on both degree and h-index by materializing the relational graph through boolean matrix multiplication. BoolAP$^+$ is a variant of BoolAP specially optimized for KGs with locally dense regions.
- GloD and PerD apply the sketch propagation framework (§ III-A) to estimate the degrees of all nodes in H. GloD returns the set of hubs based on estimated degrees, whereas PerD decides whether a query node q is a hub.
- PerD$^+$ optimizes PerD with early termination (§ III-B).
- GloH and PerH apply the pivot-based algorithm (§ IV-A) based on estimated h-indexes. GloH recursively partitions the nodes until all hubs are found, while PerH checks if a query node q is a hub using q as a pivot.
- PerH$^+$ optimizes PerH with early termination (§ IV-B).

TABLE III

STATISTICS OF META-PATHS IN THE EXPERIMENTS, WHERE $|\mathcal{E}^*|$ AND ρ^* DENOTE THE AVERAGE NUMBER OF EDGES AND THE DENSITY OF MATCHING GRAPHS INDUCED BY META-PATHS ON EACH DATASET.

	IMDB	ACM	DBLP	PubMed	FreeBase
$\|\mathcal{E}^*\|$	22.3	133154	28941.4	23588.5	63526.3
ρ^*	0.975	22.64	1.598	1.614	1.808



(a) Degree (b) H-index

Fig. 3. Running times of different algorithms on `Yelp` for meta-paths with varying length L.

Parameter Settings. By default, we set $\lambda = 0.05$ to find the hubs as the top 5% of the nodes with the highest degrees or h-indexes in a relational graph. Our results in § V-C will show that h-index-based HUB queries are well approximated using smaller values of θ and k compared to degree-based HUB queries. Hence, 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, PerH$^+$), we set $\theta = 8$ and $k = 4$. These settings also demonstrate that our proposed methods produce 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. The algorithms return a set S of $\lambda \cdot |V|$ nodes. However, there might be more than $\lambda \cdot |V|$ nodes satisfying the criteria of Problem 1 due to the tied values with $c(v^\lambda)$. Thus, we exclude nodes with centrality scores equal to $c(v^\lambda)$ in the recall calculation. To assess the effectiveness of PerD, PerD$^+$, PerH, and PerH$^+$ for personalized HUB queries, we record their accuracy in deciding whether a node q has a centrality score equal to or greater than $c(v^\lambda)$. We report the average F1 and accuracy scores across all meta-paths in Figs. 6 and 7. To evaluate the theoretical correctness of our proposed methods, we report the value of ϵ in Fig. 8 to measure the approximation ratio our methods achieved. For efficiency evaluation, we report the average CPU time of each algorithm to process a HUB query.

Environment. All experiments were carried out 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. Our source code and data are published at https://github.com/YudongNiu/HUB.

TABLE IV

RESULTS FOR THE EFFICIENCIES OF DIFFERENT ALGORITHMS. FOR EXACT METHODS BoolAP, BoolAP$^+$, ExactD, AND ExactH, THE RUNNING TIME (IN SECONDS) ON EACH DATASET IS REPORTED. FOR GloD, PerD$^+$, GloH, AND PerH$^+$, THE RUNNING TIME (IN SECONDS), AS WELL AS THE SPEEDUP RATIOS OVER THE FASTEST EXACT METHOD, ARE REPORTED. WE ALSO REPORT THE EXTRACTION TIME (IN MILLISECONDS) OF MATCHING GRAPH ON EACH DATASET AS $T(\mathcal{G}^*)$, WHICH IS ALREADY INCLUDED IN THE RUNNING TIME OF EACH METHOD.

Dataset	BoolAP	BoolAP$^+$	**Degree Centrality**				**H-index Centrality**				$T(\mathcal{G}^*)$
			ExactD	GloD	PerD	PerD$^+$	ExactH	GloH	PerH	PerH$^+$	
IMDB	0.00017	0.004	0.0009	0.0013 (**0.13**×↓)	0.0016 (**0.106**×↓)	0.0017 (**0.1**×↓)	0.0013	0.001 (**0.17**×↓)	0.0015 (**0.11**×↓)	0.0015 (**0.11**×↓)	0.88ms
ACM	19.33	0.3	4.77	1.77 (**0.17**×↓)	2.04 (**0.15**×↓)	0.81 (**0.37**×↓)	8.01	0.19 (**1.6**×↑)	0.25 (**1.2**×↑)	0.076 (**4**×↑)	4.18ms
DBLP	4.75	3.43	7.95	0.63 (**5.44**×↑)	0.67 (**5.12**×↑)	0.27 (**12.7**×↑)	14.59	0.23 (**15**×↑)	0.10 (**34.3**×↑)	0.049 (**70**×↑)	3.58ms
PubMed	15.36	10.9	5.96	0.45 (**13.2**×↑)	0.35 (**17.1**×↑)	0.13 (**45.8**×↑)	12.11	0.09 (**121**×↑)	0.056 (**195**×↑)	0.032 (**340**×↑)	8ms
FreeBase	177.04	110.7	25.15	0.87 (**29.0**×↑)	0.92 (**27.4**×↑)	0.41 (**60.9**×↑)	50.5	0.26 (**191**×↑)	0.19 (**268**×↑)	0.13 (**379**×↑)	26ms

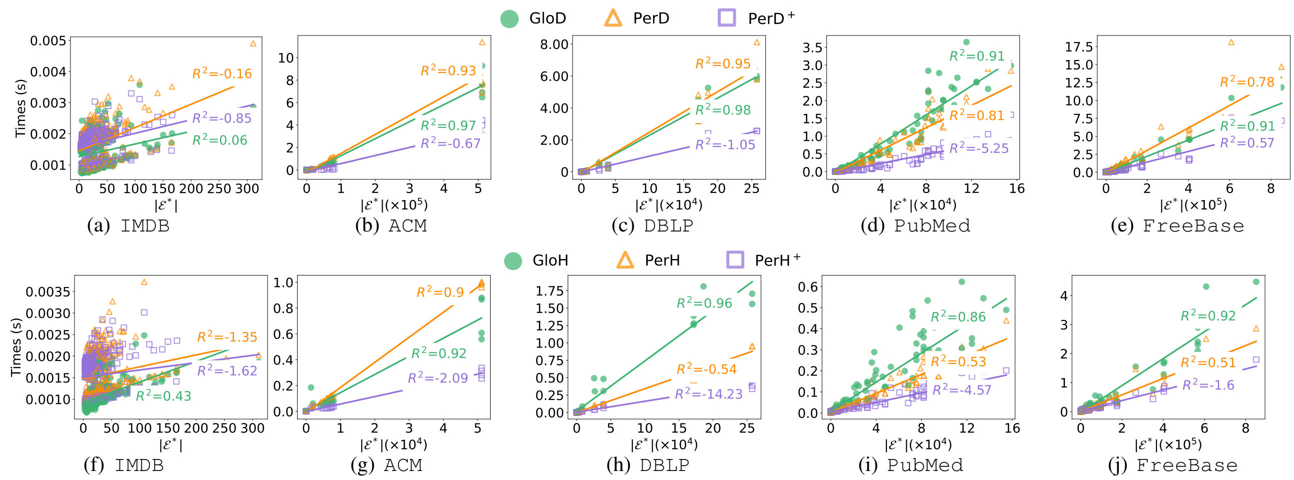


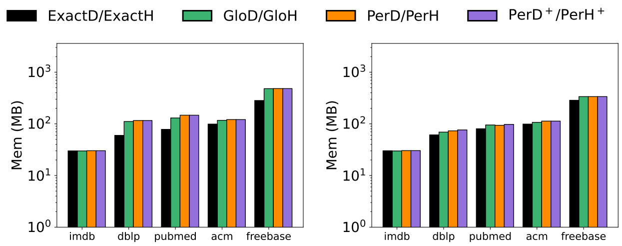
Fig. 4. Running time of our methods for each meta-path on FreeBase with x-axis representing the number of edges $|\mathcal{E}^*|$ in the corresponding matching graph. Lines of corresponding colors indicate the linear regression results of each method's running time with respect to $|\mathcal{E}^*|$. Subfigures (a)-(e): degree based hub queries; Subfigures (f)-(j): h-index based hub queries.

B. Efficiency Evaluation

In Table IV, we present the execution time per query for each method, along with its speedup ratios compared to the fastest exact method across five datasets. Based on these results, we make the following observations. First, sketch propagation-based algorithms are consistently more efficient than exact baselines ExactD and ExactH across all datasets except IMDB and provide more than 10× speedups for degree-based queries (on FreeBase, GloD is 29× faster than ExactD) and up to two orders of magnitude speedups for h-index-based queries (on FreeBase, GloH is 191× faster than ExactH). The speedup over degree-based queries is smaller than h-index-based queries because h-index-based queries are accurately approximated with smaller k as discussed in § V-C. GloD is slightly slower than ExactD for degree-based queries on IMDB since matching graphs of meta-paths on IMDB are of relatively small sizes. Specifically, as demonstrated in Table III, the average number of edges in matching graphs on IMDB is 22.3, which is much smaller than matching graphs of meta-paths on other datasets. Therefore, the worst-case optimal join is performed quickly over the matching graph on IMDB, whereas our methods require additional costs for sketch construction and degree estimation. Nevertheless, GloD answers HUB queries on IMDB in 1.3 milliseconds. BoolAP$^+$, which is specially optimized for relational graph materialization based on matching graphs with dense regions, demonstrates exceptional efficiency on ACM, where the matching graphs have a high average density of 22.64. Nevertheless, BoolAP$^+$ is much slower than our sketch propagation methods across other datasets, where the matching graphs are of lower average density, and is less scalable even than ExactD and ExactH. Thus, in the following, we will only compare our methods with ExactD and ExactH. Second, the algorithms for personalized queries, i.e., PerD$^+$ and PerH$^+$, benefit from early termination optimization and achieve more than 2× additional speedups over PerD and PerH.

Effect of Selected Meta-Paths. We investigate how the properties of specific meta-paths affect the efficiency of different methods. Fig. 3 presents the running time of different methods on Yelp varying the length of the meta-path L. We can observe that L has a significant impact on efficiency, and exact methods ExactD and ExactH exceed the 1-hour limit per meta-path when $L = 2, 3$. On the other hand, our methods based on *sketch propagation* can efficiently process all queries up to $L = 3$. Specifically, GloD and GloH can answer HUB queries based on degree and h-index in 1,438 and 114 seconds, respectively. For personalized HUB queries, PerH$^+$ and PerD$^+$ handle query meta-paths of $L = 3$ in 893 and 42 seconds.

Fig. 4 further presents the running time of our sketch propagation-based methods for each meta-path across datasets, along with the linear regression results of each method's running time with respect to the number of edges $|\mathcal{E}^*|$ in the corresponding matching graph. We have the following observations. *First*, the running time of all our methods generally increases with the increase of $|\mathcal{E}^*|$, which aligns



(a) degree

(b) h-index

Fig. 5. Memory consumption of compared methods.

with the fact that the time complexity of our methods are proportional to $|\mathcal{E}^*|$. *Second*, the running time of GloD and PerD exhibits a strong correlation with $|\mathcal{E}^*|$. For example, on FreeBase, we have R^2 coefficients of 0.91 and 0.78 for GloD and PerD respectively. On the other hand, the running time of PerD$^+$ demonstrates a weaker correlation with $|\mathcal{E}^*|$, with a R^2 coefficient of 0.57 on FreeBase. By employing early termination during sketch propagation, PerD$^+$ avoids propagating sketches over the entire matching graph, thereby reducing its dependence on $|\mathcal{E}^*|$. A similar observation can be made for GloH and PerH, whose running time is more closely correlated with $|\mathcal{E}^*|$ compared to PerH$^+$. *Third*, the running time of GloD and PerD generally exhibits a stronger correlation with $|\mathcal{E}^*|$ compared to the running time of GloH and PerH. For example, on PubMed, the R^2 coefficient is 0.91 for GloD and 0.86 for GloH. This difference arises because GloH requires multiple iterations of the sketch propagations based on randomly selected pivot nodes, making its running time dependent on both the number of sketch propagations and the time consumption for each propagation, which is influenced by $|\mathcal{E}^*|$. In contrast, GloD requires a single pass of sketch propagation for the KMV sketch construction.

We exclude Yelp from the remaining experiments on effectiveness evaluations to ensure a fair comparison, as ExactD and ExactH cannot complete on all the meta-paths.

We then report the memory consumption of different methods in Fig. 5. We can observe from Fig. 5(a) that GloD, PerD, and PerD$^+$ take relatively more memory than the exact algorithm ExactD to maintain KMV sketches for images. Nevertheless, our methods take at most 1.93 times more 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 fewer KMV sketches for accurate estimation. Specifically, our methods take at most 1.21 times more memory than ExactH (on PubMed).

C. Effectiveness Evaluation

Results for Degree-based HUB Queries. Figs. 6(a)–6(e) illustrate the effectiveness of GloD, PerD, and PerD$^+$ on each dataset by varying the parameter λ. We observe that GloD achieves F1-scores of more than 0.85 across all datasets for HUB queries, and PerD and PerD$^+$ consistently achieve accuracy rates of at least 0.98 for personalized HUB queries. Moreover, GloD provides more accurate answers with increasing λ and achieves an F1-score of over 0.9 when $\lambda = 0.05$. The reason is that for a larger λ, the $(1-\lambda)$-quantile node v^λ has a smaller degree, and HUB queries apply less strict requirements to the hubs, which can be more easily approximated. The effectiveness of GloD varies across datasets: It achieves higher F1-scores on IMDB and ACM than on the other three datasets. The variation in the effectiveness of GloD is caused by different degree distributions of relational graphs in each dataset. Specifically, matching graphs with higher density on ACM result in densely connected relational graphs, where on average 72.7% of the nodes share the same degree as v^λ. These nodes are excluded from the recall calculation, leading to a higher F1-score. In contrast, the percentage of nodes with the same degree as v^λ is 32.25% on DBLP, 38.64% on PubMed, and 30.6% on FreeBase. GloD achieves a high F1-score on IMDB because the matching graphs are relatively small, allowing accurate degree estimation in the relational graphs.

Figs. 6(f)–6(j) 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 rates due 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, which is relatively easy due to the significant difference between the degrees of q and v^λ. 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.

Figs. 6(k)–6(o) show the effectiveness of GloD, PerD, and PerD$^+$ on each dataset by varying the 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 changes in θ.

Results for HUB Queries based on H-index. The effectiveness results of GloH, PerH, and PerH$^+$ are presented in Fig. 7. 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 in the same parameter setting. 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 nodes with h-index values equal to $c(v^\lambda)$ from the recall calculation, h-

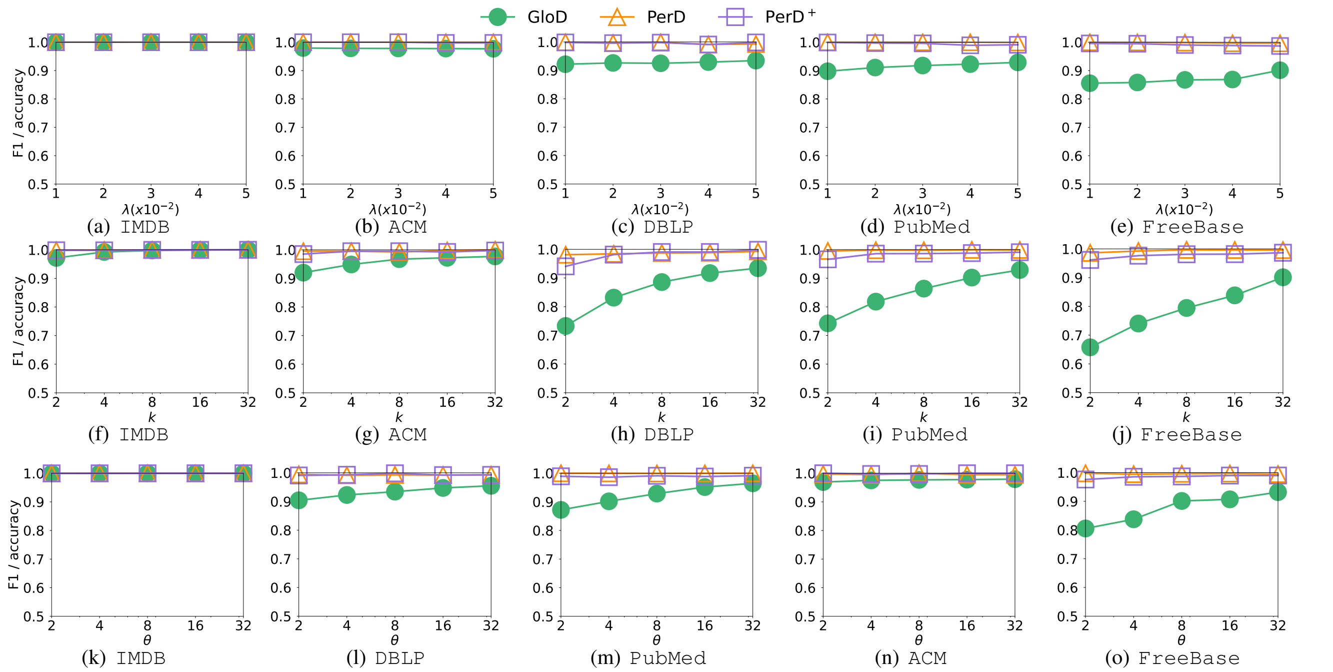


Fig. 6. Effectiveness of GloD, PerD, and PerD$^+$ for HUB queries based on degree centrality with varying parameters. Subfigures (a)-(e): varying parameter λ. Subfigures (f)-(j): varying parameter k. Subfigures (k)-(o): varying parameter θ.

index-based HUB queries become much easier to approximate.

Results for Approximation Ratio ϵ. Fig. 8 presents the approximation ratio ϵ achieved by our sketch propagation-based methods varying sketch size k. We have the following observations. *First*, our methods answer Problems 3 and 4 accurately with ϵ consistently remaining below 0.6 under the default parameter setting of k and θ. Specifically, the approximation ratio ϵ achieved by our methods decreases monotonically with the increase of k across datasets and approaches zero when $k = 32$ except on ACM. This exception arises due to relational graphs with a substantial proportion of high-degree nodes on ACM induced by matching graphs with high density, thereby requiring a larger value of k to effectively identify hubs in these relational graphs, as indicated by Theorems 1 and 3. *Second*, given fixed value of k, GloH, PerH and PerH$^+$ can achieve smaller ϵ compared to GloD, PerD and PerD$^+$. This observation complies with our observation from Fig. 6 and 7 that h-index-based queries are easier to approximate due to large number of tied h-index values in relational graphs. Note that PerD and PerH achieve the same approximation ratio as GloD and GloH, respectively, since the query node for personalized hub queries can be selected as the node leading to the maximum ϵ in global hub queries.

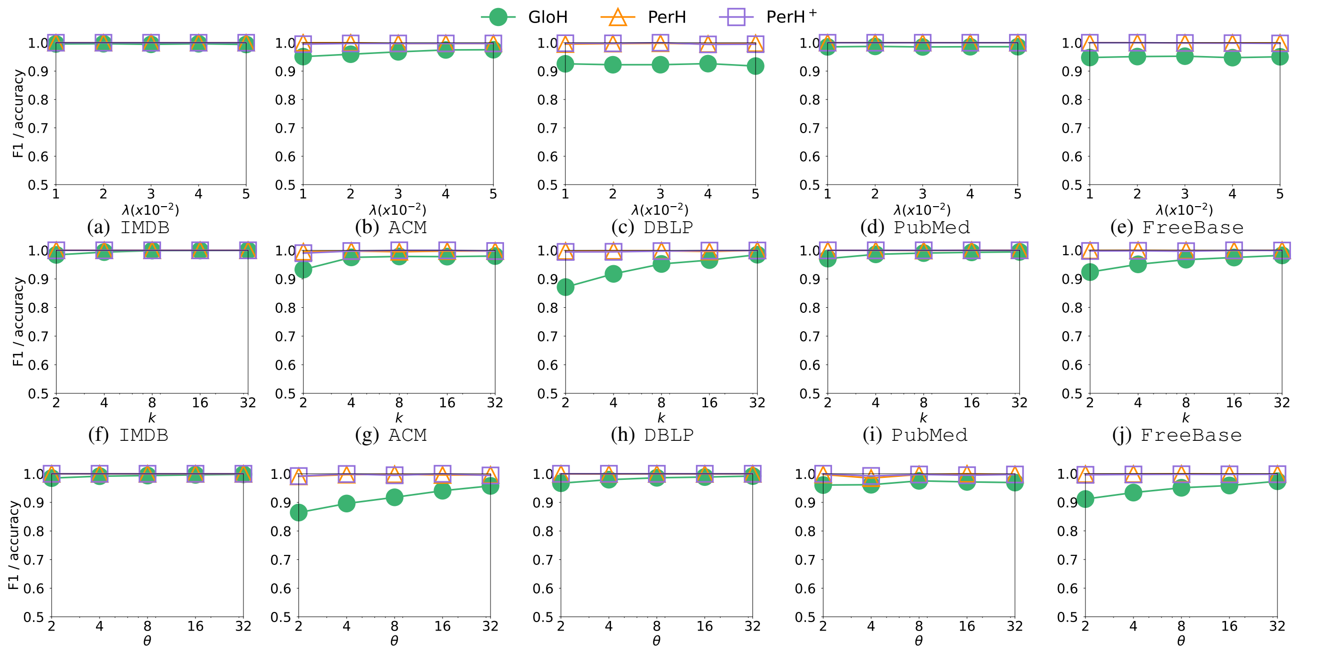
Summary. GloD, PerD, and PerD$^+$ achieve F1-scores or accuracy rates greater than 0.9 across all datasets when $k = 32$ and $\theta = 8$ for degree-based HUB queries with $\lambda = 0.05$. Similarly, for h-index-based HUB queries, GloH, PerH, and PerH$^+$ achieve F1 scores or an accuracy greater than 0.9 across all datasets when $k = 4$ and $\theta = 8$. All of these results confirm the empirical effectiveness as well as theoretical correctness of our proposed methods.

D. *Influence Analysis*

We further verify the effectiveness of HUB for influence analysis on relational graphs. We adopt the weighted cascade model for influence estimation [57], [58]. Fig. 9 reports the ratio between the influence scores of top-5% hubs returned by HUB methods (ExactD, GloD, ExactH, and GloH) and the influence scores of top-5% seeds with the highest influence estimated by a reverse influence sampling (RIS) algorithm for influence maximization [59]. We have the following observations. First, all HUB methods consistently achieve more than 90% of the influence scores achieved by the RIS algorithm, which verifies that the hubs have significant influence over the relational graph and that our proposed methods can serve as effective seeding strategies. Second, the h-index-based hubs (obtained by GloH and ExactH) outperform the degree-based hubs (obtained by GloD and ExactD) and maintain over 95% of the influence scores achieved by the RIS algorithm. In contrast to degree-based hubs, h-index-based hubs have neighbors with higher degrees, which further facilitate the spread of influence to indirect neighbors.

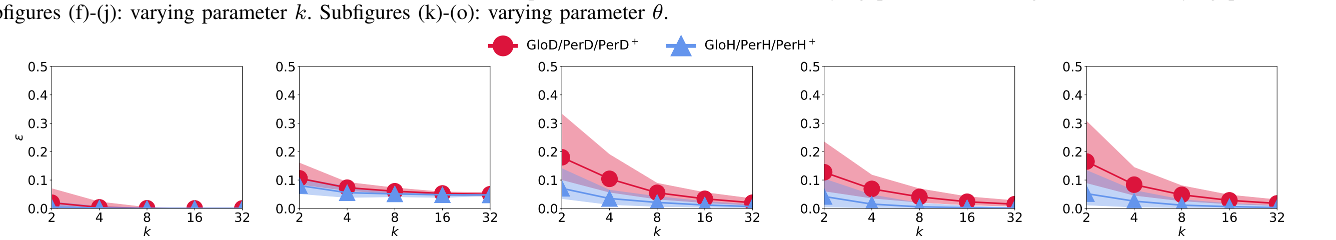
E. *Scalability Test*

We further test the scalability of our methods on large synthetic datasets BPM. Fig. 10 reports the running time and memory consumption of different methods by varying the number of nodes. We make the following observations. The exact methods ExactD and ExactH cannot scale to large datasets. Specifically, ExactD and ExactH can only process queries from the smallest BPM datasets with less than 1M nodes and exceed the 1-hour limit on larger datasets. In contrast, our methods based on *sketch propagation* scale linearly with the size of the datasets and can handle queries on BPM



(a) IMDB (b) ACM (c) DBLP (d) PubMed (e) FreeBase
(f) IMDB (g) ACM (h) DBLP (i) PubMed (j) FreeBase
(k) IMDB (l) DBLP (m) PubMed (n) ACM (o) FreeBase

Fig. 7. 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.



(a) IMDB (b) ACM (c) DBLP (d) PubMed (e) FreeBase

Fig. 8. Average approximation ratio ϵ achieved by our methods by varying the sketch size k, with shaded area denoting the variance of ϵ over 10 executions.

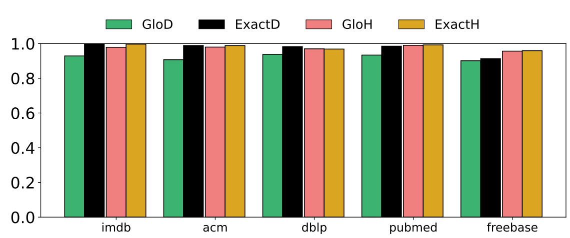
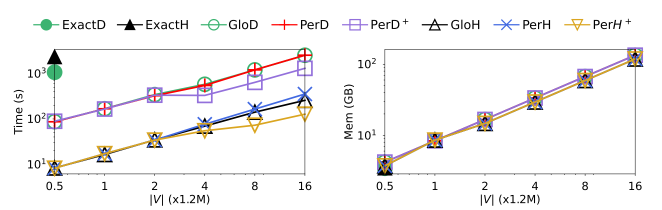


Fig. 9. Ratio between the influence scores achieved by hubs revealed by HUB methods and the highest influence scores estimated by the RIS algorithm.



(a) Running Time (b) Memory Usage

Fig. 10. Runtime and memory consumption of our methods on the BPM dataset.

datasets with up to 192M nodes and 1.92B edges within 2,552 seconds for degree-based queries and 258 seconds for h-index-based queries. In terms of memory consumption, our methods use 118.5 GB and 134 GB memory, respectively, to process queries based on the h-index and degree centrality measures on the largest BPM dataset. The difference between the memory consumption of queries based on different centrality measures is marginal because KMV sketches are memory efficient compared to the matching graph itself.

VI. Conclusion and Future Work

We introduced and studied the problem of finding degree-based and h-index-based hubs on non-materialized relational graphs (HUB). We proposed a sketch propagation framework for approximate degree-based HUB queries and extended it to h-index-based HUB queries with a pivot-based algorithm. We also considered personalized HUB queries by both centrality measures and enhanced efficiency with early termination. Theoretically, our sketch propagation and pivot-based algorithms answer approximate HUB queries correctly with high probability. We verified the effectiveness and efficiency of our proposed algorithms with extensive experimentation on real-world heterogeneous data.

For future work, it would be interesting to extend our framework for hub queries over relational graphs induced by general meta-graphs, which impose substantial technical challenges due to the potentially complex structure of meta-graphs.

Furthermore, another possible direction involves generalizing sketch-based methods for graph analysis tasks over relational graphs other than hub queries.

REFERENCES

[1] J. Jiang, Y. Chen, B. He, M. Chen, and J. Chen, "Spade+: A generic real-time fraud detection framework on dynamic graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 11, pp. 7058–7073, 2024.

[2] Y. Ji, Z. Zhang, X. Tang, J. Shen, X. Zhang, and G. Yang, "Detecting cash-out users via dense subgraphs," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22)*, 2022, pp. 687–697.

[3] K. Li, T. Yang, M. Zhou, J. Meng, S. Wang, Y. Wu, B. Tan, H. Song, L. Pan, F. Yu, Z. Sheng, and Y. Tong, "SEFraud: Graph-based self-explainable fraud detection via interpretative mask learning," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24)*, 2024, pp. 5329–5338.

[4] D. Liu, M. He, J. Luo, J. Lin, M. Wang, X. Zhang, W. Pan, and Z. Ming, "User-event graph embedding learning for context-aware recommendation," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22)*, 2022, pp. 1051–1059.

[5] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "PathSim: Meta path-based top-k similarity search in heterogeneous information networks," *Proc. VLDB Endow.*, vol. 4, no. 11, pp. 992–1003, 2011.

[6] Y. Fang, Y. Yang, W. Zhang, X. Lin, and X. Cao, "Effective and efficient community search over large heterogeneous information networks," *Proc. VLDB Endow.*, vol. 13, no. 6, pp. 854–867, 2020.

[7] Y. Zheng, C. Shi, X. Cao, X. Li, and B. Wu, "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*, 2017, pp. 317–329.

[8] K. Xirogiannopoulos and A. Deshpande, "Extracting and analyzing hidden graphs from relational databases," in *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17)*, 2017, pp. 897–912.

[9] S. Chatzopoulos, T. Vergoulis, D. Skoutas, T. Dalamagas, C. Tryfonopoulos, and P. Karras, "Atrapos: Real-time evaluation of metapath query workloads," in *Proceedings of the ACM Web Conference 2023 (WWW '23)*, 2023, pp. 2487–2498.

[10] Y. Guo, C. Ma, and Y. Fang, "Efficient core decomposition over large heterogeneous information networks," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 2024, pp. 2393–2406.

[11] C. Shi, Y. Li, P. S. Yu, and B. Wu, "Constrained-meta-path-based ranking in heterogeneous information network," *Knowl. Inf. Syst.*, vol. 49, pp. 719–747, 2016.

[12] B. Cao, M. Mao, S. Viidu, and P. S. Yu, "HitFraud: A broad learning approach for collective fraud detection in heterogeneous information networks," in *2017 IEEE International Conference on Data Mining (ICDM)*, 2017, pp. 769–774.

[13] S. Mihara, S. Tsugawa, and H. Ohsaki, "Influence maximization problem for unknown social networks," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '15)*, 2015, pp. 1539–1546.

[14] Y. Li, C. Shi, P. S. Yu, and Q. Chen, "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*, 2014, pp. 553–565.

[15] M. Wu, X. Li, C.-K. Kwoh, and S.-K. Ng, "A core-attachment based method to detect protein complexes in PPI networks," *BMC Bioinform.*, vol. 10, p. 169, 2009.

[16] S. Srihari and H. W. Leong, "A survey of computational methods for protein complex prediction from protein interaction networks," *J. Bioinform. Comput. Biol.*, vol. 11, no. 02, p. 1230002, 2013.

[17] C. Estan, G. Varghese, and M. E. Fisk, "Bitmap algorithms for counting active flows on high-speed links," *IEEE/ACM Trans. Netw.*, vol. 14, no. 5, pp. 925–937, 2006.

[18] N. Alon, Y. Matias, and M. Szegedy, "The space complexity of approximating the frequency moments," *J. Comput. Syst. Sci.*, vol. 58, no. 1, pp. 137–147, 1999.

[19] M. Durand and P. Flajolet, "Loglog counting of large cardinalities," in *Algorithms - ESA 2003, 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003, Proceedings*, 2003, pp. 605–617.

[20] S. Heule, M. Nunkesser, and A. Hall, "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)*, 2013, pp. 683–692.

[21] P. B. Gibbons, "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)*, 2001, pp. 541–550.

[22] L. Lü, T. Zhou, Q.-M. Zhang, and H. E. Stanley, "The h-index of a network node and its relation to degree and coreness," *Nat. Commun.*, vol. 7, no. 1, p. 10168, 2016.

[23] L. Egghe, "Theory and practise of the g-index," *Scientometrics*, vol. 69, pp. 131–152, 2006.

[24] P. W. Olsen, A. G. Labouseur, and J.-H. Hwang, "Efficient top-k closeness centrality search," in *2014 IEEE 30th International Conference on Data Engineering (ICDE)*, 2014, pp. 196–207.

[25] E. Bergamini, M. Borassi, P. Crescenzi, A. Marino, and H. Meyerhenke, "Computing top-k closeness centrality faster in unweighted graphs," *ACM Trans. Knowl. Discov. Data*, vol. 13, no. 5, pp. 53:1–53:40, 2019.

[26] Y. Yang, Y. Fang, X. Lin, and W. Zhang, "Effective and efficient truss computation over large heterogeneous information networks," in *36th IEEE International Conference on Data Engineering (ICDE)*, 2020, pp. 901–912.

[27] Y. Jiang, Y. Fang, C. Ma, X. Cao, and C. Li, "Effective community search over large star-schema heterogeneous information networks," *Proc. VLDB Endow.*, vol. 15, no. 11, pp. 2307–2320, 2022.

[28] K. Das, S. Samanta, and M. Pal, "Study on centrality measures in social networks: a survey," *Soc. Netw. Anal. Min.*, vol. 8, no. 13, pp. 1–11, 2018.

[29] M. Greenwald and S. Khanna, "Space-efficient online computation of quantile summaries," in *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data (SIGMOD '01)*, 2001, pp. 58–66.

[30] Y. Zhou, Y. Fang, W. Luo, and Y. Ye, "Influential community search over large heterogeneous information networks," *Proc. VLDB Endow.*, vol. 16, no. 8, pp. 2047–2060, 2023.

[31] M. K.-P. Ng, X. Li, and Y. Ye, "MultiRank: co-ranking for objects and relations in multi-relational data," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*, 2011, pp. 1217–1225.

[32] S. Lee, S. Park, M. Kahng, and S. goo Lee, "PathRank: Ranking nodes on a heterogeneous graph for flexible hybrid recommender systems," *Expert Syst. Appl.*, vol. 40, no. 2, pp. 684–697, 2013.

[33] C. Shi, Y. Li, P. S. Yu, and B. Wu, "Constrained-meta-path-based ranking in heterogeneous information network," *Knowl. Inf. Syst.*, vol. 49, no. 2, pp. 719–747, 2016.

[34] Y. Xiong, Y. Zhu, and P. S. Yu, "Top-k similarity join in heterogeneous information networks," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 6, pp. 1710–1723, 2015.

[35] C. Shi, X. Kong, Y. Huang, P. S. Yu, and B. Wu, "HeteSim: A general framework for relevance measure in heterogeneous networks," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 10, pp. 2479–2492, 2014.

[36] C. Wang, Y. Sun, Y. Song, J. Han, Y. Song, L. Wang, and M. Zhang, "RelSim: Relation similarity search in schema-rich heterogeneous information networks," in *Proceedings of the 2016 SIAM International Conference on Data Mining (SDM)*, 2016, pp. 621–629.

[37] E. Cohen, "Size-estimation framework with applications to transitive closure and reachability," *Journal of Computer and System Sciences*, vol. 55, no. 3, pp. 441–453, 1997.

[38] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck, "Sketch-based influence maximization and computation: Scaling up with guarantees," in *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management (CIKM '14)*, 2014, pp. 629–638.

[39] K. Liu, S. Wang, Y. Zhang, and C. Xing, "An efficient algorithm for distance-based structural graph clustering," *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 45:1–45:25, 2023.

[40] F. Zhang and S. Wang, "Effective indexing for dynamic structural graph clustering," *Proceedings of the VLDB Endowment*, vol. 15, no. 11, pp. 2908–2920, 2022.

[41] F. Chierichetti, A. Epasto, R. Kumar, S. Lattanzi, and V. Mirrokni, "Efficient algorithms for public-private social networks," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*, 2015, pp. 139–148.

[42] Y. Tao, C. Sheng, and J. Li, "Finding maximum degrees in hidden bipartite graphs," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD '10)*, 2010, pp. 891–902.
[43] J. Wang, E. Lo, and M. L. Yiu, "Identifying the most connected vertices in hidden bipartite graphs using group testing," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2245–2256, 2013.
[44] C. Sheng, Y. Tao, and J. Li, "Exact and approximate algorithms for the most connected vertex problem," *ACM Trans. Database Syst.*, vol. 37, no. 2, pp. 12:1–12:39, 2012.
[45] W. Cao, J. Li, Y. Tao, and Z. Li, "On top-k selection in multi-armed bandits and hidden bipartite graphs," *Advances in Neural Information Processing Systems*, vol. 28, pp. 1036–1044, 2015.
[46] P. Strouthopoulos and A. N. Papadopoulos, "Core discovery in hidden networks," *Data Knowl. Eng.*, vol. 120, pp. 45–59, 2019.
[47] H. Q. Ngo, E. Porat, C. Ré, and A. Rudra, "Worst-case optimal join algorithms," *J. ACM*, vol. 65, no. 3, pp. 16:1–16:40, 2018.
[48] M. J. Freitag, M. Bandle, T. Schmidt, A. Kemper, and T. Neumann, "Adopting worst-case optimal joins in relational database systems," *Proc. VLDB Endow.*, vol. 13, no. 11, pp. 1891–1904, 2020.
[49] D. Arroyuelo, A. Hogan, G. Navarro, J. L. Reutter, J. Rojas-Ledesma, and A. Soto, "Worst-case optimal graph joins in almost no space," in *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21)*, 2021, pp. 102–114.
[50] T. L. Veldhuizen, "Triejoin: A simple, worst-case optimal join algorithm," in *Proceedings of the 17th International Conference on Database Theory (ICDT)*, 2014, pp. 96–106.
[51] K. S. Beyer, P. J. Haas, B. Reinwald, Y. Sismanis, and R. Gemulla, "On synopses for distinct-value estimation under multiset operations," in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD '07)*, 2007, pp. 199–210.
[52] J.-Y. Audibert, R. Munos, and C. Szepesvári, "Tuning bandit algorithms in stochastic environments," in *Algorithmic Learning Theory - 18th International Conference, ALT 2007, Sendai, Japan, October 1-4, 2007, Proceedings*, 2007, pp. 150–165.
[53] J.-L. Guillaume and M. Latapy, "Bipartite structure of all complex networks," *Inf. Process. Lett.*, vol. 90, no. 5, pp. 215–221, 2004.
[54] ——, "Bipartite graphs as models of complex networks," *Phys. A: Stat. Mech. Appl.*, vol. 371, no. 2, pp. 795–813, 2006.
[55] Y. Niu, Y. Li, P. Karras, and Y. Wang, "Technical report," https://github.com/YudongNiu/HUB/blob/main/Hub_TR.pdf, 2024.
[56] C. Meilicke, M. W. Chekol, D. Ruffinelli, and H. Stuckenschmidt, "Anytime bottom-up rule learning for knowledge graph completion," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*, 2019, pp. 3137–3143.
[57] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "A data-based approach to social influence maximization," *Proc. VLDB Endow.*, vol. 5, no. 1, pp. 73–84, 2011.
[58] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)*, 2010, pp. 1029–1038.
[59] Y. Tang, Y. Shi, and X. Xiao, "Influence maximization in near-linear time: A martingale approach," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15)*, 2015, pp. 1539–1554.