Hub Discovery in Non-Materialized Relational Graphs

Anonymous Author(s)

ABSTRACT

Relational graphs encapsulate nontrivial interactions among entities in heterogeneous data sources. Identifying *hubs* in such graphs is vital in applications such as protein complex discovery and social influence analysis. However, constructing relational graphs from heterogeneous data sources incurs a substantial cost, impeding hub discovery. In this paper, we propose a novel *sketch propagation* framework that approximately identifies hubs in an induced relational graph *without* explicit materialization. Our framework provides provable guarantees for hub discovery under the notion of ϵ-separable sets. It specifically supports hub identification based on *degree* and *h-index* centrality measures. For both these measures, we devise pruning techniques for the efficient processing of *personalized* hub queries that ask whether a node q is a hub. Extensive experimentation confirms the efficacy and efficiency of our proposals, which achieve orders-of-magnitude speedups over exact methods while consistently attaining accuracy beyond 90%. Further, a case study reveals that the hubs our methods identify serve as effective seeds for influence analysis.

1 INTRODUCTION

A *hub query* identifies important nodes within a network according to a *centrality measure* [13, 31]. Such queries find real-world application in network robustness evaluation [34, 38, 49, 55] and information propagation control [12, 14, 33]. Existing works on hub query evaluation assume the data graph is readily available, trivializing the efficient processing of hub queries based on relatively simple centrality measures, such as degree amnd h-index; such works thus focus on complex centrality measures, such as closeness [7, 37] and betweenness [39] centrality. Still, in many real-world scenarios, the graph in question is a *relational* graph *induced* by a *meta-path* [17, 47, 59] that captures complex relationships among heterogeneous entities [1, 26, 27, 46]; such scenarios render the availability assumption false and thus complicate the evaluation of hub queries; they include the following:

- **Protein Complex Discovery.** Protein-protein interaction (PPI) networks ensconced in the Protein Data Bank[1] capture collective interactions among proteins in physiological processes via meta-paths like "$(protein) \longrightarrow (process) \longleftarrow (protein)$"; hubs in a PPI network are vital for the discovery of protein complexes [44, 54].
- **Fraud Detection.** Financial payment records capture connections between transactions sharing the same billing accounts via meta-paths like "$(transaction) \longrightarrow (billing) \longleftarrow (transaction)$"; hubs in such networks reveal potential fraudulent activities [8].
- **Influence Analysis.** Academic KGs record publications and induce collaboration networks via co-authorship meta-paths like "$(author) \longrightarrow (paper) \longleftarrow (author)$" [33]; hubs in such a networks denote influential researchers. Restricted relationships, such as [30, 42] like "$(author) \longrightarrow (paper, venue=\text{'DB'}) \longleftarrow (author)$", reveal influential researchers in specific areas.

[1]https://www.rcsb.org/

In this paper, we study the problem of identifying hubs under the centrality measures of *degree* [13] and *h-index* [31] in a relational graph induced by a meta-path $\mathcal{M}$; we call this core problem HUB.

Challenges. A naïve solution to the HUB problem is to enumerate all instances of $\mathcal{M}$ to construct a relational graph H, compute the degree or h-index of each node in H, and return the nodes with the highest degrees or h-indexes as hubs. However, such explicit relational graph materialization is often time- and memory-intensive, especially when the data sources are of huge volume [57]. An alternative method is to traverse the meta-path instances and maintain each node's neighbors in the relational graph without explicit materialization. Nevertheless, to find the exact hubs, we should access the neighborhood information of all nodes and potentially traverse any edge involved in each meta-path instance up to $O(|V_{\mathcal{M}}|)$ times, where $|V_{\mathcal{M}}|$ is the number of nodes in the relational graph. Thus, this approach can also falter when the relational graph is large. Furthermore, as the number of meta-paths extracted from a data source with heterogeneous entity and relationship types using existing methods can be huge (up to several thousand in our experiments), it is inefficient to identify hubs exactly for each relational graph.

Our Contributions. We find that, in many real-world scenarios, exactly enumerating all hubs is unnecessary, as it suffices to find an approximate set of hubs accurately and efficiently. Inspired by cardinality sketches [2, 15, 16, 19, 25], we propose a novel *sketch propagation* framework for approximate HUB queries. We construct a *neighborhood cardinality* sketch for each node to estimate its degree in the relational graph by iteratively propagating sketches along edges in matching instances of the given meta-path $\mathcal{M}$. Thereby, we collectively estimate the degrees of all nodes in the relational graph through a *single* propagation process that markedly enhances the efficiency of HUB queries under degree centrality.

We extend this sketch propagation framework to approximate HUB queries with the h-index centrality measure [31]. A key challenge is that the h-index of a node depends, apart from its own degree, on the degrees of its neighbors, while cardinality sketches provide information only on a node's own degree. To overcome this hurdle, we propose a *pivot-based* algorithm built on the sketch propagation framework. Instead of directly estimating h-indexes, this algorithm recursively detects nodes whose h-indexes match or exceed that of a randomly chosen pivot node using sketches, in quicksort-like fashion. This method effectively bypasses nodes that cannot be hubs to promptly resolve h-index-based HUB queries.

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

Under the notation of ϵ-separable sets, we prove that our sketch propagation framework provides *unbiased and efficient* approximations for HUB queries based on *both* measures. Our experiments reveal that the estimations can 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 querying hubs (HUB) by means of the degree and h-index centrality measures over relational graphs induced by meta-paths (§ 2).
- We propose a *sketch propagation* framework for approximate degree-based HUB queries and extend it to personalized HUB queries with early termination (§ 3).
- We further devise a *pivot-based* algorithm, also with early termination, for approximate (personalized) HUB queries by means of the h-index measure (§ 4).
- We demonstrate that our methods scale well to large graphs on which state-of-the-art exact methods fail to terminate in a reasonable time and achieve speedups of orders of magnitude over exact methods in most cases while yielding accuracy beyond 90%.
- We conduct a case study showing that our methods provide competitive seeding strategies compared to state-of-the-art methods for influence analysis on materialized relational graphs (§ 5).

2 PRELIMINARIES

In this section, we introduce the basic notation, formulate the problem of finding hubs over relational graphs (HUB), review related works, and present an exact algorithm for HUB query processing.

2.1 Notation and Problem Formulation

We model a heterogeneous data source as a knowledge graph (KG). Specifically, 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 $e(u, v) \in \mathcal{E}$, abbreviated as 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 $\mathcal{L}(e)$. Note that our formulation and methodology are independent from the format of the heterogeneous data and can be easily adapted to support other data sources such as RDBMS.

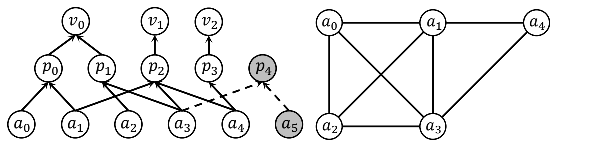
Meta-paths are commonly used to represent relational graphs [17, 58]. We provide the formal definitions of *meta-path* and its *matching instances* on the KG as follows:

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

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

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

When the context is clear, we use $M(v_0, v_1, \dots, v_L)$ or simply M to denote a matching instance and use $M(x_i)$ to denote the node v_i in M. Although our methods support any meta-path $\mathcal{M}$, following the established convention [17, 28, 58], we concatenate $\mathcal{M}$ and its inverse $\mathcal{M}^{-1}$ to induce a homogeneous relational graph $H_{\mathcal{M}}$.



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

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

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

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

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

<u>Problem Statement.</u> We study the problem of finding hubs in relational graphs using the notations in Table 1. Hubs are nodes ranking high according to a function $c : V \longrightarrow \mathbb{R}_{\geq 0}$ that measures importance. We use degree centrality [13], i.e., the number of nodes connected to a node, and h-index [31], i.e., the largest integer h such that a node has h neighbors each of degree at least h, as importance 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 query 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. Thus, we consider how to answer HUB queries approximately and efficiently using randomized algorithms under the notion of ϵ-separable sets [21].

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

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 $q \in S^-_\epsilon(v^\lambda)$.*

Table 1: Frequently used notations.

Notations	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L})$	The knowledge graph.
$\mathcal{M}, \mathcal{M}^{-1}$	The meta-path and its inverse.
$M \triangleright \mathcal{M}$	An instance of meta-path $\mathcal{M}$.
$\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$	The matching graph.
$c(u), \tilde{c}(u)$	centrality of u and its estimation
$H = (V, E)$	The meta-path graph induced by given meta-path
$\mathcal{N}(v)$	The set of neighbors of v in G_P
Λ	The max degree in H.
$\mathcal{K}(C)$	The KMV sketch of collection C
$\mathcal{I}_f(u), \mathcal{I}_b(u)$	The forward/ backward image of u
$I_f(u), I_b(u)$	The size of the forward/backward image of u
v^λ	node ranked at λ percentile
S	The set of result nodes returned.
$S^+_\epsilon(u), S^-_\epsilon(u)$	The set of nodes $\{v \in V \mid c(v) \geq (1+\epsilon)c(u)\}$ and $\{v \in V \mid c(v) \leq (1-\epsilon)c(u)\}$ respectively.

2.2 Related Work

Relational Graph Analysis. Extensive studies have been done on materializing and analyzing relational graphs induced by meta-paths from heterogeneous data sources. Chatzopoulos et al. [10] accelerates the enumeration of instances of different meta-paths by sharing workloads across meta-paths. This method can be extended to relational graph materialization. Guo et al. [24] proposed the BoolAP algorithm for relational graph materialization through boolean matrix multiplication. They also proposed an optimized BoolAP$^+$ 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 (§ 5.2).

Then, great effort [17, 28, 58, 60] has been devoted to community search problems over relational graphs. Fang et al. [17] and Yang et al. [58] proposed $(k, \mathcal{M})$-core and $(k, \mathcal{M})$-truss models for community search in relational graphs induced by meta-paths. Jiang et al. [28] and Zhou et al. [60] 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. Given that the HUB problem is already computationally challenging for non-disjoint meta-paths, we leave the incorporation of disjoint meta-paths for future work.

Xirogiannopoulos and Deshpande [57] proposed to construct compact representations of relational graphs induced by *path joins* in databases for memory-efficient query processing. Although this work focuses on a different problem from ours, our sketch propagation framework can serve as the compact structure in [57]. In addition, the exact algorithm in § 2.3 that leverages the worst-case optimal join for neighborhood computation aligns with the future direction for efficiency improvements suggested in [57].

Meta-paths were also used to rank entities based on centrality measures [29, 35, 43] and assess the similarity between entities [41, 47, 52, 56] within heterogeneous data sources. These methods often operate directly on the original heterogeneous data and do not involve the materialization of relational graphs.

In general, although relational graphs have been widely used in different applications, the fundamental problem of hub discovery 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 sources.

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. [50] proposed two instance-optimal exact algorithms over two kinds of probing strategies of hidden networks. Wang et al. [53] extended this problem to include group testing, whereby probes can verify edge connections between multiple nodes. Sheng et al. [40] proposed a sampling algorithm for hub queries, upon which Cao et al. [9] subsequently improved. Strouthopoulos and Papadopoulos [45] 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 when handling 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.

2.3 Memory-Efficient Exact Method for HUB

Since the materialization of large relational graphs can often be memory-prohibitive [57], 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 [3, 18, 36, 51] on the *matching graph* of $\mathcal{M}$ over $\mathcal{G}$ and avoid the materialization of relational graphs. We first introduce the notion of *matching graph* of a meta-path and the successors and predecessors of a node in the matching graph as follows:

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

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

Example 2.8. In Figure 1(a), unshaded nodes and solid edges denote the matching graph $\mathcal{G}^*$ of meta-path $\mathcal{M}(A, \text{'write'}, P, \text{'publish'}, V)$. Nodes a_5 and p_4 are not in $\mathcal{G}^*$ since they are not included in any instance of $\mathcal{M}$. The node set $\mathcal{V}_0 = \{a_0, a_1, \ldots, a_4\}$ consists of 'authors' contained in a matching instance of $\mathcal{M}$. Similarly, we

```
Procedure DFS($u, v, \circ$)
Input: The matching graph $\mathcal{G}^*$ and the direction $\circ \in \{+, -\}$
1 Mark $(v, \circ)$ as visited;
2 if $v \in \mathcal{V}_L$ and $\circ = +$ then DFS$(u, v, -)$;
3 if $v \in \mathcal{V}_0$ and $\circ = -$ then Add $v$ to $\mathcal{N}(u)$;
4 for $w \in \mathcal{V}^{\circ}(v)$ do
5     if $(w, \circ)$ is not visited then DFS$(u, w, \circ)$;
```

have $\mathcal{V}_1 = \{p_0, p_1, p_2, p_3\}$ and $\mathcal{V}_2 = \{v_0, v_1, v_2\}$. The set of predecessors of p_2 is $\mathcal{V}^-(p_2) = \{a_1, a_3, a_4\}$, which contains in-neighbors of p_2 in $\mathcal{G}^*$ along edge type 'write.' The set of successors of p_2 is $\mathcal{V}^+(p_2) = \{v_1\}$, which contains out-neighbors of p_2 in $\mathcal{G}^*$ along edge type 'publish.'

The matching graph $\mathcal{G}^*$ can be extracted from $\mathcal{G}$ efficiently by *forward* and *backward* breadth-first search (BFS). At each level $i \in [0, \dots, L-1]$, the forward BFS iteratively visits all neighbors of each candidate matching node of x_i via edge type y_i as candidates for x_{i+1}. Subsequently, the backward BFS visits all the neighbors of the candidates for x_{i+1} via edge type y_i^{-1} as the candidates for x_i at each level $i \in [L-1, \dots, 0]$. Only the candidate nodes and their adjacent edges visited by both forward and backward BFS are identified as the matching nodes and edges. Based on our experimental findings (see Table 3), 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 $\mathcal{M}$ and $\mathcal{M}^{-1}$. Specifically, DFS$(u, u, +)$ (Procedure DFS) runs for each node $u \in V$ and recursively traverses the current node's successors (Lines 4–5). Once we visit a node in $\mathcal{V}_L$, the direction of DFS switches from traversing instances of $\mathcal{M}$ to those of $\mathcal{M}^{-1}$ (Line 2). If the traversal reaches $\mathcal{V}_0$ via any instance of $\mathcal{M}^{-1}$, we add node v to $\mathcal{N}(u)$ (Line 3). After performing DFS for each node, we discard the set $\mathcal{N}(u)$ and keep only the degree $|\mathcal{N}(u)|$ to ensure a small memory footprint. To compute all nodes' h-indexes 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 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.

3 SKETCH PROPAGATION FRAMEWORK

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

3.1 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 [6] and defined as follows.

Definition 3.1 (KMV Sketch). Given a collection $C = \{o_0, o_1, \dots, o_{|C|}\}$ 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 $|\widetilde{C}| = \mathbb{E}[(k-1)/\zeta]$, 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 [4], an estimation of the collection size with probability bounds can be obtained as follows.

Lemma 3.2. *For any $\delta, p \in (0, 1)$, if $\theta = \Theta(\frac{|C|}{\delta k} \log \frac{1}{p})$, then $(1 - \delta) \cdot |C| \le |\widetilde{C}| \le (1 + \delta) \cdot |C|$ holds with probability at least $1 - p$.*

However, it is still challenging to apply the KMV sketches and corresponding estimators to HUB queries. The KMV sketch is designed primarily for stream processing [2, 15, 16, 19, 25], 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 the above 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 3.3 (Images). The forward image of a matching node $u \in \mathcal{V}_i$, denoted as $\mathcal{I}_f(u)$, contains a node $v \in \mathcal{V}_0$ if there exists an instance M of $\mathcal{M}$ including both u and v, i.e.,

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

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

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

Example 3.4. In Figure 1(a), the forward image $\mathcal{I}_f(p_1)$ contains two nodes a_2 and a_3, as they are connected with p_1 via $M(a_2, p_1, v_0)$ and $M(a_3, p_1, v_0)$, respectively. The backward image $\mathcal{I}_b(p_1)$ contains nodes a_0, a_1, a_2 and a_3. For example, node a_0 belongs to $\mathcal{I}_b(p_1)$ as it is connected to p_1 through $M(a_2, p_1, v_0)$ and $M'(v_0, p_0, a_0)$.

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

Forward and Backward Propagation. We build a KMV sketch for $\mathcal{N}(u)$ of each $u \in V$ by iteratively maintaining the sketches for the forward and backward images of nodes from $\mathcal{V}_0$ to $\mathcal{V}_L$.

Algorithm 1: Sketch Propagation

```
Input: KG $\mathcal{G}$, meta-path $\mathcal{M}$, quantile parameter $\lambda$, number
       of propagations $\theta$, KMV sketch size $k$
Output: A set of nodes $S$ for (approximate) HUB query
1  forall $u \in \mathcal{V}^*$ do
2      for $t = 1$ to $\theta$ do
3          $\mathcal{K}_f[u][t] \leftarrow \emptyset$ and $\mathcal{K}_b[u][t] \leftarrow \emptyset$;
4          if $u \in \mathcal{V}_0$ then add $r(u) = \texttt{Rand}(0,1)$ to $\mathcal{K}_f[u][t]$;
5  $\widetilde{N} \leftarrow \texttt{Propagate}(\mathcal{K}_f, \mathcal{K}_b)$;
6  $S \leftarrow$ top-$(\lambda \cdot |V|)$ nodes with highest degrees in $H$ w.r.t. $\widetilde{N}$;
7  return $S$;
8  Function Propagate($\mathcal{K}_f, \mathcal{K}_b$):
9      for $i = 1$ to $L$ do
10         forall $u \in \mathcal{V}_i$ do Receive($u, \mathcal{K}_f, -$);
11     forall $u \in \mathcal{V}_L$ do $\mathcal{K}_b[u] \leftarrow \mathcal{K}_f[u]$;
12     for $i = L-1$ to $0$ do
13         forall $u \in \mathcal{V}_i$ do Receive($u, \mathcal{K}_b, +$);
14     forall $u \in \mathcal{V}_0$ do
15         $\widetilde{\mu} \leftarrow 0$;
16         for $t = 1$ to $\theta$ do
17             if $|\mathcal{K}_b[u][t]| = k$ then
18                 $\widetilde{\mu} \leftarrow \widetilde{\mu} + \frac{\max(\mathcal{K}_b[u][t])}{\theta}$;
19             else
20                 $\widetilde{\mu} \leftarrow \widetilde{\mu} + \frac{k}{(|\mathcal{K}_b[u][t]|+1)\cdot\theta}$;
21         $\widetilde{N}[u] \leftarrow k/\widetilde{\mu} - 1$;
22     return $\widetilde{N}$;
23 Function Receive($u, \mathcal{K}, \circ$):
24     for $t = 1$ to $\theta$ do
25         $\mathcal{K}[u][t] \leftarrow \oplus_{v \in \mathcal{V}^{\circ}(u)} \mathcal{K}[v][t]$;
```

First, a random number $r(u)$ is generated for each node $u \in \mathcal{V}_0$. Since $\mathcal{I}_f(u) = \{u\}$, the sketch is initialized as $\mathcal{K}(\mathcal{I}_f(u)) = \{r(u)\}$. The sketches for the forward images of 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. After building the sketches for the forward images of the nodes in $\mathcal{V}_L$ (which is equal to the sketches w.r.t. their backward images), the algorithm propagates the sketches back from the nodes in $\mathcal{V}_L$ to the nodes in $\mathcal{V}_0$ in the same way as for forward propagation so as to build the sketches for the backward images of the nodes in $\mathcal{V}_0$ to estimate the degree of each node in H. The following lemma ensures the correctness of the *sketch propagation* framework.

LEMMA 3.5. *Given a meta-path $\mathcal{M}$, for each node $u \in \mathcal{V}_i$*

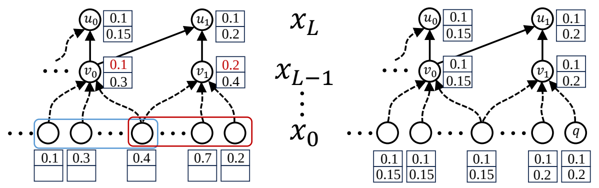
$$\mathcal{K}(\mathcal{I}_f(u)) = \oplus_{v \in \mathcal{V}^-(u)} \mathcal{K}(\mathcal{I}_f(v)) \quad \textit{and} \tag{1}$$
$$\mathcal{K}(\mathcal{I}_b(u)) = \oplus_{v \in \mathcal{V}^+(u)} \mathcal{K}(\mathcal{I}_b(v)), \tag{2}$$

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 [6, 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 $\mathcal{I}_f(u) = \bigcup_{v \in \mathcal{V}^-(u)} \mathcal{I}_f(v)$ and $\mathcal{I}_b(u) = \bigcup_{v \in \mathcal{V}^+} \mathcal{I}_b(v)$ for each node $u \in \mathcal{V}_i$.

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

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



(a) Forward propagation (b) Backward propagation

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

Example 3.6. Figure 2 illustrates the sketch propagation framework with $k = 2$ and $\theta = 1$. The forward image $\mathcal{I}_f(u_1)$ is the union of the forward images of the nodes in its predecessors $\mathcal{V}^-(u_1) = \{v_0, v_1\}$, that is, the union of nodes in the blue and red boxes in the layer x_0. The sketch $\mathcal{K}(\mathcal{I}_f(u_1))$ is constructed by taking the two smallest numbers from the union of sketches of the forward images of nodes in $\mathcal{V}^-(u_1)$. Thus, $\mathcal{K}(\mathcal{I}_f(u_1)) = \{0.1, 0.3\} \oplus \{0.2, 0.4\} = \{0.1, 0.2\}$. During the backward propagation, the sketch $\mathcal{K}(\mathcal{I}_b(v_0))$ is constructed similarly but in the reverse direction by taking the smallest two numbers from the union of sketches of backward images of nodes in $\mathcal{V}^+(v_0)$. Thus, $\mathcal{K}(\mathcal{I}_b(v_0)) = \{0.1, 0.15\} \oplus \{0.1, 0.2\} = \{0.1, 0.15\}$.

Theoretical Analysis. We prove that *sketch propagation* approximates HUB queries with high probability (w.h.p.) (cf. Theorem 3.7). Hereafter, we use $I_f(u)$ and $I_b(u)$ to denote the sizes of $\mathcal{I}_f(u)$ and $\mathcal{I}_b(u)$ for any node $u \in \mathcal{V}^*$. When the context is clear, we abbreviate $I_b(u)$ and $I_f(u)$ as I. We also use $\widetilde{I}$ to denote the estimation of I provided by sketch propagation.

THEOREM 3.7. *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 through sketch propagation. Given any nodes $u \in S_0^+(v^\lambda)$ and $v \in S_\epsilon^-(v^\lambda)$,

$$\begin{aligned}\widetilde{N}(v) &\le (1+\delta) \cdot N(v) \le (1+\delta)(1-\epsilon) \cdot N(v^\lambda) \\ &\le (1+\delta)(1-\epsilon) \cdot N(u) \le \frac{(1+\delta)(1-\epsilon)}{1-\delta} \cdot \widetilde{N}(u)\end{aligned}$$

hold by applying Lemma 3.2 over $\mathcal{K}(\mathcal{I}_b(v))$ 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 $\widetilde{N}(v) \ge (1-\delta) \cdot N(v) \ge (1-\delta)(1+\epsilon) \cdot N(v^\lambda) \ge (1-\delta)(1+\epsilon) \cdot N(u) \ge \frac{(1-\delta)(1+\epsilon)}{1+\delta} \cdot \widetilde{N}(u)$. 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 $1-p$. □

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

3.2 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 3.7 for personalized queries.

Algorithm 2: `OptimizedReceive`

```
   Input: Node $u$, arrays of KMV sketches $\mathcal{K}_f, \mathcal{K}_b$
   Output: If sketch propagation can be terminated
1  Receive($u, \mathcal{K}_b$, +);
2  $\widetilde{I}_f \leftarrow 0$ and $\widetilde{I}_b \leftarrow 0$;
3  for $t = 1$ to $\theta$ do
4      if $|\mathcal{K}_f[u][t]| = k$ then $\widetilde{I}_f \leftarrow \widetilde{I}_f + \frac{\max(\mathcal{K}_f[u][t])}{\theta}$;
5      else $\widetilde{I}_f \leftarrow \widetilde{I}_f + \frac{k}{(|\mathcal{K}_f[u][t]|+1)\cdot\theta}$;
6      if $|\mathcal{K}_b[u][t]| = k$ then $\widetilde{I}_b \leftarrow \widetilde{I}_b + \frac{\max(\mathcal{K}_b[u][t])}{\theta}$;
7      else $\widetilde{I}_b \leftarrow \widetilde{I}_b + \frac{k}{(|\mathcal{K}_b[u][t]|+1)\cdot\theta}$;
8  $\widetilde{I}_f \leftarrow k/\widetilde{I}_f - 1$ and $\widetilde{I}_b \leftarrow k/\widetilde{I}_b - 1$;
9  if $\widetilde{I}_f \ge \lambda|V|$ and $\widetilde{I}_b \ge N(q)$ then return TRUE;
10 return FALSE;
```

COROLLARY 3.8. *When $\theta = \Theta(\frac{\Lambda}{\epsilon k} \log \frac{|V|}{p})$, the sketch propagation framework 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 *sketch propagation* early once we have determined that q is not a hub with high confidence. First, the following lemma inspires the design of early termination optimization.

LEMMA 3.9. *Given a query node q, if there exists a node $u \in \mathcal{V}^*$ such that $I_f(u) \ge \lambda|V|$ and $I_b(u) > N(q)$, then the answer to the personalized HUB query is FALSE.*

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

Based on Lemma 3.9, we can terminate *sketch propagation* when the estimations $\widetilde{I}_f(u)$ and $\widetilde{I}_b(u)$ of $I_f(u)$ and $I_b(u)$ for a node $u \in \mathcal{V}^*$ by KMV sketches satisfy $\widetilde{I}_f(u) \ge \lambda|V|$ and $\widetilde{I}_b(u) > N(q)$. Algorithm 2 presents the procedure of `OptimizedReceive`, an optimized version of `Receive` at Line 23 in Algorithm 1 with early termination. Specifically, `OptimizedReceive` returns whether the sketch propagation should end at node u. It calls `Receive` in Line 23 of Algorithm 1 to construct the KMV sketch for the backward image of u (Line 1) and estimates $I_f(u)$ and $I_b(u)$ accordingly (Lines 2–8). Finally, it returns whether the estimated $\widetilde{I}_f(u)$ and $\widetilde{I}_b(u)$ have satisfied the early termination conditions (Lines 9–10).

Example 3.10. Continuing with Example 3.6 where $\mathcal{K}(\mathcal{I}_b(v_0)) = \{0.1, 0.15\}$, the algorithm estimates $\widetilde{I}_b(v_0) = \frac{2}{0.15} - 1 \approx 12.3$. The size $\widetilde{I}_f(v_0) = \frac{2}{0.3} - 1 \approx 5.7$ is estimated from the sketch $\mathcal{K}(\mathcal{I}_f(v_0))$ in Figure 2(a). Assuming $\lambda = 0.01$, $|V| = 500$, $N(q) = 9$, we have $\widetilde{I}_f(u_1) > 0.01 \cdot 500$ and $\widetilde{I}_b(u_1) > 9$, and thus the propagation can be terminated early.

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

Theorem 3.11. *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 $\beta = \min(\frac{\widetilde{I}_f(u)}{\lambda|V|}, \frac{\widetilde{I}_b(u)}{N(q)})$.*

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

$$O(\frac{\Lambda}{\epsilon k}\log\frac{|V|}{p}) = O(\frac{\Lambda}{\delta k}\log\frac{1}{p^{\delta/\epsilon}}) \tag{3}$$

Combining Lemma 3.2 and Eqn. 3, if $\theta = O(\frac{\Lambda}{\epsilon k}\log\frac{|V|}{p})$, we have

$$\widetilde{I}_f(u) \le (1+\delta)\cdot I_f(u);\ \widetilde{I}_b(u) \le (1+\delta)\cdot I_b(u). \tag{4}$$

Each holds with probability at least $1 - p^{\delta/\epsilon}$. According to Eqn. 4, for any $\delta < \beta - 1$, we have

$$I_f(u) \ge \frac{\widetilde{I}_f(u)}{1+\delta} \ge \frac{\beta\lambda|V|}{1+\delta} > \lambda|V| \tag{5}$$

and

$$I_b(u) \ge \frac{\widetilde{I}_b(u)}{1+\delta} \ge \frac{\beta N(q)}{1+\delta} > N(q) \tag{6}$$

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 $\widetilde{I}_f(u) \ge \lambda|V|$ and $\widetilde{I}_b(u) > N(q)$, but $I_f(u) < \lambda|V|$ or $I_b(u) \le N(q)$. Thus, a personalized HUB query with answer TRUE is incorrectly answered due to early termination at u with probability at most $2\cdot p^{(\beta-1)/\epsilon}$. □

4 EXTENSION TO H-INDEX

In this section, we extend the sketch propagation framework to process approximate HUB queries based on the h-index measure. To compute the h-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* [31].

Definition 4.1 (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 $h(u)$.

4.1 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., $Q_u^+ = \{v \in V \mid h(v) \ge h(u)\}$ and $Q_u^- = \{v \in V \mid h(v) < h(u)\}$. As such, we can decide that $Q_u^+ \subseteq S_0^+(v_\lambda)$ if $u \in S_0^+(v_\lambda)$ since all nodes in Q_u^+ have h-index values no less than $h(u)$ and thus are ranked higher than u, or $Q_u^- \cap S_0^+(v_\lambda) = \emptyset$ 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.

Algorithm 3: Pivot-based Algorithm

```
Input: KG $\mathcal{G}$, meta-path $\mathcal{M}$, quantile parameter $\lambda$, number
       of propagations $\theta$, KMV sketch size $k$
Output: The set of nodes $S$ for (approximate) HUB query
1  forall $u \in \mathcal{V}^*$ do
2      for $t = 1$ to $\theta$ do
3          $\mathcal{K}_f[u][t] \leftarrow \emptyset$ and $\mathcal{K}_b[u][t] \leftarrow \emptyset$;
4          if $u \in \mathcal{V}_0$ do add $r(u) = \mathrm{Rand}(0,1)$ to $\mathcal{K}_f[u][t]$;
5  $\widetilde{N} \leftarrow$ Propagate($\mathcal{K}_f, \mathcal{K}_b$);
6  $Q \leftarrow V$ and $S \leftarrow \emptyset$;
7  while $|Q| > 0$ do
8      Sample a random node from $Q$ as the pivot node $u$;
9      Sort the neighbors $\mathcal{N}(u)$ of $u$ in descending order by $\widetilde{N}$;
10     Initialize $\widetilde{h}(u) \leftarrow 0$;
11     forall $v \in \mathcal{N}(u)$ do
12         if $\widetilde{N}(v) \ge \widetilde{h}(u)$ then $\widetilde{h}(u) \leftarrow \widetilde{h}(u) + 1$;
13     forall $v \in \mathcal{V}^*$ do
14         for $t = 1$ to $\theta$ do
15             $\mathcal{K}_f[v][t] \leftarrow \emptyset$ and $\mathcal{K}_b[v][t] \leftarrow \emptyset$;
16             if $v \in \mathcal{V}_0$ and $\widetilde{N}(v) \ge \widetilde{h}(u)$ then
17                 Add $r(v) = \mathrm{Rand}(0,1)$ to $\mathcal{K}_f[v][t]$;
18     $\widetilde{h} \leftarrow$ Propagate($\mathcal{K}_f, \mathcal{K}_b$);
19     Initialize $Q_u^+ \leftarrow \emptyset$ and $Q_u^- \leftarrow \emptyset$;
20     forall $v \in Q$ do
21         if $\widetilde{h}(v) \ge \widetilde{h}(u)$ then add $v$ to $Q_u^+$ else add $v$ to $Q_u^-$;
22     if $|Q_u^+| + |S| \le \lambda|V|$ then
23         $Q \leftarrow Q_u^-$ and $S \leftarrow S \cup Q_u^+$;
24         if $|S| \le \lambda|V|$ then $S \leftarrow S \cup \{u\}$;
25     else
26         $Q \leftarrow Q_u^+$;
27 return $S$;
```

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 two nodes $u, v \in V$, we have $h(v) \ge h(u)$ if and only if v has at least $h(u)$ neighbors in H with degrees larger than or equal to $h(u)$. Based on the above observation, we introduce a two-stage pivotal partitioning method for our pivot-based algorithm.

Stage (I): 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.

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 arrays $\mathcal{K}_f$ and $\mathcal{K}_b$ of KMV sketches to estimate the degree of each node in H (Lines 1–5). Note that the estimated degrees will be reused across all iterations for partitioning. The iterative process proceeds until Q, which contains the nodes to be partitioned in each iteration, is empty (Lines 7–26). In each iteration, it first randomly chooses a pivot node u from Q and estimates its h-index $\widetilde{h}(u)$ (Lines 8-12). Then, it re-initializes arrays $\mathcal{K}_f$ and $\mathcal{K}_b$ of KMV sketches for the nodes with degrees greater than $\widetilde{h}(u)$ (Lines 13–17). After that, the `Propagate` function is performed over $\mathcal{K}_f$ and $\mathcal{K}_b$ to estimate the number of neighbors with degrees greater than $\widetilde{h}(u)$ for each node in Q (Line 18). Accordingly, it partitions Q into Q_u^+ and Q_u^- according to $\widetilde{h}$ (Lines 19–21) and updates the result set S and the candidate node set Q based on Q_u^+ and Q_u^- (Lines 22–26). Finally, when Q is empty, the result set S is returned (Line 27). The ties are broken arbitrarily.

Theoretical Analysis. Algorithm 3 returns the correct answer for h-index 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 4.2. *For any $\delta, p \in (0,1)$, if $\theta = \Theta(\frac{\Lambda}{\delta k}\log\frac{|V|}{p})$, we have $(1-\delta)\cdot h(u) \le \widetilde{h}(u) \le (1+\delta)\cdot h(u)$ 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 $N(v_i) \ge h(u)$ for $0 \le i \le h(u)-1$ and $N(v_i) \le h(u)$ for $h(u) \le i \le N(u)-1$. By combining Lemma 3.2 with the union bound over V, when $\theta = \Theta(\frac{\Lambda}{\delta k}\log\frac{|V|}{p})$ in the first stage, we have $\widetilde{N}(v_i) \ge (1-\delta)N(v_i) \ge (1-\delta)h(u)$ for $0 \le i \le h(u)-1$ with probability at least $1-p$. Thus, node u has $h(u) \ge (1-\delta)h(u)$ neighbors with estimated degrees of at least $(1-\delta)h(u)$. Thus, $\widetilde{h}(u) \ge (1-\delta)h(u)$ with probability at least $1-p$. Similarly, we have $\widetilde{N}(v_i) \le (1+\delta)N(v_i) \le (1+\delta)h(u)$ for $i \ge h(u)$ 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 $\widetilde{h}(u) \le (1+\delta)h(u)$ with probability at least $1-p$. □

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

Lemma 4.3. *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 4.4. *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 4.3, 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 $O(\frac{\Lambda|\mathcal{E}^*|}{\epsilon}\log^2\frac{|V|}{p})$ since it invokes *sketch propagation* $O(\log|V|)$ times in amortization. The space complexity of Algorithm 3 is $O(k\theta|\mathcal{V}^*| + |\mathcal{E}^*|)$, which is equal to that of Algorithm 1.

4.2 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 4.3, the following corollary indicates that the above algorithm provides correct answers for approximate personalized HUB queries w.h.p.

Corollary 4.5. *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 h-index with probability at least $1-p$.*

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

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

Proof. Each pair of nodes in $\mathcal{I}_b(u)$ and $\mathcal{I}_f(u)$ are neighbors of each other in H according to the proof of Lemma 3.9. Thus, each node $v_1 \in \mathcal{I}_b(u)$ has at least $I_f(u)$ neighbors with degrees larger than or equal to $I_b(u)$. Since $I_f(u) > h(q)$ and $I_b(u) \geq \max(h(q), \lambda \cdot |V|)$, there exist at least $I_b(u) \geq \lambda|V|$ 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 4.6, 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 $\widetilde{I}_f(u) > \widetilde{h}(q)$ and $\widetilde{I}_b(u) \geq \max(\widetilde{h}(q), \lambda|V|)$.

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 4.7. *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 $\beta = \min(\frac{\widetilde{I}_f(u)}{\widetilde{h}(q)}, \frac{\widetilde{I}_b(u)}{\lambda|V|})$.*

Proof. By Lemma 4.2 and Eqn. 3, for any $\delta < (1-\epsilon)\beta - 1$,

$$I_f(u) \geq \frac{\widetilde{I}_f(u)}{1+\delta} \geq \frac{\beta\widetilde{h}(q)}{1+\delta} \geq \frac{\beta(1-\epsilon)h(q)}{1+\delta} > h(q) \qquad (7)$$

and

$$\begin{aligned} I_b(u) &\geq \frac{\widetilde{I}_b(u)}{1+\delta} \geq \frac{\beta \cdot \max(\widetilde{h}(q), \lambda|V|)}{1+\delta} \\ &\geq \frac{\beta \cdot \max((1-\epsilon)h(q), \lambda|V|)}{1+\delta} > \max(h(q), \lambda|V|) \end{aligned} \qquad (8)$$

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 $\widetilde{I}_f(u) \geq h(q)$ and $\widetilde{I}_b(u) > \max(h(q), \lambda|V|)$, but $I_f(u) < h(q)$ or $I_b(u) \leq \max(h(q), \lambda|V|)$. Thus, a personalized HUB query with answer TRUE is incorrectly answered due to early termination at u with probability at most $p + 2p^{((1-\epsilon)\beta-1)/\epsilon}$. □

Remark. The values of θ in Theorems 3.7-4.7 reflect the worst-case scenarios. In our experiments, we observe that sketch-based algorithms achieve accurate results with a much smaller θ.

Table 2: 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

5 EXPERIMENTS

5.1 Experimental Setup

Datasets. The statistics of the datasets are reported in Table 2. 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[2]. Yelp is a KG built from user ratings and comments on businesses at different locations. BPM represents a series of large synthetic datasets generated by the Bipartite Preferential Model [22, 23, 57], which are used for scalability tests (§ 5.5). Please refer to [5] for the generation of BPM.

Query Generation. For each dataset, we use AnyBURL [32] 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 2 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 sketch propagation-based algorithms with exact algorithms based on the generic worst-case optimal join and boolean matrix multiplication. The compared methods are listed as follows:

- ExactD and ExactH are the exact algorithms (§ 2.3) to get hubs by computing the degrees and h-indexes of all nodes in H.
- BoolAP [24] returns exact hubs based on both degree and h-index by materializing the relational graph H 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 (§ 3.1) 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 (§ 3.2).
- GloH and PerH apply the pivot-based algorithm (§ 4.1) to find the hubs based on estimated h-indexes. GloH recursively partitions the nodes until all hubs are found, whereas PerH uses a query node q as the pivot to check whether it is a hub.
- PerH$^+$ optimizes PerH with early termination (§ 4.2).

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

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 § 5.3 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, PerD$^+$), 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.

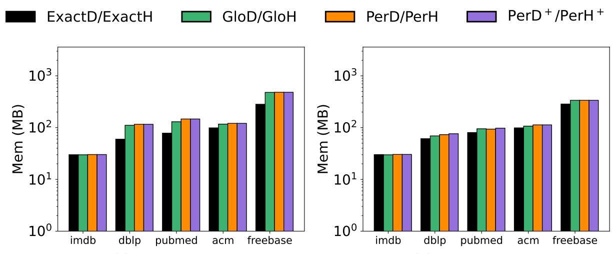
Evaluation Metrics. We evaluate the quality of the hubs each algorithm returns for global HUB queries (GloD and GloH) with the F1-score. Our algorithms return a node set S containing $\lambda \cdot |V|$ nodes. However, there might be more than $\lambda \cdot |V|$ nodes satisfying the criteria of Problem 1 due to the tied values with v^λ. Thus, we exclude nodes with centralities equal to v^λ in the recall calculation. To assess the effectiveness of PerD, PerD$^+$, PerH, and PerH$^+$ for personalized HUB queries, we record their accuracy in determining whether a query node q has equal or greater centrality compared to v^λ. We report the average F1 and accuracy scores across all meta-paths in Figures 5 and 6. For efficiency evaluation, we report the average running time of each algorithm to process a HUB query.

Environment. All our 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. The source code and data are published anonymously.[3]

5.2 Efficiency Evaluation

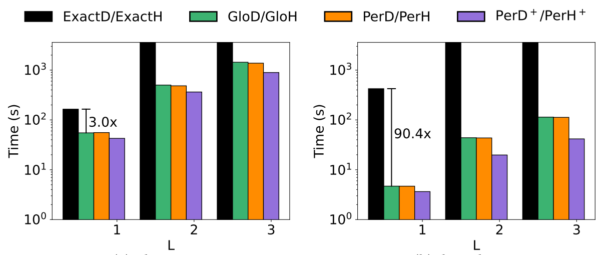
In Table 3, we present the execution time per query for each method, along with its speedup ratios compared to the fastest exact method across five datasets. Based on these results, we make the following observations. First, sketch propagation-based algorithms are consistently more efficient than exact baselines ExactD and ExactH across all datasets except IMDB and provide more than 10× speedups for degree-based global queries (on FreeBase, GloD is 29× faster than ExactD) and up to two orders of magnitude speedups for h-index-based global queries (on FreeBase, GloH is 191× faster than ExactH). The speedup over degree-based global queries is smaller than h-index-based queries because h-index-based queries are accurately approximated with smaller k and θ as discussed in § 5.3. GloD is slightly slower than ExactD for degree-based queries on IMDB since IMDB is very small, where the worst-case optimal join is performed quickly over the matching graph, whereas our methods require additional costs for sketch construction and degree estimation. Nevertheless, GloD answers HUB queries on IMDB in 1.3ms. BoolAP$^+$ is exceptionally fast on ACM since it's specially optimized for KGs with dense regions and ACM is significantly denser than other datasets. Nevertheless, BoolAP$^+$ is much slower than our sketch-based methods across other datasets and is less scalable even than ExactD and ExactH. Thus, in the following, we mainly compare our sketch-based methods with ExactD and ExactH. Second, the algorithms for personalized queries, i.e., PerD$^+$

[3]https://anonymous.4open.science/r/HUB-F5CF/readme.md



(a) degree (b) h-index

Figure 3: Memory consumption of compared methods.



(a) degree (b) h-index

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

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

Then, we investigate the impact of the length of the meta-path L on the efficiency of different methods on Yelp. The results are presented in Figure 4. 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 global HUB queries based on degree and h-index in 1,438s and 114s, respectively. For personalized HUB queries, PerH$^+$ and PerD$^+$ handle query meta-paths of $L = 3$ in 893s and 42s.

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

Finally, we report the memory consumption of different methods in Figure 3. We can observe from Figure 3(a) that GloD, PerD, and PerD$^+$ take relatively more memory than the exact algorithm ExactD in order to maintain KMV sketches for images. Nevertheless, our methods take at most 1.93 times 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).

5.3 Effectiveness Evaluation

Results for Degree-based HUB Queries. Figures 5(a)–5(e) illustrate the effectiveness of GloD, PerD, and PerD$^+$ on each dataset by

Table 3: Results for the efficiencies of different algorithms. For exact methods, the running time on each dataset is reported. For GloD, PerD$^+$, GloH, and PerH$^+$, the running time, as well as the speedup ratios over the fastest exact method, are reported.

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

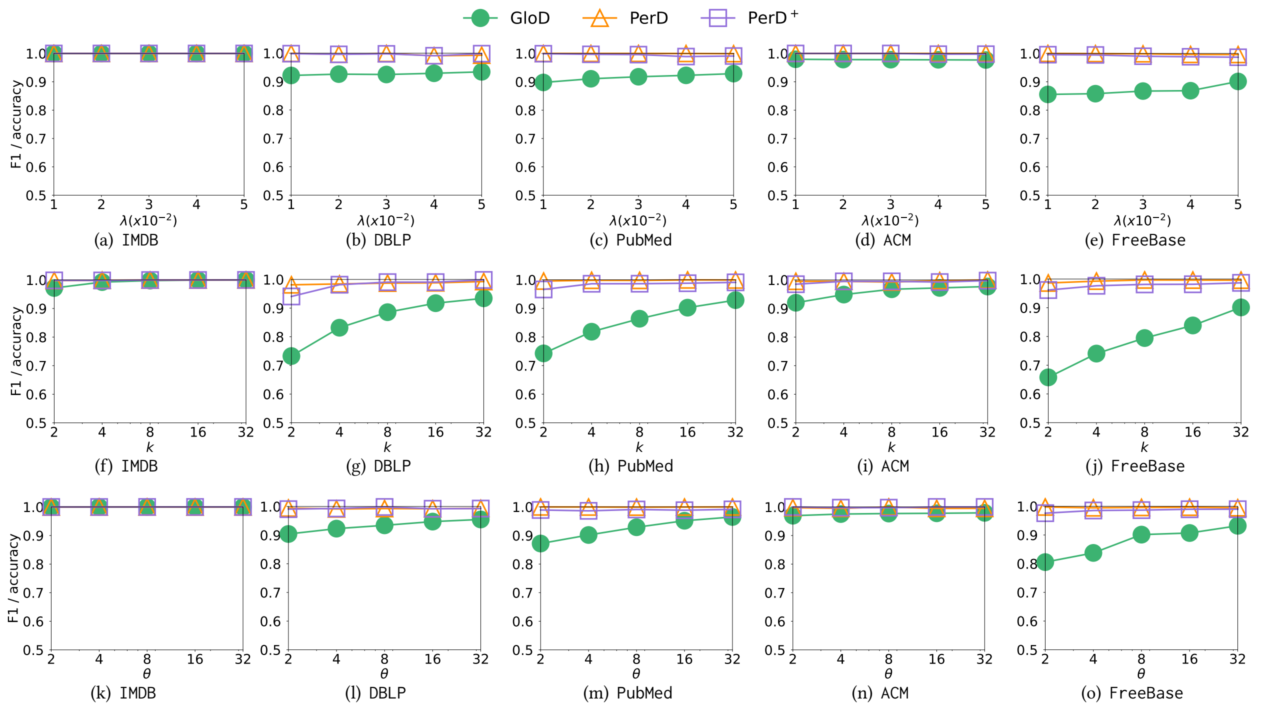


Figure 5: Effectiveness of GloD, PerD, and PerD$^+$ for HUB queries based on degree centrality with varying parameters. Subfigures (a)-(e): varying parameter λ. Subfigures (f)-(j): varying parameter k. Subfigures (k)-(o): varying parameter θ.

varying the parameter λ. We observe that GloD achieves F1-scores of more than 0.85 across all datasets for global HUB queries, and PerD and PerD$^+$ consistently achieve at least 0.98 accuracy for personalized HUB queries. Moreover, GloD provides more accurate answers with increasing λ and achieves 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 the HUB queries apply less strict requirements to the hubs, which are more easily approximated.

Figures 5(f)–5(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 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.

Figures 5(k)–5(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 Figure 6. Similar observations can be made for the methods for h-index-based HUB queries as those for degree-based HUB queries. Furthermore, we note that GloH achieves a higher F1-score for h-index-based HUB queries compared to GloD 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 tied values to v^λ from the recall evaluation, h-index-based HUB queries become much easier to approximate.

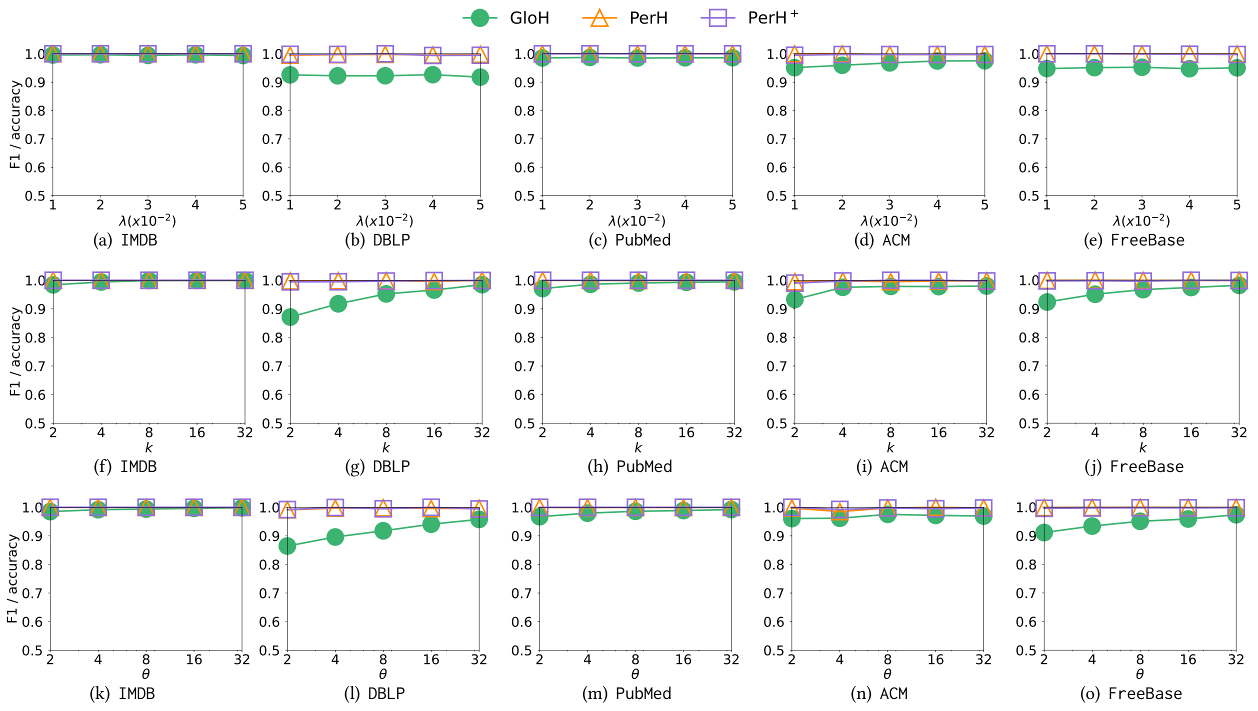


Figure 6: Effectiveness of GloH, PerH, and PerH$^+$ for HUB queries based on h-index with varying parameters. Subfigures (a)-(e): varying parameter λ. Subfigures (f)-(j): varying parameter k. Subfigures (k)-(o): varying parameter θ.

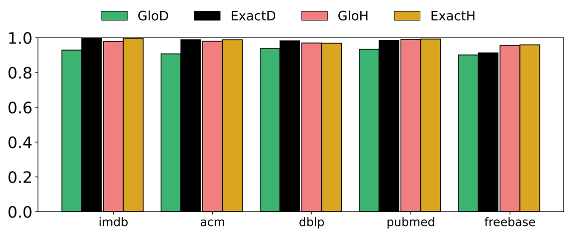


Figure 7: Ratio between the influence scores achieved by hubs revealed by HUB methods and the highest influence scores estimated by the RIS algorithm.

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

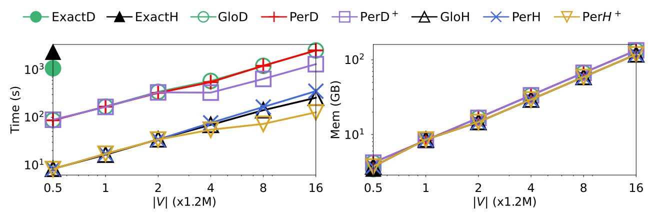
5.4 Influence Analysis

We further verify the effectiveness of HUB for influence analysis on relational graphs. We adopt the weighted cascade model for influence estimation [11, 20]. Figure 7 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 [48]. 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.

5.5 Scalability Test

We further test the scalability of our methods on large synthetic datasets BPM. Figure 8 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 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.

6 CONCLUSION



(a) Running Time (b) Memory Usage

Figure 8: Running time and memory consumption of our methods on the BPM dataset.

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 proposals with extensive experimentation on real-world heterogeneous data.

REFERENCES

[1] Daniel J Abadi, Adam Marcus, Samuel R Madden, and Kate Hollenbach. 2009. SW-Store: a vertically partitioned DBMS for Semantic Web data management. *VLDB J.* 18 (2009), 385–406.

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

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

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

[5] Anonymous Author(s). 2024. Technical Report. https://anonymous.4open.science/r/HUB-F5CF/Hub_TR.pdf.

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

[7] Patrick Bisenius, Elisabetta Bergamin, Eugenio Angriman, and Henning Meyerhenke. 2018. Computing top-k closeness centrality in fully-dynamic graphs. In *2018 Proceedings of the Twentieth Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM, 21–35.

[8] Bokai Cao, Mia Mao, Siim Viidu, and S Yu Philip. 2017. HitFraud: A broad learning approach for collective fraud detection in heterogeneous information networks. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 769–774.

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

[10] Serafeim Chatzopoulos, Thanasis Vergoulis, Dimitrios Skoutas, Theodore Dalamagas, Christos Tryfonopoulos, and Panagiotis Karras. 2023. Atrapos: Real-time Evaluation of Metapath Query Workloads. In *Proceedings of the ACM Web Conference 2023 (WWW '23)*. Association for Computing Machinery, New York, NY, USA, 2487–2498.

[11] Wei Chen, Chi Wang, and Yajun Wang. 2010. 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)*. Association for Computing Machinery, New York, NY, USA, 1029–1038.

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

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

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

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

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

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

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

[19] Phillip B. Gibbons. 2001. Distinct Sampling for Highly-Accurate Answers to Distinct Values Queries and Event Reports. In *Proceedings of 27th International Conference on Very Large Data Bases (VLDB '01)*. Morgan Kaufmann, 541–550.

[20] Amit Goyal, Francesco Bonchi, and Laks VS Lakshmanan. 2011. A data-based approach to social influence maximization. *Proc. VLDB Endow.* 5, 1 (2011), 73–84.

[21] Michael Greenwald and Sanjeev Khanna. 2001. Space-Efficient Online Computation of Quantile Summaries. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data (SIGMOD '01)*. Association for Computing Machinery, New York, NY, USA, 58–66.

[22] Jean-Loup Guillaume and Matthieu Latapy. 2004. Bipartite structure of all complex networks. *Inf. Process. Lett.* 90, 5 (2004), 215–221.

[23] Jean-Loup Guillaume and Matthieu Latapy. 2006. Bipartite graphs as models of complex networks. *Phys. A: Stat. Mech. Appl.* 371, 2 (2006), 795–813.

[24] Yucan Guo, Chenhao Ma, and Yixiang Fang. 2024. Efficient Core Decomposition over Large Heterogeneous Information Networks. In *40th IEEE International Conference on Data Engineering (ICDE)*. IEEE, to appear.

[25] Stefan Heule, Marc Nunkesser, and Alexander Hall. 2013. HyperLogLog in Practice: Algorithmic Engineering of a State of the Art Cardinality Estimation Algorithm. In *Proceedings of the 16th International Conference on Extending Database Technology (EDBT '13)*. Association for Computing Machinery, New York, NY, USA, 683–692.

[26] Aidan Hogan et al. 2021. Knowledge graphs. *ACM Comput. Surv.* 54, 4, Article 71 (2021), 37 pages.

[27] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Trans. Neural Netw. Learn. Syst.* 33, 2 (2021), 494–514.

[28] Yangqin Jiang, Yixiang Fang, Chenhao Ma, Xin Cao, and Chunshan Li. 2022. Effective Community Search over Large Star-Schema Heterogeneous Information Networks. *Proc. VLDB Endow.* 15, 11 (2022), 2307–2320.

[29] Sangkeun Lee, Sungchan Park, Minsuk Kahng, and Sang goo Lee. 2013. PathRank: Ranking nodes on a heterogeneous graph for flexible hybrid recommender systems. *Expert Syst. Appl.* 40, 2 (2013), 684–697.

[30] Yitong Li, Chuan Shi, Philip S Yu, and Qing Chen. 2014. HRank: A path based ranking method in heterogeneous information network. In *Web-Age Information Management - 15th International Conference, WAIM 2014, Macau, China, June 16-18, 2014. Proceedings*. Springer, 553–565.

[31] Linyuan Lü, Tao Zhou, Qian-Ming Zhang, and H. Eugene Stanley. 2016. The H-index of a network node and its relation to degree and coreness. *Nat. Commun.* 7, 1 (2016), 10168.

[32] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. 2019. Anytime Bottom-Up Rule Learning for Knowledge Graph Completion. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 3137–3143.

[33] Shodai Mihara, Sho Tsugawa, and Hiroyuki Ohsaki. 2015. Influence Maximization Problem for Unknown Social Networks. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '15)*. Association for Computing Machinery, New York, NY, USA, 1539–1546.

[34] Mark E. J. Newman. 2002. Assortative mixing in networks. *Phys. Rev. Lett.* 89, 20 (2002), 208701.

[35] Michaek Kwok-Po Ng, Xutao Li, and Yunming Ye. 2011. 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)*. Association for Computing Machinery, New York, NY, USA, 1217–1225.

[36] Hung Q. Ngo, Ely Porat, Christopher Ré, and Atri Rudra. 2018. Worst-case Optimal Join Algorithms. *J. ACM* 65, 3, Article 16 (2018), 40 pages.

[37] Paul W Olsen, Alan G Labouseur, and Jeong-Hyon Hwang. 2014. Efficient top-k closeness centrality search. In *2014 IEEE 30th International Conference on Data Engineering*. IEEE, 196–207.

[38] Gerald Paul, Sameet Sreenivasan, Shlomo Havlin, and H. Eugene Stanley. 2006. Optimization of network robustness to random breakdowns. *Phys. A: Stat. Mech. Appl.* 370, 2 (2006), 854–862.

[39] Dimitrios Prountzos and Keshav Pingali. 2013. Betweenness centrality: algorithms and implementations. In *Proceedings of the 18th ACM SIGPLAN symposium on Principles and practice of parallel programming.* 35–46.

[40] Cheng Sheng, Yufei Tao, and Jianzhong Li. 2012. Exact and approximate algorithms for the most connected vertex problem. *ACM Trans. Database Syst.* 37, 2, Article 12 (2012), 39 pages.

[41] Chuan Shi, Xiangnan Kong, Yue Huang, Philip S. Yu, and Bin Wu. 2014. HeteSim: A General Framework for Relevance Measure in Heterogeneous Networks. *IEEE Trans. Knowl. Data Eng.* 26, 10 (2014), 2479–2492.

[42] Chuan Shi, Yitong Li, Philip S Yu, and Bin Wu. 2016. Constrained-meta-path-based ranking in heterogeneous information network. *Knowl. Inf. Syst.* 49 (2016), 719–747.

[43] Chuan Shi, Yitong Li, Philip S. Yu, and Bin Wu. 2016. Constrained-meta-path-based ranking in heterogeneous information network. *Knowl. Inf. Syst.* 49, 2 (2016), 719–747.

[44] Sriganesh Srihari and Hon Wai Leong. 2013. A survey of computational methods for protein complex prediction from protein interaction networks. *J. Bioinform. Comput. Biol.* 11, 02 (2013), 1230002.

[45] Panagiotis Strouthopoulos and Apostolos N. Papadopoulos. 2019. Core discovery in hidden networks. *Data Knowl. Eng.* 120 (2019), 45–59.

[46] Wen Sun, Achille Fokoue, Kavitha Srinivas, Anastasios Kementsietsidis, Gang Hu, and Guotong Xie. 2015. SQLGraph: An Efficient Relational-Based Property Graph Store. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15).* Association for Computing Machinery, New York, NY, USA, 1887–1901.

[47] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *Proc. VLDB Endow.* 4, 11 (2011), 992–1003.

[48] Youze Tang, Yanchen Shi, and Xiaokui Xiao. 2015. Influence Maximization in Near-Linear Time: A Martingale Approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15).* Association for Computing Machinery, New York, NY, USA, 1539–1554.

[49] Toshihiro Tanizawa, Shlomo Havlin, and H. Eugene Stanley. 2012. Robustness of onionlike correlated networks against targeted attacks. *Phys. Rev. E* 85, 4 (2012), 046109.

[50] Yufei Tao, Cheng Sheng, and Jianzhong Li. 2010. Finding Maximum Degrees in Hidden Bipartite Graphs. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD '10).* Association for Computing Machinery, New York, NY, USA, 891–902.

[51] Todd L. Veldhuizen. 2014. Triejoin: A Simple, Worst-Case Optimal Join Algorithm. In *Proceedings of the 17th International Conference on Database Theory (ICDT).* OpenProceedings.org, 96–106.

[52] Chenguang Wang, Yizhou Sun, Yanglei Song, Jiawei Han, Yangqiu Song, Lidan Wang, and Ming Zhang. 2016. RelSim: Relation Similarity Search in Schema-Rich Heterogeneous Information Networks. In *Proceedings of the 2016 SIAM International Conference on Data Mining (SDM).* SIAM, 621–629.

[53] Jianguo Wang, Eric Lo, and Man Lung Yiu. 2013. Identifying the Most Connected Vertices in Hidden Bipartite Graphs Using Group Testing. *IEEE Trans. Knowl. Data Eng.* 25, 10 (2013), 2245–2256.

[54] Min Wu, Xiaoli Li, Chee-Keong Kwoh, and See-Kiong Ng. 2009. A core-attachment based method to detect protein complexes in PPI networks. *BMC Bioinform.* 10 (2009), 169.

[55] Zhi-Xi Wu and Petter Holme. 2011. Onion structure and network robustness. *Phys. Rev. E* 84, 2 (2011), 026106.

[56] Yun Xiong, Yangyong Zhu, and Philip S. Yu. 2015. Top-k Similarity Join in Heterogeneous Information Networks. *IEEE Trans. Knowl. Data Eng.* 27, 6 (2015), 1710–1723.

[57] Konstantinos Xirogiannopoulos and Amol Deshpande. 2017. Extracting and Analyzing Hidden Graphs from Relational Databases. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17).* Association for Computing Machinery, New York, NY, USA, 897–912.

[58] Yixing Yang, Yixiang Fang, Xuemin Lin, and Wenjie Zhang. 2020. Effective and Efficient Truss Computation over Large Heterogeneous Information Networks. In *36th IEEE International Conference on Data Engineering (ICDE).* IEEE, 901–912.

[59] Yuyan Zheng, Chuan Shi, Xiaohuan Cao, Xiaoli Li, and Bin Wu. 2017. Entity Set Expansion with Meta Path in Knowledge Graph. In *Advances in Knowledge Discovery and Data Mining - 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part I.* Springer, Cham, 317–329.

[60] Yingli Zhou, Yixiang Fang, Wensheng Luo, and Yunming Ye. 2023. Influential Community Search over Large Heterogeneous Information Networks. *Proc. VLDB Endow.* 16, 8 (2023), 2047–2060.