Hubs Discovery from Hidden Networks in Knowledge Graphs

Anonymous Author(s)

ABSTRACT

Extensive research has been dedicated to the exploration of hidden networks within highly heterogeneous knowledge graphs (KGs). In this paper, we study the fundamental problem of finding hubs from the hidden network defined by any given meta-path. The main challenge in efficiently identifying hubs is to circumvent the prohibitive cost of explicitly constructing large hidden networks, while still being able to evaluate the node centrality scores effectively. To this end, we propose a novel *sketch propagation* framework that accurately estimates the node degrees and discovers hubs under the degree centrality measure without explicit network construction. Subsequently, we extend the sketch propagation framework to identify hubs measured by h-index. Furthermore, we develop efficient pruning techniques to expedite *personalized* queries, i.e., whether a node q is a hub, under both degree centrality and h-index measures. Extensive experiments confirm the efficacy and efficiency of our proposed solutions. Our methods demonstrate a significant two orders of magnitude speedup compared to efficient exact methods, while consistently maintaining accuracy exceeding 95%.

ACM Reference Format:

Anonymous Author(s). 2023. Hubs Discovery from Hidden Networks in Knowledge Graphs. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Knowledge Graphs (KGs) serve as crucial repositories for organizing highly heterogeneous information, yet extracting meaningful insights from them remains a challenge [**?**]. To address this challenge, meta-paths [8, 17, 25] are widely employed to model implicit relationships between nodes in the KG and define hidden networks induced by these paths. For instance, in academic KGs like DBLP [1] or MAKG [2], a meta-path such as $(author)A-(paper)P-(author)A$ can reveal a hidden co-author network. The connections in the hidden graph facilitate a deeper understanding of the KG's underlying structure, unveiling hidden patterns and facilitating analysis across various domains, such as recommendation systems [10, 20, 21] and information retrieval [5, 7, 14]. **[[Find more citations from different domains]]**

In this paper, we investigate the fundamental problem of *Hubs Discovery* (HUD) in the hidden networks. Specifically, we aim to identify nodes within the top-λ percentile of *the hidden network induced by a given meta-path $\mathcal{M}$ in the knowledge graph $\mathcal{G}$, based on the measures of degree centrality and h-index.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Techniques. One can construct a hidden network explicitly and then identify the hubs. However, the construction is often time and memory-prohibitive for large hidden networks [24]. Alternatively, one can efficiently compute the neighbors of a node by traversing meta-path instances in the KG without explicit construction. However, computing hubs requires accessing neighborhood information for all nodes in the hidden graph, and the exact method is not scalable as each edge involved in any meta-path instances is repeatedly traversed, up to $O(|V_{\mathcal{M}}|)$ times (where $|V_{\mathcal{M}}|$ denote the number of nodes in the hidden graph).

Inspired by cardinality sketches [**?**], we propose a novel *sketch propagation* framework for the HUD problem. To estimate the degrees of nodes in the hidden network, we construct neighborhood cardinality sketches for each node by iteratively propagating the sketches along the edges that appeared in the matching instances of a given meta-path $\mathcal{M}$. As such, the degree information is jointly estimated for all nodes in a *single* propagation process, thereby significantly improving the efficiency of identifying hubs under the degree centrality.

We further extend our *sketch propagation* framework to address the HUD problem under the h-index measure [**?**]. The h-index of a node is determined by considering the degrees of its neighbors. However, the sketches only capture information about the degree of each node, without considering the degrees of its neighbors. To overcome this limitation, we introduce a pivot-based algorithm that builds upon our sketch propagation framework. Instead of directly estimating the h-index for each node, our pivot-based algorithm identifies the hubs by iteratively selecting nodes with an h-index higher than that of a randomly chosen pivot node, in a similar spirit to the quick-sort algorithm.

Furthermore, we provide efficient solutions for the personalized HUD queries, to quickly determine if a given query node is a hub in the hidden network. By leveraging the sketches and maintaining a lower bound on the number of nodes with higher centrality than the query node, we enable early termination when the lower bound exceeds the desired rank percentile in the hidden network. We design the early termination optimization such that it applies to the personalized HUD query under both degree and h-index centrality.

Theoretically, we prove that the sketch propagation framework can provide unbiased and quick estimations for *all* HUD queries under both centrality measures. Our experiments also reveal that the estimations converge much faster than the theoretical upper bound in terms of the number of propagations, enabling orders of magnitude speedups against efficient exact methods.

Our contributions are summarized as follows:

- We investigate the novel problem of *hubs discovery* (HUD) in hidden networks obtained from knowledge graphs through meta-paths. Our study focuses on both degree and h-index centrality measures (Section 2).
- We propose the *sketch propagation* framework for efficient degree estimations and further devise an early termination optimization for the personalized HUD problem. (Section 3).

- We further devise a novel pivot-based algorithm that utilizes the sketches to identify hubs, and we adapt it to handle personalized queries based on the h-index centrality (Section 4).
- Through extensive experiments, we have demonstrated the effectiveness and efficiency of our *sketch propagation* framework. Our methods have proven to scale well for large hidden networks, outperforming exact methods in scenarios where they fail. In other cases, our methods achieve significant speedups of two orders of magnitude compared to baseline approaches, while maintaining an accuracy of over 0.95. (Section 5).

2 PRELIMINARIES

In this section, we first introduce the basic notations and the formulation of *hub discovery* (HUD) problems on hidden networks induced by meta-paths from knowledge graphs. Next, we present a basic exact algorithm for the HUD problems.

2.1 Notations and Problem Statement

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

Meta-paths are commonly used to extract hidden networks from a KG [**?**]. [[Could we change the symbols of edges in meta-paths as $A \xrightarrow{\text{write}} P$?]] We provide a formal definition of *meta-path* on the KG as:

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

Next, we define the matching instance to a meta-path as:

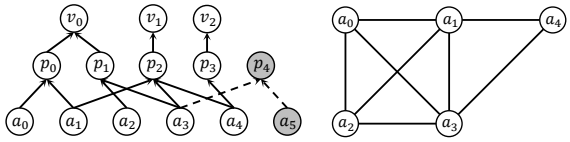
Definition 2.2 (Matching Instance). A matching instance M to a meta-path $\mathcal{M}$ is a sequence of node and edge instances in $\mathcal{G}$ denoted by $M(v_0, e_0, v_1, \ldots, v_{L-1}, e_{L-1}, v_L) \triangleright \mathcal{M}$ satisfying that:

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

When the context is clear, we use $M(v_0, v_1, \ldots, v_L)$ or simply M to denote the matching instance and use $M(x_i)$ to denote the node v_i in M. [[The following should be better explained, not only cite existing literature.]] Although our methods can support any meta-path $\mathcal{M}$, we concatenate $\mathcal{M}$ and $\mathcal{M}^{-1}$ to define a homogeneous hidden network $H_{\mathcal{M}}$ as follows in line with existing literature [**?**]:

Definition 2.3 (Hidden Network). Given a KG $\mathcal{G}$ and a meta-path $\mathcal{M}$, $H_{\mathcal{M}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ is a hidden network induced by $\mathcal{M}$ from $\mathcal{G}$ if its node set $V_{\mathcal{M}}$ contains all the nodes in $\mathcal{V}$ that match x_0 in any instance of $\mathcal{M}$ and there is an edge $e = (u, v)$ in its edge set $E_{\mathcal{M}}$ for any two nodes $u, v \in V_{\mathcal{M}}$ such that there exists $M \triangleright \mathcal{M}$ and $M' \triangleright \mathcal{M}^{-1}$ in $\mathcal{G}$ with (1) $M(x_0) = u$, $M'(x_0) = v$ and (2) $M(x_L) = M'(x_L)$.

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



(a) Exemplary knowledge graph $\mathcal{G}$ (b) Hidden network H

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

Example 2.4. Figure 1 presents an exemplary academic KG with three node types A ('author'), P ('paper'), and V ('venue') and two edge types $A \rightarrow P$ ('write') and $P \rightarrow V$ ('publish'). A sequence $(A,$ 'write', $P,$ 'publish', $V)$ denotes a meta-path $\mathcal{M}$, which induces the hidden collaboration network 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 reverse $\mathcal{M}^{-1}$ of $\mathcal{M}$. More intuitively, two authors a_0 and a_2 are connected since they ever published papers on the same venue v_0.

Problem Statement. In this paper, we study the problem of HUb Discovery (HUD) from hidden networks. A node v is considered a hub in H if it is ranked in the top percentiles according to a specific function $c : V \rightarrow \mathbb{R}_{\geq 0}$ that measures the importance (e.g., degree centrality and h-index) of each node v in H. [[We should explain more about the definition of hubs.]]

Problem 1 (Top-λ HUD Query). *Given a hidden network H and $\lambda \in (0, 1)$, find every node $u \in V$ such that $c(u) \geq c(v^\lambda)$, where v^λ is ranked the $(1-\lambda)$-th percentile under a function $c(\cdot)$.*

We also consider *personalized* queries, i.e., to determine whether a query node q is a top-λ hub of H.

Problem 2 (Personalized HUD Query). *Given a hidden network H, a query node q, and $\lambda \in (0, 1)$, determine if $c(q) \geq c(v^\lambda)$.*

Processing HUD queries on large-scale hidden networks is computationally expensive. In this work, we consider approximate HUD queries, which can be answered efficiently by randomized algorithms based on the notion of ϵ-separable sets [9]. [[To my understanding, this notion comes from the approximation for heavy hitters and quantiles. For our case, "quantile" or "percentile"?]]

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

Accordingly, we define the approximate versions of top-λ HUD and personalized HUD queries as follows:

Problem 3 (ϵ-Approximate Top-λ HUD Query). *Given a hidden network H and $\epsilon, \lambda \in (0, 1)$, return a set S of nodes in H such that $S_\epsilon^+(v^\lambda) \subseteq S$ and $S \nsubseteq S_\epsilon^-(v^\lambda)$.*

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

[[We may put the Bernstein inequality later, before the first time it is used.]] We use the Bernstein inequality [3] to calculate the confidence bounds of our proposed algorithms.

Theorem 2.6 (Bernstein Inequality). *Let $X_1, X_2, \ldots, X_\theta$ be real-valued i.i.d. random variables such that $X_i \in [0, r]$, $\mathbb{E}[X_i] = \mu$, and $\mathrm{Var}[X_i] = \sigma^2$. Let $\overline{X}_\theta = \frac{1}{\theta}\sum_{i=1}^{\theta} X_i$ denote the empirical mean. With probability at least $1-p$, it holds that*

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

In this work, we first study the HUD queries based on the degree centrality [6] and then extend our algorithms to consider the h-index centrality [11]. Before presenting our algorithms, we first give the definition of *h-index*.

Definition 2.7 (H-index). Given a node $v \in H$, the h-index of v is the largest value $h(v)$ such that v has no fewer than $h(v)$ neighbors of degree at least $h(v)$.

2.2 The Exact Algorithms

[[Some parts are missing: (1) a formal algorithm description; (2) essential background of the worst-case optimal join; (3) a formal time complexity analysis and whether the h-index version has higher time complexity.]]

Constructing large hidden networks explicitly can often be memory-prohibitive [24]. To address this, we propose a memory-efficient baseline solution for exact HUD queries, which computes the exact centrality of each node in H using the worst-case optimal join [13] over the *matching graph* of $\mathcal{M}$ over $\mathcal{G}$. We first introduce the notion of *matching graph* of a meta-path as follows:

Definition 2.8 (Matching Graph). Given a KG $\mathcal{G}$ and a meta-path $\mathcal{M}$, the matching graph of $\mathcal{M}$ over $\mathcal{G}$ is a subgraph $\mathcal{G}^*$ of $\mathcal{G}$ 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 $\mathcal{V}_{i+1}$.

For each node $u \in \mathcal{V}_i$, we use $\mathcal{V}^+(u)$ to denote the set of nodes in $\mathcal{V}_{i+1}$ connected with u through edge type y_i, i.e., the successors of u in the matching graph; and use $\mathcal{V}^-(u)$ to denote the set of nodes in $\mathcal{V}_{i-1}$ connected with u through edge type y_{i-1}, i.e., the predecessors of u in the matching graph. The matching graph can be extracted from $\mathcal{G}$ efficiently by a *forward* and *backward* breadth-first search (BFS). At each level $i \in (0, \ldots, L-1)$, the forward BFS iteratively visits all neighbors of each candidate matching node of x_i via edge type y_i as the candidates for x_{i+1}. Subsequently, the backward BFS visits all neighbors of the candidates for x_{i+1} via edge type y_i as the candidates for x_i at each level $i \in (L-1, \ldots, 0)$. Only the candidate nodes and their adjacent edges visited by both forward BFS and backward BFS are identified as the matching nodes and edges.

To avoid generating excessive intermediate results, the worst-case optimal join [**?**] is employed on the matching graph to compute the neighbors of each node $v \in V$. This process starts with a depth-first search (DFS) from node v and recursively backtracks the matching instances of $\mathcal{M}$ and $\mathcal{M}^{-1}$. To compute the h-index without *materializing* the hidden network, the worst-case optimal join is performed in two rounds. The first round also computes and stores the degree of each node $v \in V$. Then, the second round computes the h-index of each node v by combining its neighborhood $N(v)$ with the degrees stored in the first round.

The above worst-case optimal join-based algorithm encounters a significant bottleneck due to repeated edge traversals. Each edge in $\mathcal{E}^*$ is traversed up to $|V|$ times to compute the neighborhood of each node in V. Consequently, this leads to a time complexity of $O(|V| \cdot |\mathcal{E}^*|)$, which can be prohibitively time-consuming when dealing with large hidden networks.

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 top-λ HUD queries based on degree centrality. Additionally, we propose a sketch-based early termination technique to improve the efficiency of personalized HUD query processing.

3.1 Sketch Propagation for Approximate Top-λ HUD Queries

Our basic idea is to approximate the degree of each node in H by constructing KMV sketches for the neighborhood $N(v)$ of each node $v \in V$ without materializing H.

[[The KMV sketch was initially proposed for streaming distinct cardinality estimation [4].

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

There are two key challenges for using KMV sketches and the corresponding estimator for the HUD problem directly. First, the KMV sketch and other similar sketches are typically designed for streaming problems [], where one-pass scan of the collection C is affordable. However, for the HUD problem, scanning the neighborhood of each node is unaffordable since we cannot obtain $N(v)$ for each node v in G. Secondly, previous works only ensures asymptotic error bounds for the estimation when $|C| \to \infty$. Although it's reasonable to assume a large number of elements in streaming problems, the size of neighborhood of each node in G can be relatively small. Thus, the original estimator for KMV sketch does not provide meaningful error bounds for the HUD problem.

To address these two challenges, we first propose the *sketch propagation* framework that can generate a KMV sketch for $N(v)$ of each $v \in H$ without explicitly generating and scanning $N(v)$ and then proposed a different estimator which can provide tighter error bound for the estimation result.]]

Our framework is based on the following notion of *images*. [[Is "*image*" a formal name in this context? To my understanding, the images directly come from the definition of meta-path and hidden networks and are precisely the intermediate results for meta-path enumeration.]]

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

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

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

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

For the definition of hidden networks, it is obvious that $N(v) = \mathcal{I}_b(v)$ for each node $v \in V$.

[[We should be consistent between u/v, node/vertex.]]

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

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

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

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

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

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

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

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

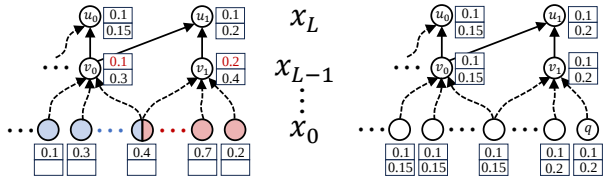
Algorithm 1: Sketch Propagation

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

1 **forall** $u \in \mathcal{V}^*$ **do**
2 **for** $t = 1$ **to** θ **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 $r(u) = \text{Rand}(0,1)$ to $\mathcal{K}_f[u][t]$;
5 $\widetilde{N} \leftarrow$ Propagate$(\mathcal{K}_f, \mathcal{K}_b)$;
6 $S \leftarrow$ top-$(\lambda \cdot |V|)$ nodes with the 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** θ **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** θ **do**
25 $\mathcal{K}[u][t] \leftarrow \oplus_{v \in \mathcal{V}^\circ(u)} \mathcal{K}[v][t]$;

Algorithm 1 describes the procedure of the *sketch propagation* framework. It receives a knowledge graph $\mathcal{G}$, a meta-path $\mathcal{M}$, a percentile $\lambda \in (0,1)$, 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 as the result for the (approximate) top-λ HUD query. It first initializes two arrays $\mathcal{K}_f$ and $\mathcal{K}_b$ to store θ KMV sketches for the respective forward and backward images of each node $u \in \mathcal{V}^*$ (Lines 1–4). Next, it calls the function Propagate, which constructs KMV sketches within $\mathcal{K}_f$ and $\mathcal{K}_b$ (Lines 9–13) and estimates the degree of each node u in H using $\mathcal{K}_b$ (Lines 14–21). Specifically, the function Receive depicts the step for KMV sketch construction when each node receives the sketches propagated from other nodes. Given a node u, the array $\mathcal{K}$ to store sketches, and a parameter $\circ$ indicating the propagation direction, it scans the random numbers in the sketches of nodes in either $\mathcal{V}^+(u)$

or $\mathcal{V}^-(u)$ and maintains the k-smallest numbers in $\mathcal{K}[u][t]$ with a min-heap of size k (Lines 24–25). Finally, it returns the set of $\lambda|\mathcal{V}_0|$ nodes with the largest estimated degrees as S (Lines 6–7).



(a) Forward propagation (b) Backward propagation

Figure 2: Illustration of forward and backward propagation with $k = 2$ and $\theta = 1$. Blue and red nodes represents the forward images $\mathcal{I}_f(v_0)$ and $\mathcal{I}_f(v_1)$, respectively.

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

Theoretical Analysis. Next, we will prove that the *sketch propagation* framework accurately approximates the top-λ HUD query with high probability (w.h.p.) (cf. Theorem 3.7). For the proof, we first establish the following two lemmas: (1) Lemma 3.5 indicates the unbiasedness of sketch propagation in estimating the image sizes for each node in the matching graph; (2) Lemma 3.6 demonstrates the concentration properties that guarantee tight approximations around the true values.

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

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

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

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

which indicates that $I = \frac{k}{\mu} - 1$. □

Remark.If $I < k$, a KMV sketch will trivially obtain the true value of I as the number of random numbers in the sketch.

[[The following lemma shows that the estimator in Lemma 3.5 can be concentratedly bounded with Bernstein's inequality for images of any sizes, which is different from the original estimator of KMV sketch in Def. 3.1.]]

Lemma 3.6. *Given any $\delta, p \in (0, 1)$ and node $u \in \mathcal{V}^*$, if $\theta = O(\frac{n}{\delta k} \ln \frac{1}{p})$, $(1-\delta) \cdot I \leq \widetilde{I} \leq (1+\delta) \cdot I$ will hold with probability at least $1 - p$.*

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

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

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

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

According to Bernstein's inequality [3], for any $p \in (0, 1)$, we have

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

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

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

Or equivalently,

$$\frac{\delta I k}{(I+1)(I+1+\delta I)} \cdot \theta - \sqrt{\sigma^2 \ln \frac{2}{p}} \cdot \sqrt{\theta} - 2\ln\frac{1}{p} \geq 0.$$

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

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

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

Theorem 3.7. *Let $\theta = O(\frac{n}{\epsilon k} \log \frac{|V|}{p})$ for any $\epsilon \in (0, 1)$, the sketch propagation framework answers an ϵ-approximate top-λ HUD query under degree centrality correctly with probability at least $1 - p$.*

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

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

hold due to Lemma 3.6 and the definitions of $S_\epsilon^-(v^\lambda)$ and $S_0^+(v^\lambda)$. By setting $\delta < \frac{\epsilon}{2-\epsilon}$, we have $|\widetilde{N}(v)| < |\widetilde{N}(u)|$, i.e., the sketches rank v lower than u, with probability at least $1 - p$. Taking the union bound over all nodes in $S_0^+(v^\lambda)$ and setting $\theta = O(\frac{n}{\epsilon k} \log \frac{|V|}{p})$, the

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

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

3.2 More Efficient Processing of Approximate Personalized Top-λ HUD Queries

Given a query node q, a personalized HUD query can be answered directly using the node degrees in the hidden network estimated through *sketch propagation*. Specifically, if q is ranked in the top-λ percentile 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.

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

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

Lemma 3.9. *Given a query node q, if there exists a node $u \in \mathcal{V}^*$ such that $I_f(u) \geq \lambda|V|$ and $I_b(u) \geq |N(q)|$, then the answer to the personalized top-λ HUD query is* FALSE.

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

Based on Lemma 3.9, 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 that $\widetilde{I}_f(u) \geq \lambda|V|$ and $\widetilde{I}_b(u) \geq |N(q)|$. However, this may cause a false negative result once $I_f(u)$ and $I_b(u)$ are overestimated. To avoid such errors, we only terminate sketch propagation when $\widetilde{I}_f(u)$ and $\widetilde{I}_b(u)$ are significantly larger than $\lambda|V|$ and $|N(q)|$, respectively. Specifically, the early termination is performed only when a node u satisfies $\widetilde{I}_f(u) \geq (1+\beta)\cdot\lambda|V|$ and $\widetilde{I}_b(u) \geq (1+\beta)\cdot|N(q)|$. By selecting an appropriate value of β, we can ensure that the probability of a false negative is small enough.

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

Algorithm 2: OptimizedReceive

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

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

[[Similarly, after algorithm description; do we need an additional for this example?]]

Example 3.10. In Figure 2, during the backward propagation, the sketch is constructed as $\mathcal{K}(\mathcal{I}_b(v_0)) = \mathcal{K}(\mathcal{I}_b(u_0)) \oplus \mathcal{K}(\mathcal{I}_b(u_1)) = \{0.1, 0.15\} \oplus \{0.1, 0.2\} = \{0.1, 0.15\}$. Thus, the algorithm estimates $\widetilde{I}_b(v_0) = \frac{2}{0.15} - 1 \approx 12.3$. The size $\widetilde{I}_f(v_0) = \frac{2}{0.3} - 1 \approx 5.7$ is estimated from the sketch $\mathcal{K}(\mathcal{I}_f(v_0))$ in Figure 2(a). Assuming $\lambda = 0.01$, $|V| = 500$, $|N(q)| = 9$ and setting $\beta = 0$, we have $\widetilde{I}_f(u_1) > 0.01 \cdot 500$ and $\widetilde{I}_b(u_1) > 9$, and thus sketch propagation can be early terminated.

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

Theorem 3.11. *If $\theta = O(\frac{n}{\epsilon k}\log\frac{|V|}{p})$ and $\beta = \frac{\ln p_e - \ln|\mathcal{V}^*|}{\ln p - \ln|V|}\cdot\epsilon$, the probability that a personalized HUD query whose answer is* TRUE *is incorrectly answered with* FALSE *is at most p_e.*

Proof. Early termination occurs only when there exists a node $u \in \mathcal{V}^*$ such that $\widetilde{I}_f(u) \geq (1+\beta)\cdot\lambda|V|$ and $\widetilde{I}_b(u) \geq (1+\beta)\cdot|N(q)|$.

Thus, sketch propagation will not be early terminated incorrectly if

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

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

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

Combining Theorem 3.6 and Eqn. 6, if $\theta = O(\frac{n}{\epsilon k}\log\frac{|V|}{p})$, we have

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

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

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

holds with probability at least $1-p_e$.

According to Lemma 3.9, since the answer to a personalized HUD query is TRUE, $I_f(u) < \lambda|V|$ or $I_b(u) < |N(q)|, \forall u \in \mathcal{V}^*$. Combined with Eqn. 7, Eqn. 5 holds with probability at least $1-p_e$. □

[[Add complexity analysis if early termination leads to lower complexity (amortized complexity for all nodes is acceptable)]]

4 EXTENSION TO H-INDEX

In this section, we extend the sketch propagation framework to process approximate top-λ HUD queries based on the h-index measure. To compute the h-indexes of the nodes in H, we need to compute the degrees of their neighbors. However, the KMV sketches can only estimate the neighborhood sizes of each node but fail to provide the degree estimations of their neighbors. Therefore, we propose a novel pivot-based algorithm that can estimate the set of nodes ranked in the top-λ percentile in terms of h-index without explicitly calculating the h-index of every node in H.

4.1 Pivot-based Algorithm for Approximate Top-λ HUD Queries

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-indexes in comparison with $h(u)$, i.e., $Q^+(u) = \{v \in V \mid h(v) > h(u)\}$ 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 greater h-index values 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. Such a partitioning strategy allows for efficient bisection when identifying the nodes in the top-λ percentile, i.e., $S_0^+(v^\lambda)$. If $u \in S_0^+(v_\lambda)$, we can simply add all nodes in $Q^+(u)$ to the result set and exclude them from consideration in the subsequent iteration. On the contrary, if $u \in S_0^-(v^\lambda)$, the nodes in $Q^+(u)$ may contain those in both $S_0^+(v^\lambda)$ and $S_0^-(v^\lambda)$, which require further partitioning in next iterations. Meanwhile, all nodes in $Q^-(u)$ must not be in $S_0^+(v^\lambda)$ and are excluded from consideration. The above iterative process proceeds until all nodes have been decided. Then, all the nodes in $S_0^+(v^\lambda)$ have been added to the result set.

Pivot-based Partitioning Method. The key challenge lies in partitioning the nodes in H without explicitly computing the h-index of each node. In fact, we can compare the h-indexes of two nodes using only one of them. By the definition of h-index, for any two nodes $u, v \in V$, we have $h(v) \ge h(u)$ 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): Estimating $h(u)$. We first employ *sketch propagation* on the matching graph to estimate the degrees of all nodes in H. Subsequently, at each iteration, we scan through the set of neighbors $N(u)$ 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 $\tilde{h}(u)$ of node u.

[[How is the case of $h(u) = h(v)$ handled?]]

Stage (II): Estimating $Q^+(u)$ and $Q^-(u)$. Once we have the approximate h-index $\tilde{h}(u)$, we proceed to eliminate nodes in H with degrees smaller than $\tilde{h}(u)$. 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 $\tilde{h}(u)$ in the original graph H, denoted as $\tilde{H}(u)$. Finally, we obtain $Q^+(u)$ as the set of nodes with estimated degrees greater than $\tilde{h}(u)$ in the induced subgraph of $\tilde{H}(u)$, while the remaining nodes are added to $Q^-(u)$.

[[Use $\mathcal{N}$ for the neighbor set to distinguish them with the estimated degree $\tilde{N}$? Or use $\tilde{n}$ for the estimated degree? $\mathcal{K}_f$ and $\mathcal{K}_b$ are reused and should be changed in the While loop.]]

Algorithm 3 depicts the procedure of our pivot-based algorithm. It also receives a knowledge graph $\mathcal{G}$, a meta-path $\mathcal{M}$, the percentile 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) top-λ HUD 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 $\tilde{h}(u)$ (Lines 8-12). Then, it initializes two new arrays $\mathcal{K}_f$ and $\mathcal{K}_b$ of KMV sketches for the nodes with degrees greater than $\tilde{h}(u)$ (Lines 13–17). After that, the Propagate function is performed over $\mathcal{K}_f$ and $\mathcal{K}_b$ to estimate the number of neighbors with degrees larger than $\tilde{h}(u)$ for each node in Q (Line 18). Accordingly, it partitions Q into $Q^+(u)$ and $Q^-(u)$ according to $\tilde{h}$ (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).

Theoretical Analysis. Next, we prove that the pivot-based algorithm returns the correct answer to an approximate HUD query based on h-index centrality 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.

Algorithm 3: Pivot-based Algorithm

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

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

Lemma 4.1. *For any $\delta, p \in (0,1)$, if $\theta = O(\frac{n}{\delta k}\log\frac{|V|}{p})$, we have $(1-\delta)\cdot h(u) \leq \tilde{h}(u) \leq (1+\delta)\cdot h(u)$ with probability at least $1-p$ for the pivot node u in each iteration.*

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

By combining Lemma 3.6 with the union bound over V, when $\theta = O(\frac{n}{\delta k}\log\frac{|V|}{p})$ in the first stage, we have $|\tilde{N}(v_i)| \geq (1-\delta)|N(v_i)| \geq (1-\delta)h(u)$ for $0 \leq i \leq h(u)-1$ with probability at least $1-p$. Thus, node u has $h(u) \geq (1-\delta)h(u)$ neighbors with estimated degrees larger than or equal to $(1-\delta)h(u)$. Thus, $\tilde{h}(u) \geq (1-\delta)h(u)$ with probability at least $1-p$. Similarly, we have $|\tilde{N}(v_i)| \leq (1+\delta)|N(v_i)| \leq (1+\delta)h(u)$ for $i \geq h(u)$ with probability at least $1-p$. Thus, node u has no more than $h(u)$ neighbors with estimated degrees larger than $(1+\delta)h(u)$, and we have $\tilde{h}(u) \leq (1+\delta)h(u)$ with probability at least $1-p$. □

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

Lemma 4.2. *For any $\delta' \in (0,1)$ and pivot node u, by setting $\theta = O(\frac{n}{\delta' k}\log\frac{|V|}{p})$, we have $S^-_{\delta'}(u) \subseteq Q^-(u)$ and $S^+_{\delta'}(u) \subseteq Q^+(u)$ hold with probability at least $1-p$.*

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

Subsequently, we can formally analyze the correctness of the pivot-based algorithm for processing ϵ-approximate top-λ HUD queries with h-index centrality in the following theorem.

Theorem 4.3. *For any $\epsilon \in (0,1)$, by setting $\theta = O(\frac{n}{\epsilon k}\log\frac{|V|}{p})$, the pivot-based algorithm returns the answer to an ϵ-approximate top-λ HUD query based on h-index centrality correctly with probability at least $1-p$.*

Proof. We prove the theorem by examining three different cases of the ranges of the h-index $h(u)$ for the pivot node u. For each case, we show the correctness of the first iteration of the pivot-based algorithm, while subsequent iterations are proven by induction. By selecting $\delta' < \frac{\epsilon}{2}$ as Lemma 4.2, we establish that:

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

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

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

Finally, the amortized time complexity of the sorting-based algorithm is $O(\frac{n\cdot|\mathcal{E}^*|}{\epsilon}\cdot \ln\frac{|V|}{p}\cdot \ln|V|)$ since the pivot-based algorithm performs sketch propagation framework $O(\ln V)$ times in amortize.

4.2 More Efficient Processing of Approximate Personalized Top-λ HUD Queries

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

Corollary 4.4. *For any $\epsilon \in (0,1)$, by setting $\theta = O(\frac{n}{\epsilon k}\log\frac{|V|}{p})$, the pivot-based algorithm answers an ϵ-approximate personalized top-λ HUD query based on h-index centrality correctly with probability at least $1-p$.*

The proof is similar to that of Lemma 4.2 and omitted due to space limitations.

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

Lemma 4.5. *Given a query node q and percentile λ, if there exists a node $u \in \mathcal{V}^*$ such that $I_f(u) \geq h(q)$ and $I_b(u) \geq \max(h(q), \lambda|V|)$, then the answer to the personalized top-λ HUD query must be* FALSE.

Proof. Note that each pair of nodes in $\mathcal{I}_b(u)$ and $\mathcal{I}_f(u)$ are a neighbor of each other in H according to the proof of Lemma 3.9. 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 $I_f(u) \geq h(q)$ and $I_b(u) \geq \max(h(q), \lambda\cdot|V|)$, there exists at least $I_b(u) \geq \lambda|V|$ nodes with h-indexes larger than or equal to $h(q)$. Therefore, q is not ranked top-λ percentile in terms of h-index. □

Based on Lemma 4.5, sketch propagation can be early terminated during *Stage (I)* of the pivot-based partitioning method. Specifically, we use the KMV sketches to estimate $I_f(u)$ and $I_b(u)$ for each $u \in \mathcal{V}^*$. If there exists a node $u \in \mathcal{V}^*$ such that $\tilde{I}_f(u) \geq (1+\beta)N(q)$ and $\tilde{I}_b(u) \geq (1+\beta)\max(N(q), \lambda|V|)$, the algorithm can be early terminated since $N(q)$ is an upper bound of $h(q)$.

[[Take care of the notations n, N, $\tilde{N}$. They may be misused.]]

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

Theorem 4.6. *If $\theta = O(\frac{n}{\epsilon k}\log\frac{|V|}{p})$ and $\beta = \frac{\ln p_e - \ln|\mathcal{V}^*|}{\ln p - \ln|V|}\cdot\epsilon$, the pivot-based algorithm with early termination answers a personalized top-λ HUD query with probability at least $1-p_e$.*

The proof is similar to that of Theorem 3.11 and omitted due to space limitations.

5 EXPERIMENTS

5.1 Experiments Setup

Datasets. We conduct our experiments with seven real-world network datasets. The network statistics used in our experiments are reported in Table 1.

Table 1: Network statistics.

Dataset	$\|V\|$	$\|E\|$	$\|\mathcal{T}\|$	$\|\mathcal{T}_E\|$	$\|\mathbb{M}\|$
acm	10.9K	548K	4	8	20
dblp	26.1K	239.6K	4	6	48
freebase	180K	1057.7K	8	36	151
imdb	21.4K	86.6K	4	6	679
pubmed	63.1K	236.5K	4	10	172
yelp	82.5K	29.9M	4	4	2230

Query Generation. For each dataset, we use AnyBURL [12] to mine a set of candidate meta-paths $\mathbb{M}$. We set confidence threshold to 0.1 for AnyBURL and take the snapshot at 10 seconds. Note that AnyBURL discovers meta-paths with additional constraints and thus the possible number of meta-paths for validation can be much larger than the combinations of node types and edge types in KG, which renders processing HUD queries efficiently more significant. Column $|\mathbb{M}|$ in Table 1 denote the number of meta-paths mined for each dataset. For personalized HUD queries, we randomly sample 10 nodes as query nodes from $\mathcal{V}_0$ for each meta-path.

Compared Methods. To our best knowledge, no previous work addresses the HUD problem. Thus, we compare our *sketch propagation* framework based methods with the exact algorithm based on worst-case optimal join.

- ExactD applies worst-case optimal join to compute the exact degrees of each node in G induced by a given meta-path and returns the set of top-λ impactful nodes in G in terms of degree.
- ExactH applies worst-case optimal join to compute the exact h-indexes of each node in G and returns the set of top-λ impactful nodes in G in terms of h-index.
- GloD applies sketch propagation framework to estimate degrees of each node in G and returns the set of top-λ impactful nodes in G in terms of degree.
- PerD applies sketch propagation framework to estimate degrees of each node in G and only returns whether a given node q is top-λ impactful according to the estimated degrees.
- PerD$^+$ optimizes PerD with early termination strategy.
- GloH applies pivot-based algorithm to estimates the set of nodes in G that is top-λ impactful in terms of h-index.
- PerH applies the pivotal partitioning method with the given query node q as pivot node and estimates the set of nodes with h-index larger than q to determine if q is top-λ impactful in terms of h-index in G.
- PerH$^+$ optimizes PerH with early termination strategy.

Parameter Setup. We set $\lambda = 0.05$ by default, i.e. we are interested in top 5% impactful nodes in hidden networks. According to 5.3, the h-index based HUD queries can be well-approximated with smaller θ and k than degree based HUD queries, thus we set $\theta = 8$ and $k = 32$ by default for algorithms targeting degree based HUD queries (i.e. GloD,PerD,PerD$^+$), and set $\theta = 8$ and $k = 4$ for algorithms targeting h-index based HUD queries (i.e. GloH,PerH and PerD$^+$). For the early termination optimization, we set $\beta = 0$ by default.

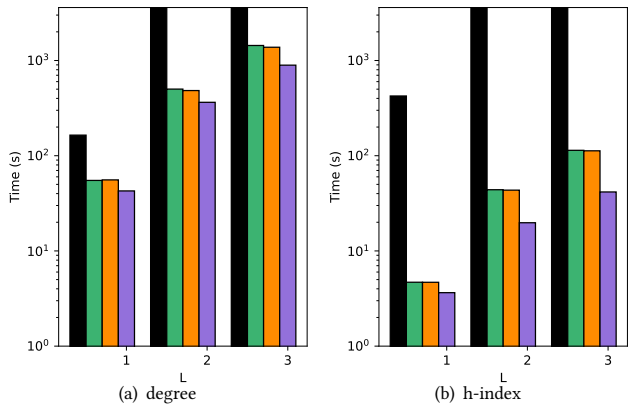


Figure 3: Efficiency of methods on `yelp` for meta-paths with varying meta-path length L.

Evaluation Metric. To avoid error caused by tied values in node centralities in G, we use a variant of the F1 score for evaluating the effectiveness of algorithms for global HUD queries (GloD andGloH). Let the S^* denote the set of nodes with centrality strictly larger than v^λ and S^+ denote the set of nodes with centrality larger than or equal to v^λ, we define $precision = \frac{|S \cap S^*|}{|S^*|}$ and $recall = \frac{|S \cap S^+|}{|S|}$. The variant F1 score is then defined as $F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$.

For personalized HUD queries, we evaluate the effectiveness of PerD, PerD$^+$,PerH and PerH$^+$ by the accuracy of each algorithm for determining whether a query node q is top-λ impactful in G.

We report the average of the F1 score and accuracy over all candidate meta-paths.

For efficiency evaluation, we report the average running time of algorithms for processing one a HUD query.

Environment. All experiments are conducted on a Linux Server with AMD EPYC 7643 CPU and 256 GB memory running Ubuntu 20.04. All algorithms are implemented using C++ with O3 optimization and executed with a single thread.

5.2 Efficiency Evaluation

In this section, we verify the efficiency improvement of our sketch propagation framework based methods over baselines. For global queries, we compare GloD and GloH with ExactD and ExactH. For personalized queries, we compare the most optimized PerD, PerD$^+$ and PerH, PerH$^+$ with ExactD and ExactH. Table. 2 shows the execution time per query of each algorithm as well as the speedup of our methods over baselines across datasets. We have the following observations.

First, our sketch propagation based algorithms is consistently more efficient than baselines across all datasets except `imdb` and provides upto two-orders magnitude speedup for both global queries and personalized queries (on `pubmed`, GloH and PerH$^+$ achieve 99.1x and 286.3x speedup over ExactH respectively). Our sketch propagation based algorithms are slightly slower than ExactD and ExactH on `imdb` since `imdb` is a very small datasets and thus the worst-case optimal join can be performed efficiently over the matching graph whereas our methods require additional times for sketch construction and centrality estimation.

Baselines ExactD and ExactH both cannot finish even one global or personalized query within 100 seconds since `yelp` is very dense with 29.9M edges and thus the matching graph can be very dense, which renders the worst-case optimal join algorithm prohibitive over `yelp`. In contrast, our methods can process both global and personalized queries efficiently. To further test the scalability of our sketch propagation framework based methods, we conduct experiments on `yelp` and extend the time limit to one hour.

5.3 Effectiveness Evaluation

In this section, we evaluate the effectiveness of our sketch propagation framework based methods for both HUD queries based on degree (GloD, PerD, PerD$^+$) and h-index (GloH, PerH, PerH$^+$).

Fig. 4(a) to 4(e) report the effectiveness of GloD,PerD and PerD$^+$ on each dataset varying parameter λ. We first observe that GloD achieves around 0.9 F1 score across all datasets for global HUD queries, and PerD and PerD$^+$ achieves consistently over 0.99 accuracy for personalized HUD queries on datasets `imdb`, `dblp`, `pubmed` and `acm`, and over 0.95 accuracy on `freebase`. Secondly, the effectiveness of GloD slightly increases with the increase of parameter λ. The reason is that given a larger λ, the HUD queries apply less strict requirement to nodes in the set of top-λ impactful nodes, which can be approximated easier.

Fig. 4(f) to 4(j) report the effectiveness of GloD,PerD and PerD$^+$ on each dataset varying parameter k and we have the following observations. First, the effectiveness GloD and PerD,PerD$^+$ all increase with the increase of parameter k. With a larger size of KMV sketch, the sketch propagation based methods can approximate the HUD queries with high F1 score or accuracy. Second, the personalized HUD queries can be approximated well by PerD and PerD$^+$ with high accuracy (over 0.9) even with a relatively smaller parameter k. The reason is that the personalized HUD queries only need to compare between the degree of q and v^λ. For the default setting of relatively small $\lambda = 0.05$, the degrees of most nodes in G tends to have a relatively large gap from the degree of v^λ, which makes it easier to compare the degree of q and the degree of v^λ. Nevertheless, in the technical report [], we conduct experiments with larger λ varying from 0.1 to 0.5, the result shows that PerD and PerD$^+$ still achieve high accuracy for personalized HUD queries consistently. Third, the gap between the accuracy of PerD$^+$ and the accuracy of PerD are very small even when $k = 2$ (with maximum accuracy gap 0.044 on `freebase`) and are further narrowed with the increase of k. The reason is that for larger k, the sketches can approximate the size of images with higher accuracy and thus the false early termination will happens with lower probability.

Fig. 4(k) to 4(o) report the effectiveness of GloD,PerD and PerD$^+$ on each dataset varying parameter θ. First, we still observe that the effectiveness of GloD increases monotonically with the increase of θ since the HUD queries can be approximated better with more KMV sketches. Besides, we further observe that the effectiveness of GloD is less sensitive to parameter θ than parameter k. The reason is that we set $k = 32$ by default, with which the GloD can already provide relatively accurate approximation to the HUD queries. For personalized HUD queries, the PerD and PerD$^+$ achieves consistently over 0.95 accuracy across all datasets varying θ.

Table 2: Overall efficiency. For ExactD and ExactH, the execution time over each dataset is reported. For GloD,PerD$^+$,GloH and PerH$^+$, the execution time (first sub-column for each method) as well as the speedup over baselines (second sub-column for each method) is reported.

Centrality	Degree							H-index						
Method	ExactD	GloD		PerD		PerD$^+$		ExactH	GloH		PerH		PerH$^+$	
imdb	0.0009s	0.0013s	0.66x	0.0016s	0.55x	0.0017s	0.53x	0.0013s	0.001s	1.23x	0.0015s	0.82x	0.0015s	0.81x
dblp	7.95s	0.63s	12.59x	0.67s	11.79x	0.27s	29x	14.59s	0.23s	64.32x	0.1s	148x	0.049s	298x
pubmed	5.96s	0.45s	13.24x	0.35s	17.05x	0.13s	45x	12.11s	0.09s	135.7x	0.056s	215x	0.032s	381x
acm	4.77s	1.77s	2.7x	2.04s	2.34x	0.81s	5.91x	8.01s	0.19s	41.7x	0.25s	32x	0.076s	105x
freebase	25.15s	0.87s	29x	0.92s	27.4x	0.41s	60.9x	50.5s	0.26s	191x	0.19s	268x	0.13s	379x

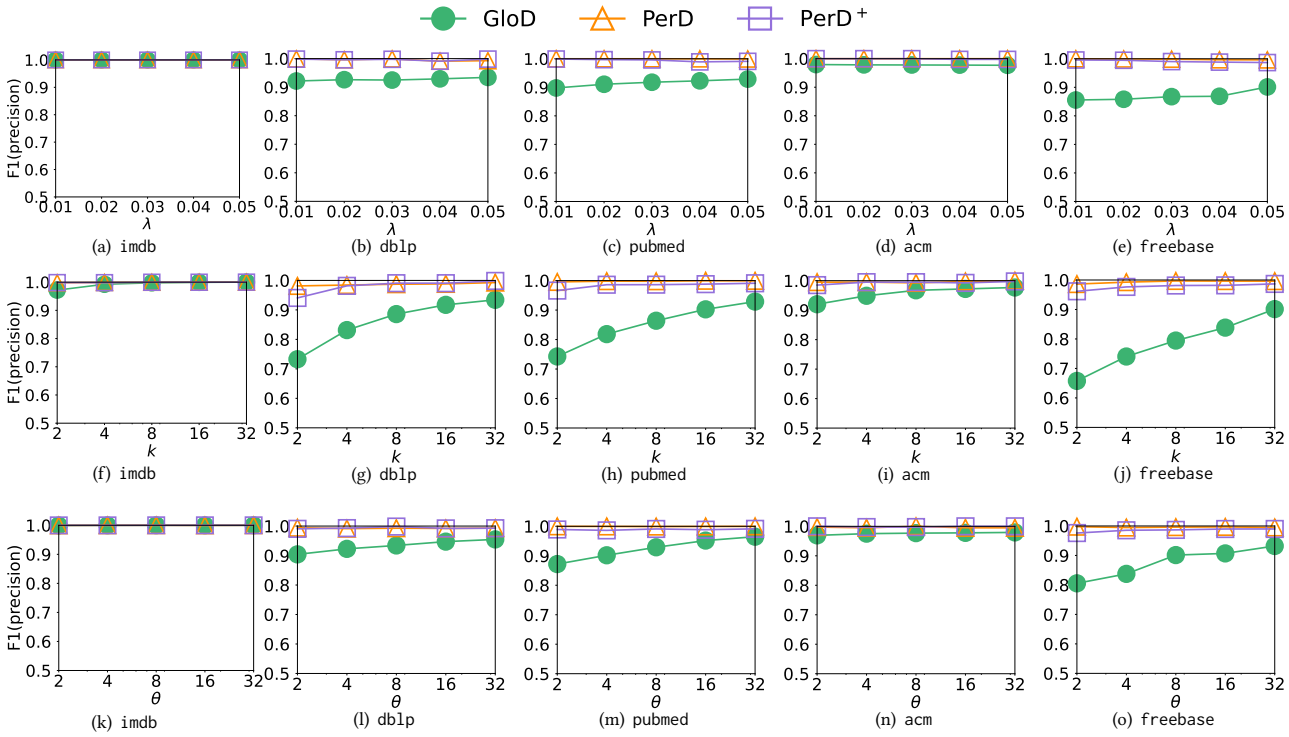


Figure 4: Effectiveness of GloD,PerD and PerD$^+$ for HUD queries based on degree centrality varying parameters. Subfigures (a)-(e): varying parameter λ. Subfigures (f)-(j): varying parameter k. Subfigures (k)-(o): varying parameter θ.

The effectiveness of methods GloH and PerH,PerH$^+$ are reported in Fig. 5. Similar observations can be obtained for methods processing h-index based HUD queries as for methods processing degree based HUD queries. Besides, we further observe that the GloH achieves higher F1 score for HUD queries based on h-index than GloD under same parameter settings. The reason is that nodes in G tend to take same values of h-index with other nodes, i.e. there are more repeated value of h-indexes in G then repeated value of degrees. Since our metric has removed the influence of repeated centrality values, HUD queries based on h-index can be approximated easier than HUD queries based on degree.

Overall, all the three methods GloD,PerD and PerD$^+$ achieve over 0.9 F1 score or accuracy across all datasets when $k = 32$ and $\theta = 8$ for HUD queries based on degree, whereas GloH,PerH and PerH$^+$ achieve over 0.9 F1 score or accuracy across all datasets when $k = 4$ and $\theta = 8$, which lead to the default parameter setting in our experiment.

6 RELATED WORK

We review studies in: (1) community search with semantic explanations,

Hidden network extraction. Hidden graph induced by join queries in relational database has been studied. Several works focused on

The problem of hub discovery has been considered over general hidden graphs [15, 18, 22], where the algorithm can only check the existence of an edge in the network through probe operation. Note that these algorithms are not applicable to hidden networks induced by meta-paths since the probe operation on these networks requires time-consuming BFS on the matching graph.

Recently, [24] proposed to

Meta-path Analysis. Meta-paths have been widely studied and used to extract information from KGs. To discover similar nodes in KGs, various similarity metrics have been proposed based on meta-paths [16, 17, 19, 23]. Pathsim [17] measure the similarity of two nodes in the heterogeneous networks according to the number

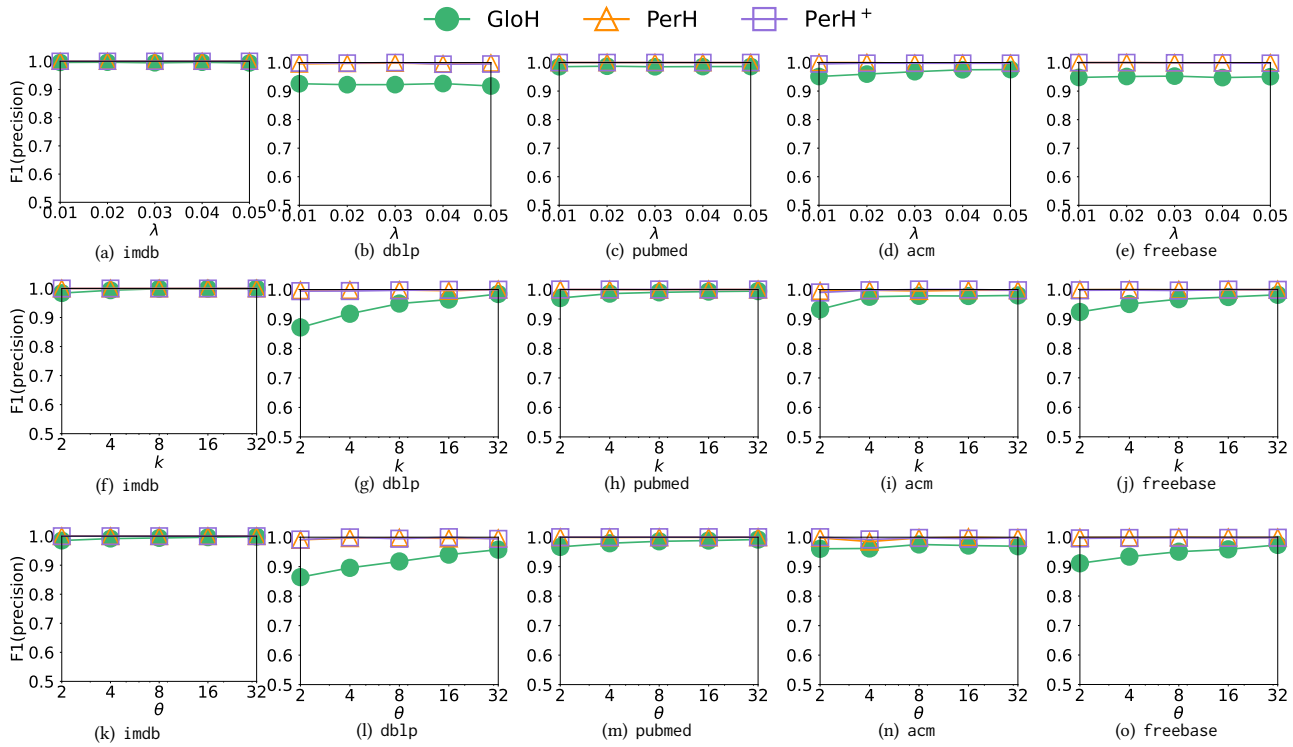


Figure 5: Effectiveness of GloH,PerH and PerH$^+$ for HUD queries based on h-index centrality varying parameters. Subfigures (a)-(e): varying parameter λ. Subfigures (f)-(j): varying parameter k. Subfigures (k)-(o): varying parameter θ.

of matching instances between then. Joinsim [23] is a similarity metric based on meta-paths that satisfying the triangle inequality to efficiently discover top-k similarity node pairs.

The meta-paths are also used for community search over KGs. Besides, the meta-paths are also used in recommendation systems based on KGs.

Although these works utilize the information extracted from the KGs by meta-paths, the hidden network induced by the meta-path has barely been examined. In this work, we studied the hidden network directly and discover hubs within these hidden networks.

7 CONCLUSION

We introduce the problem of Hubs Discovery over hidden networks (HUD) to seek impactful nodes given a hidden network induced by a meta-path in KG. We studied two kinds of centrality measurement. For the degree centrality, we devise the *sketch propagation* framework to estimate the. Then, we further optimize the propagation algorithm by early termination. Extensive experiments verified the effectiveness and efficiency of our solutions.

Future work. In this work, we focus on semantics represented by meta-paths. More sophisticated semantics can be represented with patterns in knowledge graphs. The complex structure of patterns presents unique technical challenges for impact estimation of entities in semantic graphs and we leave the HUD problem considering semantics represented by patterns for future work.

REFERENCES

[1] [n.d.]. DBLP. https://dblp.org.
[2] [n.d.]. Mircrosoft Academic Knowledge Graph. https://makg.org.
[3] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. 2007. Tuning Bandit Algorithms in Stochastic Environments. In *Proceedings of the 18th International Conference on Algorithmic Learning Theory (ALT 2007)*. Springer, Berlin, Heidelberg, 150–165.
[4] Kevin S. Beyer, Peter J. Haas, Berthold Reinwald, Yannis Sismanis, and Rainer Gemulla. 2007. On Synopses for Distinct-Value Estimation under Multiset Operations. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD '07)*. Association for Computing Machinery, New York, NY, USA, 199–210.
[5] Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. 365–374.
[6] 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.
[7] Faezeh Ensan and Ebrahim Bagheri. 2017. Document retrieval model through semantic linking. In *Proceedings of the tenth ACM international conference on web search and data mining*. 181–190.
[8] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. 2020. Effective and efficient community search over large heterogeneous information networks. *Proceedings of the VLDB Endowment* 13, 6 (2020), 854–867.
[9] 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.
[10] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering* 34, 8 (2020), 3549–3568.
[11] 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.
[12] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. 2019. Anytime Bottom-Up Rule Learning for Knowledge Graph Completion.. In *IJCAI*. 3137–3143.

[13] 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.
[14] Hadas Raviv, Oren Kurland, and David Carmel. 2016. Document retrieval using entity-based language models. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 65–74.
[15] Cheng Sheng, Yufei Tao, and Jianzhong Li. 2012. Exact and approximate algorithms for the most connected vertex problem. *ACM Transactions on Database Systems (TODS)* 37, 2 (2012), 1–39.
[16] Chuan Shi, Xiangnan Kong, Yue Huang, S Yu Philip, and Bin Wu. 2014. Hetesim: A general framework for relevance measure in heterogeneous networks. *IEEE Transactions on Knowledge and Data Engineering* 26, 10 (2014), 2479–2492.
[17] 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. *Proceedings of the VLDB Endowment* 4, 11 (2011), 992–1003.
[18] 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*. 891–902.
[19] 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*. SIAM, 621–629.
[20] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 417–426.
[21] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge graph convolutional networks for recommender systems. In *The world wide web conference*. 3307–3313.
[22] Jianguo Wang, Eric Lo, and Man Lung Yiu. 2012. Identifying the most connected vertices in hidden bipartite graphs using group testing. *IEEE Transactions on Knowledge and Data Engineering* 25, 10 (2012), 2245–2256.
[23] Yun Xiong, Yangyong Zhu, and S Yu Philip. 2014. Top-k similarity join in heterogeneous information networks. *IEEE Transactions on Knowledge and Data Engineering* 27, 6 (2014), 1710–1723.
[24] 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*. 897–912.
[25] 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 21*. Springer, 317–329.