Effective and Efficient Discovery of Top-k Meta Paths in Heterogeneous Information Networks

Zichen Zhu, Tsz Nam Chan, *Member, IEEE*, Reynold Cheng, *Member, IEEE*, Loc Do, Zhipeng Huang, Haoci Zhang

Abstract—*Heterogeneous information networks (HINs)*, which are typed graphs with labeled nodes and edges, have attracted tremendous interest from academia and industry. Given two HIN nodes s and t, and a natural number k, we study the discovery of the k most important meta paths in real time, which can be used to support friend search, product recommendation, anomaly detection, and graph clustering. In this work, we argue that the shortest path between s and t may not necessarily be the most important path. As such, we combine several ranking functions, which are based on *frequency* and *rarity*, to redefine the unified importance function of the meta paths between s and t. Although this importance function can capture more information, it is very time-consuming to find top-k meta paths using this importance function. Therefore, we integrate this importance function into a multi-step framework, which can efficiently filter some impossible meta paths between s and t. In addition, we combine bidirectional searching algorithm with this framework to further boost the efficiency performance. The experiment on different datasets shows that our proposed method outperforms state-of-the-art algorithms in terms of effectiveness with reasonable response time.

Index Terms—Heterogeneous Information Networks, top-k, meta path

1 Introduction

Given a graph G and two nodes s and t, a *top-k shortest path query* retrieves the k shortest paths between s and t [6]. This query, which enables the retrieval of relationship information between two graph nodes, has been studied extensively in various applications, including bibliographical, social, and road networks. The continuous growth of the sizes of these graphs and the need of online performance drives the development of efficient algorithms and data structures (e.g., [2], [11], [12], [18], [26]). The query is natively supported in graph database engines (e.g., Neo4j [3] and Pregel [13]).

In this paper, we ask the question: is top-k shortest path query the best way for retrieving relationship from a *heterogeneous information network* (or *HIN*)? An HIN is essentially a *typed graph*, whose nodes and edges are tagged with "type labels" to indicate their meanings. Due to its huge amount of information, HINs have recently raised a lot of interest [8], [20], [21], [23], [24]. Figure 1 illustrates an ACM bibliographical network [20], where each node and edge has a type (e.g., Jiawei Han[1] is an *author*; IEEE[2] is an *affiliation*; Jiawei Han *belong to* IEEE). Suppose that we want to know how Philip Yu[3] and Jiawei Han are related in this HIN. A simple way is to evaluate a top-1 shortest path query

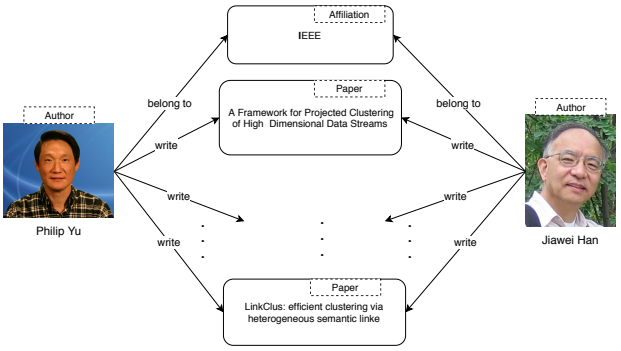


Fig. 1: Shortest paths between *Philip Yu* and *Jiawei Han* in the ACM network.

on it. However, there are plenty of shortest paths (13 in total) between them. Should all these paths be returned to the query user (who may then get a query result with numerous paths)? Is it better to *summarize* the common properties of these paths (e.g., from these shortest paths, we learnt that the two researchers are close collaborators as they have co-authored many papers)?

To address the above questions, we study how to leverage the type information of an HIN to return the k best paths to a query user. Particularly, we propose the *top-k meta path query*, which computes the k best *meta paths* between nodes s and t. A *meta path*, first proposed in [23], is essentially a sequence of node types and edge types. Two possible meta paths, derived from the shortest paths between Philip Yu and Jiawei Han in the HIN of Figure 1, are:

$$\mathcal{P}_1: \quad Author \xrightarrow{belong\ to} Affiliation \xrightarrow{belong\ to^{-1}} Author$$

$$\mathcal{P}_2: \quad Author \xrightarrow{write} Paper \xrightarrow{write^{-1}} Author$$

where $X \xrightarrow{R^{-1}} Y$ means the reverse direction of edge

- *Z. Zhu, R. Cheng and Z. Huang are with the Department of Computer Science, The University of Hong Kong, Hong Kong. E-mail: {zczhu2, ckcheng, zphuang}@cs.hku.hk*
- *T. N. Chan is with the Department of Computer Science, Hong Kong Baptist University, Hong Kong. E-mail: {edisonchan}@comp.hkbu.edu.hk*
- *L. Do is with the Alibaba, China. E-mail: hldo.2012@phdis.smu.edu.sg*
- *H. Zhang is with the Facebook Inc., USA. E-mail: hz2450@columbia.edu*

1. https://hanj.cs.illinois.edu/
2. https://ieeexplore.ieee.org/
3. https://www.cs.uic.edu/~psyu/

R (i.e., $X \xleftarrow{R} Y$). The above meta paths summarize the underlying paths between Philip Yu and Jiawei Han. For example, $\mathcal{P}_1$ states that the two authors belong to the same affiliation (i.e., IEEE), while $\mathcal{P}_2$ is based on the fact they have co-authored some paper. A meta path abstracts the paths and provides important insights about paths. This allows a query user to focus on the high-level relationship patterns, rather than on the detailed path instances. Meta paths have also been shown to be useful in node relevance computation [8], [20], [23], graph clustering [24], and recommendation [21].

To evaluate a top-k meta path query, we need to decide the k *best* meta paths. This is not trivial. Let us again consider the top-1 meta path query for (Philip Yu, Jiawei Han) in Figure 1. Should we choose $\mathcal{P}_1$ or $\mathcal{P}_2$? Should we only consider the k shortest meta paths? To answer the first question, notice that although $\mathcal{P}_1$ and $\mathcal{P}_2$ have the same length, $\mathcal{P}_2$ is in fact *more important* than $\mathcal{P}_1$. This is because Philip Yu and Jiawei Han have co-authored 12 papers, and so $\mathcal{P}_2$ summarizes 12 distinct paths between them. On the other hand, $\mathcal{P}_1$ only represents one path (i.e., *Philip Yu – IEEE – Jiawei Han*). Moreover, we found that most authors in the HIN are registered IEEE members, so $\mathcal{P}_1$ is not a very special relationship between these two researchers. On the other hand, it is relatively uncommon that two authors have co-authored in more than ten papers. Hence, in this scenario, $\mathcal{P}_2$ is arguably a better candidate, compared with $\mathcal{P}_1$, for the result of the top-1 meta path query.

For the second question (i.e., whether a shorter meta path is better), let us consider two other meta paths between Philip and Jiawei.

$$\mathcal{P}_3: \quad Author \xrightarrow{write} Paper \xrightarrow{cite^{-1}} Paper \xrightarrow{write^{-1}} Author$$

$$\mathcal{P}_4: \quad Author \xrightarrow{write} Paper \xrightarrow{cite} Paper \xrightarrow{write^{-1}} Author$$

which depict "citation" relationship (i.e., an author's paper cites another author's work). We found that $\mathcal{P}_3$ and $\mathcal{P}_4$ both represent more than 20 paths, and are much more than the single path that instantiates $\mathcal{P}_1$. Also, $\mathcal{P}_3$ and $\mathcal{P}_4$ are less common among other authors than $\mathcal{P}_1$. Hence, although $\mathcal{P}_1$ has a shorter length than these two meta paths, it may not necessarily be ranked higher than $\mathcal{P}_3$ and $\mathcal{P}_4$.

Our contributions. From the above examples, we observe that in answering a top-k meta path query, we should not only focus on the shortest meta paths, but should also consider other factors (e.g., number of paths pertaining to the meta paths, and the "uniqueness", or *rarity*, of a meta path). In our preliminary work [28], we present a multi-step searching algorithm framework, and propose an *importance function*, which captures several properties of a meta path (e.g., its number of instances (or "support"), degree of commonality, and meta path lengths) for determining its ranking with respect to the query. Because the importance function incorporates existing relevance measures for meta paths, our proposed solution is generic and supports these measures. In this work, we observe that evaluating the proposed importance function in this multi-step framework is still inefficient. Therefore, we further combine the bidirectional searching algorithm with this multi-step framework to further boost the performance. Unlike the existing bidirectional searching method [1], [7] (mainly used for shortest path query), we need to derive new upper bound function which is specialized for our importance function in order to enable efficient bidirectional searching method.

Organization. The rest of this paper is as follows. Section 2 discusses related work. Section 3 formally defines the top-k meta path query. Section 4 lists existing importance function which can be applied for top-k meta path querying and Section 5 introduces (s,t)-dependent importance functions. Section 6 presents a multi-step framework to explore top-k importance-based meta paths and Section 7 illustrates how to integrate bidirectional searching into this framework to further reduce the evaluation time. Section 8 shows our experimental results. We conclude in Section 9.

2 Related Work

There are a wide range of studies to find the suitable meta paths in the heterogeneous information networks (HINs) for different applications which can be divided into two camps.

Learning from predefined meta paths. In this type of work, researchers either aim to learn the best combination [9], [21], [24], [27] of the predefined set of meta paths or propose to reduce the predefined meta path set to a smaller subset [25]. However, as pointed out in [14], it is difficult to have the good predefined meta paths from domain experts, especially for large and complex HINs. On the other hand, unlike our work, this type of work finds the same set of meta-paths for a given HIN, which is independent of the (s,t)-pair. However, it is possible for different (s,t)-pairs to have different top-k meta-paths. In addition, some literatures, e.g., [9], further restrict the maximum length of each meta path to be l. However, it is still unknown how to set the suitable length l in different applications.

Meta path discovery. To mitigate the above drawbacks, some literatures [14], [19] consider how to automatically discover important meta paths in large HINs. Shi et al. [19] propose a method to mine interesting meta paths in a hierarchical HIN. However, it is still unknown whether this method can extend to non-hierarchical HINs. Meng et al. [14] propose the method *Forward Stage-wise Path Generation* (FSPG) to discover meta paths based on some example node pairs. However, there are also two drawbacks of this method. First, it takes large number of node pairs into account in order to obtain meta paths with reasonable accuracy which is time-consuming. Second, it only finds the best meta path from each HIN in which it is possible that different meta paths are suitable for different node pairs (s,t). Compared with above methods, our proposed top-k meta path query can discover the meta paths with any length based on the locality of the node pairs (s,t) in large and complex HINs.

In above discussion, we mainly focus on the effectiveness issues of retrieving top-k meta paths. However, this retrieval process is not efficient, especially for our proposed importance function (cf. Equation 7 in Section 5.3). In existing literatures, many techniques, including Yen's algorithm [6], [26], A* prune algorithm [11], iterative-bounding algorithm [2] and bidirectional search [1], [7], have been developed for efficiently finding paths in the graph. However, all of these methods mainly focus on shortest-path-based problems and do not take the complicated score function (e.g., the importance function in this work) into account and as such, they cannot be directly applied for our top-k meta

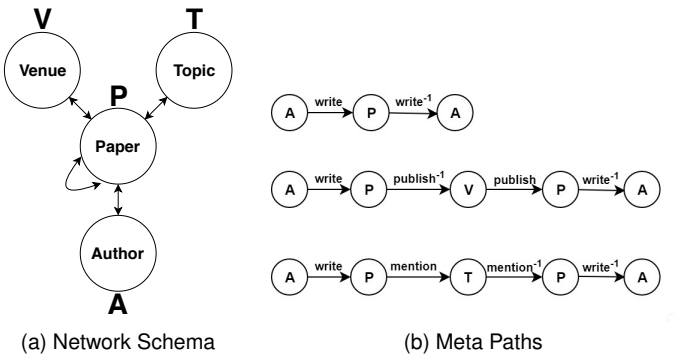


Fig. 2: DBLP dataset

path retrieval problem. In this work, we investigate how to combine bidirectional searching method with our preliminary work [28] to further boost the efficiency performance.

3 PRELIMINARIES

We introduce some terminologies (cf. Definitions 1-3), which are frequently used in this paper, and give a formal definition of the top-k meta path query (cf. Problem 1).

Definition 1. *A Heterogeneous Information Network (HIN) [23] is a directed graph $G = (V, E)$ with an object type mapping $\phi : V \rightarrow \mathcal{A}$ and a link type mapping $\psi : E \rightarrow \mathcal{R}$ subject to $|\mathcal{A}| > 1$ and $|\mathcal{R}| > 1$, where V denotes the object set and $E \subseteq V \times V$ denotes the link set, and $\mathcal{A}$ denotes the object type and $\mathcal{R}$ denotes the link type.*

A typical HIN is the bibliographic network of DBLP [10], as shown in Figure 1. Each node belongs to one type of entities $\mathcal{A}$, which can be author, paper and affiliation in this example. However, HINs can be very large and it is complicated to analyze without the abstract structure, called Network Schema [22], which predefines the possible relationship between different nodes.

Definition 2. *The network schema [22] is a meta template for a heterogeneous information network $G = (V, E)$ with the object type mapping $\phi : V \rightarrow \mathcal{A}$ and the link type mapping $\psi : E \rightarrow \mathcal{R}$, which is a directed graph defined over objects types $\mathcal{A}$, with edges as relations from $\mathcal{R}$, noted as $T_G = (\mathcal{A}, \mathcal{R})$.*

Figure 2a shows the network schema of DBLP dataset [10]. There are four types of entities: Paper, Venue, Author and Topic. This network schema defines the possible connection between different nodes in the HIN. For example: node with type Author can connect with node with type Paper which denotes the "writing" or "written-by" relations. In addition, node with type Venue cannot directly connect to node with type Topic. Once we have the network schema, we can define the concept of meta path based on it.

Definition 3. *A meta path $\mathcal{P}$ [23] is a path defined on the network schema $T_G = (\mathcal{A}, \mathcal{R})$, and is denoted in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} A_{l+1}$. If there is no ambiguity, we can shorten it as $R = R_1 \circ R_2 \circ \cdots \circ R_l$ between $A_1, A_2, \cdots, A_{l+1}$, where $\circ$ denotes the concatenation operator on relations.*

Figure 2b shows three common meta paths between two authors in DBLP dataset, which are (1) two authors wrote the same paper together, (2) two authors published some paper in the same venue and (3) two authors published some paper with the same topic. Here, we denote $Author \xrightarrow{write} Paper \xrightarrow{publish^{-1}} Venue \xrightarrow{publish} Paper \xrightarrow{write^{-1}} Author$ as the meta path of (2). Based on above definitions, we formally define the top-k meta path query.

Problem 1 (Top-k meta path query)**.** *Given an HIN $G = (V, E)$, two nodes (s, t) where $s, t \in V$ and $\phi(s), \phi(t) \in \mathcal{A}$, we define the importance function $I_{s,t}(\mathcal{P})$, which denotes the importance of a meta path $\mathcal{P}$ between a source node s and a target node t, where $I_{s,t}(\mathcal{P}) \geq 0$. This problem aims to find k meta paths, where each meta path $\mathcal{P}$ has the top-k importance value $I_{s,t}(\mathcal{P})$.*

Different importance functions $I_{s,t}(\mathcal{P})$ have been developed in existing literatures [21], [27]. However, they normally cannot provide accurate top-k result. As such, we will discuss how to develop accurate importance function in later section.

In this paper, we use lower-case letters (e.g., a_i) to denote objects in an HIN and upper-case letters (e.g., A) to denote object types in T_G. We say a concrete path $p = (a_1 a_2 \cdots a_{l+1})$ in HIN G follows the meta path $\mathcal{P}$ (or p is a **path instance** of the meta path $\mathcal{P}$), if $\forall a_i \in V, \phi(a_i) = A_i$ and $\forall e_i = \langle a_i, a_{i+1} \rangle \in E, \psi(e_i) = R_i$ in $\mathcal{P}$. Mathematically, we also denote it as $p \in \mathcal{P}$. In addition, if p is the path instance with source node s and target node t and $p \in \mathcal{P}$, we denote it as $s \xrightarrow{\mathcal{P}} t$.

4 EXISTING IMPORTANCE FUNCTIONS FOR TOP-K META PATHS RETRIEVAL

In this section, we review several importance functions in existing work, for example: length-penalty-based importance function [14], [21], [23], [27] and strength-based importance function [21], [27].

4.1 Length-Penalty-based Importance Functions

The simplest way for measuring the importance for a given meta path is based on its length. Sun et al. [23] experimentally show that a long meta path is not necessarily useful to determine the similarity between two objects in an HIN. Based on this observation, some length-penalty-based importance functions, which we term all of them as $Penalty(\mathcal{P})$, have been proposed for finding the meta paths [14], [21], [27], as shown in Table 1. These penalty functions are strictly decreasing with respect to the length $|\mathcal{P}|$.

TABLE 1: Length-penalty-based importance functions

$Penalty(\mathcal{P})$	Ref.
$e^{-\|\mathcal{P}\|}$	[21]
$\beta^{\|\mathcal{P}\|}$ $(0 < \beta < 1)$	[14]
$\frac{1}{\|\mathcal{P}\|}$	[27]

4.2 Strength-based Importance Functions

In large and complex HINs, it is common to have many meta paths with same length. Using Figure 3 as a toy example,

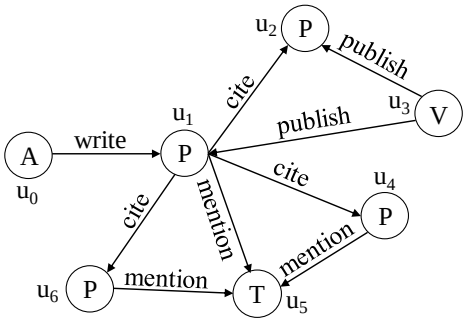


Fig. 3: Bibliographical network example, following the HIN schema in Figure 2a, which is used to illustrate the concepts of different importance functions

there are many length-1 meta paths, e.g., V $\xrightarrow{\text{publish}}$ P, P $\xrightarrow{\text{mention}}$ T.

As such, length-penalty-based importance function cannot distinguish which paths are more important in this case. Therefore, except for considering only the length of the meta paths, existing literatures [21], [27] also take the topological structure, e.g., the in-degree and out-degree of node types, into account and define the new importance function $Strength(\mathcal{P})$. Mathematically, if $\mathcal{P} = \text{A} \xrightarrow{\text{R}} \text{B}$, where A and B are two node types, the strength function is:

$$Strength(A \xrightarrow{R} B) = \frac{1}{\sqrt{OD(A \xrightarrow{R} B) \times ID(A \xrightarrow{R} B)}} \quad (1)$$

where $OD(A \xrightarrow{R} B)$ is the average out-degree of A-type nodes, and $ID(A \xrightarrow{R} B)$ is the average in-degree of B-type nodes on the relation type R from A to B. Using V $\xrightarrow{\text{publish}}$ P in Figure 3 as an example, there is only one node V in this HIN, and there are two edges with link "publish" connecting to it. Therefore, the average out-degree $OD(\text{V} \xrightarrow{\text{publish}} \text{P}) = \frac{2}{1} = 2$. Similarly, there are two nodes P and each node has one link "publish" which connects from the node V. Therefore, the average in-degree $ID(\text{V} \xrightarrow{\text{publish}} \text{P}) = \frac{2}{2} = 1$.

If $\mathcal{P}$ contains more than one relations, i.e., $\mathcal{P} = A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} A_{l+1}$, we can extend Equation 1 to the general form.

$$Strength(\mathcal{P}) = \prod_{i=1}^{l} Strength(A_i \xrightarrow{R_i} A_{i+1}) \quad (2)$$

5 (s,t)-DEPENDENT IMPORTANCE FUNCTIONS

Existing meta path discovery methods (cf. Section 4) find the meta paths which is independent of the source-target (s,t)-pairs. However, it is possible that different meta paths are suitable for different (s,t) pairs. In this section, we develop two new importance functions, namely rarity and path count, which take (s,t)-pair into account.

5.1 Rarity Function

As discussed in Section 1, some paths are very common which cannot provide any new information. For example: many authors in ACM network are also the members in IEEE. Therefore, it is not meaningful to return the common meta path Author $\xrightarrow{\text{belong to}}$ Affiliation $\xrightarrow{\text{belong to}^{-1}}$ Author rather than some other rare meta paths, e.g., Author $\xrightarrow{\text{write}}$ Paper $\xrightarrow{\text{write}^{-1}}$ Author. In order to capture this effect, we develop the rarity function $\mathcal{R}_{s,t}(\mathcal{P})$ which evaluates how rare the meta path $\mathcal{P}$ for (s,t) pair in a given HIN $G = (V,E)$. Before we formally define rarity function, we first define how to select those similar pairs (noted by $SIM_{s,t}$) as follows:

$$SIM_{s,t} = \{(s,v)|\phi(v) = \phi(t)\} \cup \{(v,t)|\phi(v) = \phi(s)\} \quad (3)$$

Using Figure 3 as an example, given the source node $s = u_0$ and the target node $t = u_2$, we have $SIM_{s,t} = \{(u_0,u_1),(u_0,u_2),(u_0,u_4),(u_0,u_6)\}$ based on the above definition. After we get $SIM_{s,t}$, we can evaluate the rarity for a given meta path $\mathcal{P}$ as follows[4]:

$$\mathcal{R}_{s,t}(\mathcal{P}) = \log\Big(1 + \frac{|SIM_{s,t}|}{|\{(u,v) \in SIM_{s,t}, u \xrightarrow{\mathcal{P}} v\}|}\Big) \quad (4)$$

As an example, we consider the following two meta paths for the above (s,t)-pair ($s = u_0$ and $t = u_2$):

$$\mathcal{P}_5: \quad \text{Author} \xrightarrow{\text{write}} \text{Paper} \xrightarrow{\text{cite}} \text{Paper}$$
$$\mathcal{P}_6: \quad \text{Author} \xrightarrow{\text{write}} \text{Paper} \xrightarrow{\text{publish}^{-1}} \text{Venue} \xrightarrow{\text{publish}} \text{Paper}$$

We obtain the rarity values for these two paths, $\mathcal{R}_{s,t}(\mathcal{P}_5) = \log(1+\frac{4}{3}) = 0.8473$ and $\mathcal{R}_{s,t}(\mathcal{P}_6) = \log(1+\frac{4}{2}) = 1.0986$. Since $\mathcal{P}_6$ is rarer compared with meta path $\mathcal{P}_5$ for (u_0,u_2)-pair in Figure 3, it is natural to assign larger rarity value for meta path $\mathcal{P}_6$.

5.2 Path Count Function

Recall from Figure 1 (in Section 1), Jiawei Han and Philip Yu have coauthored more than ten papers which are very uncommon for other author pairs. As such, it is more suitable to use the meta path Author $\xrightarrow{\text{write}}$ Paper $\xrightarrow{\text{write}^{-1}}$ Author to describe the relationship between Jiawei Han and Philip Yu. Unfortunately, none of the existing importance functions (cf. Section 4) can capture this information for the given meta path. Therefore, we utilize path count [8] as another importance function, for the meta path $\mathcal{P}$ between the node pair (s,t), denoted as $PC_{s,t}(\mathcal{P})$. As an example, we consider the following meta path $\mathcal{P}_7$:

$$\mathcal{P}_7: \quad \text{Author} \xrightarrow{\text{write}} \text{Paper} \xrightarrow{\text{cite}} \text{Paper} \xrightarrow{\text{mention}} \text{Topic}$$

In Figure 3, since there are two paths from u_0 to u_5 which are identical to this meta path $\mathcal{P}_7$, we have $PC_{u_0,u_5}(\mathcal{P}_7) = 2$. However, there is one major drawback for using $\mathcal{PC}_{s,t}(\mathcal{P})$ as an importance function. We consider the following meta paths (We omit relation type to save space):

$$\mathcal{P}_8: \quad A \to P \to A$$
$$\mathcal{P}_9: \quad A \to P \to A \to P \to A \to P \to A$$

4. The definition of this rarity function for a meta path $\mathcal{P}$ is inspired by the inverse document frequency (**IDF**) calculation in TF-IDF model [16]

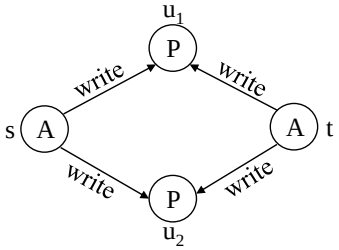


Fig. 4: Illustration of the infeasibility for using $\mathcal{PC}_{s,t}(\mathcal{P})$ in this HIN

Although both two meta paths can connect s and t in Figure 4, we can notice that $PC_{s,t}(\mathcal{P}_8) = 2$ and $PC_{s,t}(\mathcal{P}_9) = 8$. If we append another $\rightarrow P \rightarrow A \rightarrow P \rightarrow A$ to the original meta path (e.g., $\mathcal{P}_8$), we can achieve 4 times more of the original path count value, as shown in Figure 5 for meta path $\mathcal{P}_9$. Therefore, it is possible for the path count value to be very large once we increases the length of meta path.

	A → P → A → P → A → P → A						
p_1	s	u_1	t	u_1	s	u_1	t
p_2	s	u_1	t	u_1	s	u_2	t
p_3	s	u_1	t	u_2	s	u_1	t
p_4	s	u_1	t	u_2	s	u_2	t
p_5	s	u_2	t	u_1	s	u_1	t
p_6	s	u_2	t	u_1	s	u_2	t
p_7	s	u_2	t	u_2	s	u_1	t
p_8	s	u_2	t	u_2	s	u_2	t

Fig. 5: Path count for $\mathcal{P}_9$ ($PC_{s,t}(\mathcal{P}_9) = 8$)

Instead of counting all path instances (e.g., p_1 to p_8 of $\mathcal{P}$ in Figure 5), we only choose MNI, the minimum number of node instances a_i for each attribute type A_i in the meta path $\mathcal{P} = A_1 \rightarrow A_2 ... \rightarrow A_{l+1}$, except A_1 and A_{l+1}.

$$MNI_{s,t}(\mathcal{P}) = \begin{cases} \min\limits_{\substack{1<i\leq l \\ a_1=s, a_{l+1}=t}} |\{a_i | p \in \mathcal{P}, p = (a_1, a_2, ..., a_{l+1})\}| & \text{if } l > 1 \\ 1 & \text{if } l = 1 \end{cases} \quad (5)$$

As an example, the total node instances for the 1^{st} P in $\mathcal{P}_9$ are both u_1 and u_2 (cf. column 2 in Figure 5). Therefore, the minimum number of node instances is $\min(2, 1, 2, 1, 2) = 1$. We omit the first and last node types because the first and last node instances must be s and t respectively.

However, there is still one drawback for $MNI_{s,t}(\mathcal{P})$. We consider the following meta paths.

$$\mathcal{P}_{10} : \text{Paper} \xrightarrow{\text{mention}} \text{Term} \xrightarrow{\text{mention}^{-1}} \text{Paper}$$
$$\mathcal{P}_{11} : \text{Paper} \xrightarrow{\text{publish}^{-1}} \text{Venue} \xrightarrow{\text{publish}} \text{Paper}$$

Using Figure 6 as an example, we notice that the paper s mentions three different terms which are also mentioned by the paper t in the left HIN network. However, since each paper can mention several terms (Paper $\xrightarrow{\text{mention}}$ Term), it is easy to achieve high MNI value for meta path $\mathcal{P}_{10}$. In this example, $MNI_{s,t}(\mathcal{P}_{10}) = 3$. On the other hand, the paper s is published in v_1 where t is also published in v_1 in the right HIN. Since each paper can be only published in one venue, the MNI value of Paper $\xrightarrow{\text{publish}^{-1}}$ Venue $\xrightarrow{\text{publish}}$ Paper is normally low. In this example, $MNI_{s,t}(\mathcal{P}_{11}) = 1$. To get rid of the bias in Equation 5, we penalize relations like P_{10} using Strength-based importance function (cf. Equation 2). Then we can redefine the new path count function (cf. Equation 6) as follows.

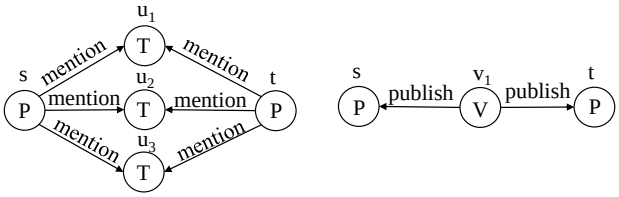


Fig. 6: $MNI_{s,t}(\mathcal{P})$ is biased to common relations, e.g., Paper $\xrightarrow{\text{mention}}$ Term and Paper $\xrightarrow{\text{publish}^{-1}}$ Venue.

$$PC^{new}_{s,t}(\mathcal{P}) = MNI_{s,t}(\mathcal{P}) \times Strength(\mathcal{P}) \quad (6)$$

5.3 Unified Importance Function (MNIS)

As discussed in previous sections, different importance functions can capture different information, for example: length (cf. Section 4.1), rarity (cf. Section 5.1) and path count (cf. Section 5.2). If we can develop an unified importance function which can capture all these information, it is possible to retrieve more accurate top-k meta paths (cf. Problem 1) for each (s, t)-pair compared with only using single importance function. To capture all the advantages of previous importance functions, we cascade them in the following way (cf. Equation 7). We term this importance function as strengthened-MNI-support (MNIS) [28].

$$I_{s,t}(\mathcal{P}) = \begin{cases} Penalty(\mathcal{P}) \times \mathcal{R}_{s,t}(\mathcal{P}) \times PC^{new}_{s,t}(\mathcal{P}), & \text{if } s \xrightarrow{\mathcal{P}} t \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

As a remark, this importance function exhibits the **symmetric property** (which will be used in Section 7.2). Let $\mathcal{P}^{-1}$ be the reverse meta path of P, e.g., $\mathcal{P}_5^{-1}$ = Paper $\xrightarrow{\text{cite}^{-1}}$ Paper $\xrightarrow{\text{write}^{-1}}$ Author, we notice that $I_{s,t}(\mathcal{P}) = I_{t,s}(\mathcal{P}^{-1})$.

6 Top-k Meta Path Retrieval via Multi-Step Searching Framework

Even though the above unified importance function $I_{s,t}(\mathcal{P})$ can capture more information, and thus more accurate, retrieving top-k important meta paths (cf. Problem 1) with this importance function is very inefficient. Theoretically, there are infinite meta paths (keep appending more nodes) between s and t. As such, it is impossible to enumerate all these meta paths. On the other hand, some meta paths with extremely long length are not in the top-k set. Therefore, by combining the upper bound function and existing multi-step searching framework [17], we can filter least potential meta path candidates for the top-k query.

6.1 Multi-step searching

Figure 7 illustrates the searching process of multi-step framework (cf. Algorithm 1). Initially, the algorithm starts at the initial source node s with its node type (A in this example). In each iteration, new meta paths will be generated by appending connected edge types. As an example, edges with types "write" and "'belong to" are connected to source node s (with node type A). Therefore, two meta paths $\mathcal{P}_1 = \text{A} \xrightarrow{\text{write}} \text{P}$ and $\mathcal{P}_2 = \text{A} \xrightarrow{\text{belong to}} \text{F}$ are generated. For each generated meta path $\mathcal{P}$, we denote the set of meta paths with prefix $\mathcal{P}$ as $PRE_{\mathcal{P}}$ (e.g., $\mathcal{P}_1$, $\mathcal{P}_3, \mathcal{P}_5 \in PRE_{\mathcal{P}_1}$), the multi-step framework evaluates its upper bound value for all meta paths in $PRE_{\mathcal{P}}$, denoted as $UB(I_{s,t}(PRE_{\mathcal{P}}))$. In the whole searching process, the multi-step framework maintains a priority queue (cf. Figure 8a) from which the meta path (e.g., $\mathcal{P}_1$) with the highest upper bound value (e.g., $UB(I_{s,t}(PRE_{\mathcal{P}_1}))$) will be selected and expanded into new meta paths (e.g., $\mathcal{P}_3$ to $\mathcal{P}_5$) in each iteration. Once the expanded meta path reaches the target node t, the multi-step algorithm evaluates the exact importance function (cf. Equation 7). As an example, let t be u_4, this algorithm evaluates Equation 7 once $\mathcal{P}_5$ is reached. The multi-step algorithm terminates once the highest upper bound value is still smaller than the best known smallest top-k exact importance values τ (cf. $UB(I_{s,t}(PRE_{\mathcal{P}_8}))$ in Figure 8b).

In the following subsections (Sections 6.2, 6.3 and 6.4), we will discuss the upper bound functions for different components in the importance function (cf. Equation 7) for the set $PRE_{\mathcal{P}}$ of meta paths, where:

$$\begin{aligned} &UB(I_{s,t}(PRE_{\mathcal{P}})) = UB(Penalty(PRE_{\mathcal{P}})) \\ &\times UB(R_{s,t}(PRE_{\mathcal{P}})) \times UB(PC^{new}_{s,t}(PRE_{\mathcal{P}})) \end{aligned} \tag{8}$$

6.2 Upper bound $UB(Penalty(PRE_{\mathcal{P}}))$

Recall from Table 1, the penalty function is monotonic decreasing with the length of the meta path. Since all the meta paths in $PRE_{\mathcal{P}}$ have the prefix meta path $\mathcal{P}$, they are longer than $\mathcal{P}$. We can conclude:

$$UB(Penalty(PRE_{\mathcal{P}})) = Penalty(\mathcal{P}) \tag{9}$$

6.3 Upper bound $UB(\mathcal{R}_{s,t}(PRE_{\mathcal{P}}))$

One trivial upper bound of the rarity function (cf. Equation 4) for all meta paths in $PRE_{\mathcal{P}}$ is $\max_{\widehat{\mathcal{P}} \in PRE_{\mathcal{P}}} \mathcal{R}_{s,t}(\widehat{\mathcal{P}})$. However, since the set $PRE_{\mathcal{P}}$ contains infinite meta paths, directly evaluating this upper bound is computationally infeasible. Observe from Equation 4, we notice that only the denominator term $|\{(u,v) \in SIM_{s,t}, u \xrightarrow{\mathcal{P}} v\}|$ depends on the meta path $\mathcal{P}$, once we can develop efficient lower bound for this term, which is not dependent on $\mathcal{P}$, we can obtain the efficient upper bound $UB(\mathcal{R}_{s,t}(PRE_{\mathcal{P}}))$ as:

$$UB(\mathcal{R}_{s,t}(PRE_{\mathcal{P}})) = \log\left(1 + \frac{|SIM_{s,t}|}{lb}\right) \tag{10}$$

One trivial way for the lower bound of $|\{(u,v) \in SIM_{s,t}, u \xrightarrow{\mathcal{P}} v\}|$ is the value $lb = 1$ [28], since we have at least one pair $(s,t) \in SIM_{s,t}$ which is also connected by the meta path $\mathcal{P}$. Here, we assume the $\mathcal{P}$ should connect between (s,t) pair since the importance function is 0 once it is not connected (cf. Equation 7). In Appendix (cf. Section 10.1), we discuss how to obtain the tighter lb, which can achieve tighter upper bound value $UB(\mathcal{R}_{s,t}(PRE_{\mathcal{P}}))$.

Algorithm 1 Top-K Important Meta Paths Discovery

Input: Network G, two nodes $s, t \in G$ and the number k
Output: Top-k Important Meta Paths between s and t

create a max-heap H
create a min-heap H_{result}
$m_{root} \leftarrow \{objs : \{s\}, \mathcal{P} : \text{Empty Path}, ub : \infty\}$
$H.push(m_{root})$
while $H.size \neq 0$ **do**
  $m \leftarrow H.pop()$
  if $|H_{result}| = k$ **then**
    $m_{small} \leftarrow H_{result}.top()$
    if $m.ub < m_{small}.score$ **then**
      $break$
    end if
  end if
  $pathToObjs \leftarrow$ Empty Map
  for each object node $u \in m.objs$ **do**
    for each neighbor node v of u **do**
      $\mathcal{P}_{tmp} \leftarrow m.\mathcal{P} \circ \psi(uv)$
      if $\mathcal{P}_{tmp}$ not in $pathToObjs$ **then**
        $pathToObjs[\mathcal{P}_{tmp}] \leftarrow$ Empty Set
      end if
      $pathToObjs[\mathcal{P}_{tmp}].insert(v)$
    end for
  end for
  for each pair $(\mathcal{P}, objs) \in pathToObjs$ **do**
    $m_{tmp} \leftarrow \{objs, \mathcal{P}, \infty\}$
    if $t \in objs$ **then**
      $m_{tmp}.score \leftarrow I_{s,t}(\mathcal{P})$
      $m_{small} \leftarrow H_{result}.top()$
      if $m_{small}.score \leq m_{tmp}.score$ and $|H_{result}| = k$ **then**
        $H_{result}.pop()$
      end if
      $H_{result}.push(m_{tmp})$
    else
      $m_{tmp}.ub \leftarrow UB(I_{s,t}(\mathcal{P}))$
    end if
    H.push(m_{tmp})
  end for
end while
return H_{result}

6.4 Upper bound $UB(\mathcal{PC}^{new}_{s,t}(PRE_{\mathcal{P}}))$

Recall from Equation 6, this modified path count function depends on two components, which are $MNI_{s,t}(\mathcal{P})$ (cf. Equation 5) and $Strength(\mathcal{P})$ (cf. Equation 2). In the following, we discuss how to efficiently obtain both $UB(MNI_{s,t}(PRE_{\mathcal{P}}))$ and $UB(Strength(PRE_{\mathcal{P}}))$ and achieve:

$$\begin{aligned} &UB(\mathcal{PC}^{new}_{s,t}(PRE_{\mathcal{P}})) \\ &= UB(MNI_{s,t}(PRE_{\mathcal{P}})) \times UB(Strength(PRE_{\mathcal{P}})) \end{aligned} \tag{11}$$

One simple upper bound for $UB(MNI_{s,t}(PRE_{\mathcal{P}}))$ is $\max_{\widehat{\mathcal{P}} \in PRE_{\mathcal{P}}} MNI_{s,t}(\widehat{\mathcal{P}})$. However, as stated in Section 6.3, there can be infinite number of meta paths in the set $PRE_{\mathcal{P}}$. Therefore, directly evaluating $\max_{\widehat{\mathcal{P}} \in PRE_{\mathcal{P}}} MNI_{s,t}(\widehat{\mathcal{P}})$ is infeasible. However, based on the property of Equation 5, we can derive the following upper bound function (cf. Equation 12):

$$\begin{aligned} &UB(MNI_{s,t}(PRE_{\mathcal{P}})) \\ &= \min_{\substack{1<i\leq|P| \\ a_1=s}} |\{a_i | p \in \mathcal{P}, p = (a_1, a_2, ..., a_{|P|+1})\}| \end{aligned} \tag{12}$$

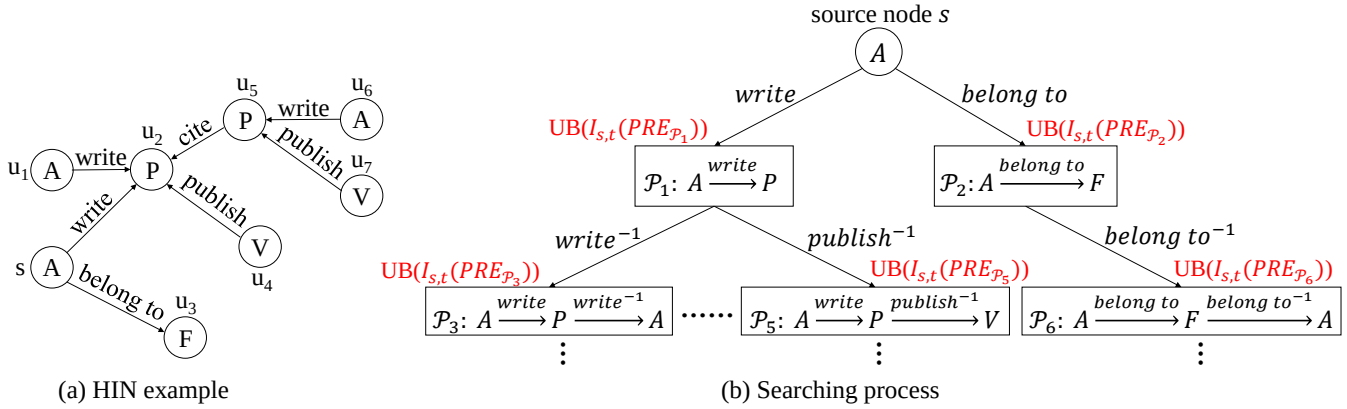


Fig. 7: Multi-step approach to explore the new meta paths from the source node s to the target node $t = u_4$ in a given HIN, each generated meta path (rectangle) is augmented with the upper bound value for all meta paths which use this meta path as prefix (i.e., descendants)

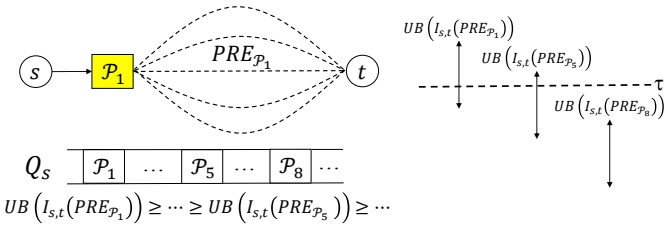


Fig. 8: Mechanism for multi-step framework

Note that this upper bound function only depends on $\mathcal{P}$, instead of $PRE_{\mathcal{P}}$, which can be computed efficiently. We leave the derivation of this upper bound function in Appendix (cf. Section 10.2).

We proceed to discuss how to achieve the upper bound function $UB(Strength(PRE_{\mathcal{P}}))$. Recall from Equation 1, we can notice that $Strength(A \xrightarrow{R} B) \leq 1$. As such, the strength function (cf. Equation 2) is also the monotonic decreasing function, with respect to the length. Based on the similar concept of Section 6.2, we have the following upper bound function (cf. Equation 13):

$$UB(Strength(PRE_{\mathcal{P}})) = Strength(\mathcal{P}) \tag{13}$$

We formally derive this upper bound function in Appendix (cf. Section 10.3).

7 Bidirectional Searching

Even though the multi-step searching framework (cf. Section 6.1) and existing upper bound functions (cf. Sections 6.2-6.4) can enable the retrieval of top-k meta paths, given the (s,t)-pair. The evaluation time is still very inefficient. As will be discussed in the experimental section, evaluating the top-k meta path query for only one (s,t)-pair takes nearly 13 minutes in ACM dataset using the multi-step searching framework. Therefore, there is a need to further speedup the response time.

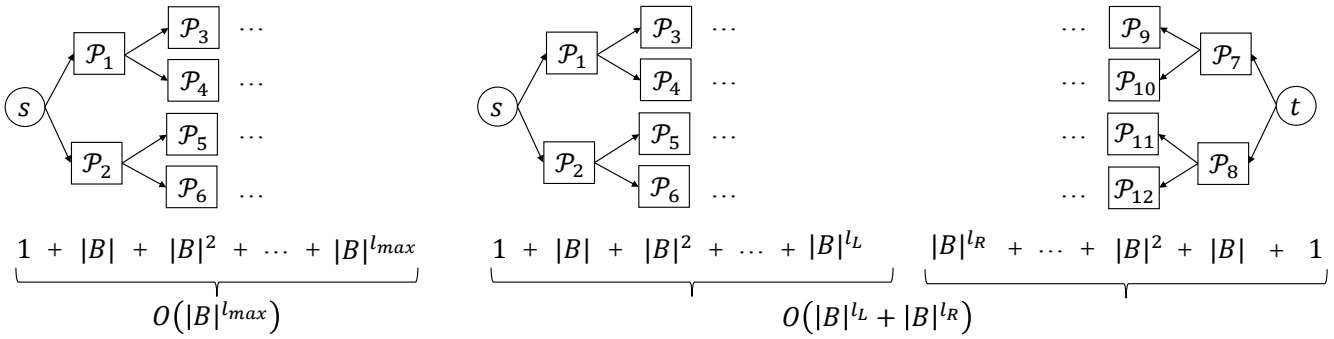
We notice that many existing literatures [1], [7] and graph databases such as Neo4j [3] use bidirectional breadth first search to further boost the efficiency performance of evaluating shortest path query. However, all these existing work do not take the scoring function (importance function (cf. Equation 7)) into account and as such, their methods cannot directly be applied to our top-k meta path query (cf. Problem 1). Therefore, one natural question is how can we incorporate the bidirectional searching method with the multi-step searching framework to further boost the efficiency performance? In this section, we first review the benefit of bidirectional searching (cf. Section 7.1). Then, we discuss how to integrate this bidirectional searching method into the multi-step framework to further boost the efficiency performance for the top-k meta path query (cf. Section 7.2). After that, we perform the case study on the tightness of bound functions for both uni-directional search and bi-directional search in Section 7.3.

7.1 Why do we need to use bidirectional searching?

Recall from Figure 7b, the multi-step searching process follows a tree-like structure to discover new meta paths in each iteration, which follows the unidirectional searching process (cf. Figure 9a). However, instead of only searching from the source node s, if we can also search from the target node t, it is possible to further reduce the searching cost (cf. Figure 9b). To simplify the analysis, we denote the branching factor of the tree to be $|B|$ (which can be estimated by the average degree in the HIN) and the maximum length of the meta paths that the multi-step algorithm discovers to be l_{max}. As shown in Figure 9a, we model the time and the space cost of the unidirectional searching to be $O(|B|^{l_{max}})$, which can be much larger than the cost ($O(|B|^{l_L} + |B|^{l_R})$, where l_L and l_R are smaller than l_{max}) of the bidirectional searching (cf. Figure 9b).

7.2 How can we extend bidirectional searching for top-k meta path query (Problem 1)?

In bidirectional searching method (cf. Figure 10), we need to maintain two priority queues Q_s and Q_t for both the source node s and target node t respectively. Each entry in the priority queues maintains the upper bound value. For example: let $\mathcal{P}_L$ and $\mathcal{P}_R$ be any two meta paths (entries in priority queues) which are stored in Q_s and Q_t respectively, the



(a) Unidirectional searching (b) Bidirectional searching

Fig. 9: Unidirectional Searching vs Bidirectional Searching (Branching factor: $|B|$)

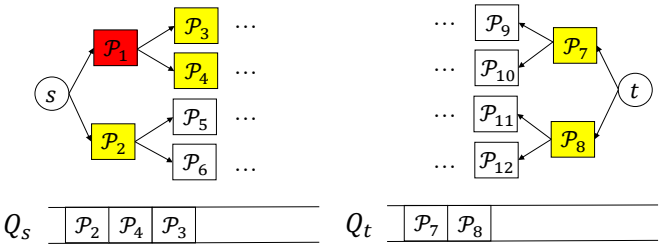


Fig. 10: Combination of multi-step framework and bidirectional searching (Yellow nodes have been explored by the algorithm and kept in the priority queues, and red nodes have been popped out of the priority queues)

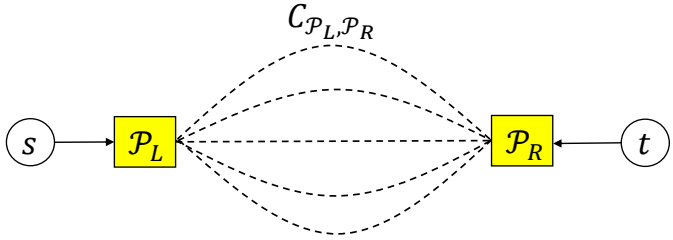


Fig. 11: The set $C_{\mathcal{P}_L,\mathcal{P}_R}$ of meta paths with prefix $\mathcal{P}_L$ (from s) and prefix $\mathcal{P}_R$ (from t)

upper bounds for these two entries are $UB(I_{s,t}(PRE_{\mathcal{P}_L}))$ and $UB(I_{t,s}(PRE_{\mathcal{P}_R}))$. Notice that $UB(I_{t,s}(PRE_{\mathcal{P}_R}))$ can also act as the upper bound function due to the symmetric property of $\mathcal{I}_{s,t}(\mathcal{P})$ (cf. Section 5.3).

Similar to the existing bidirectional search, we can obtain the new upper bound value based on these two bound values ($UB(I_{s,t}(PRE_{\mathcal{P}_L}))$ and $UB(I_{t,s}(PRE_{\mathcal{P}_R}))$). For example, the simplest way for obtaining the proper upper bound is to utilize the following equation.

$$\min(UB(I_{s,t}(PRE_{\mathcal{P}_L})), UB(I_{t,s}(PRE_{\mathcal{P}_R}))) \quad (14)$$

However, this upper bound turns out to be loose as it does not consider the properties of different components of the importance function and it regards $PRE_{\mathcal{P}_L}$ and $PRE_{\mathcal{P}_R}$ as two independent sets of prefix paths. Therefore, bidirectional searching method cannot achieve any speedup (or can be even slower) by adopting this upper bound function (cf. Equation 14). In the following, we discuss how to obtain the tighter upper bound function.

Tighter upper bound function: To develop the tighter upper bound function, we need to analyze the upper bound functions for different components in the importance function. First, we denote $C_{\mathcal{P}_L,\mathcal{P}_R}$ (cf. Figure 11) to be the set of meta paths which have the prefix $\mathcal{P}_L$ (from s) and prefix $\mathcal{P}_R$ (from t).

Lemma 1. *The upper bounds in different components (Penalty, Rarity and Path Count) for the set $C_{\mathcal{P}_L,\mathcal{P}_R}$ are denoted by:*

$$\begin{aligned}
UB(Penalty(C_{\mathcal{P}_L,\mathcal{P}_R})) &= Penalty(\mathcal{P}_L) \times Penalty(\mathcal{P}_R) \\
UB(R_{s,t}(C_{\mathcal{P}_L,\mathcal{P}_R})) &= \min \begin{cases} UB(R_{s,t}(PRE_{\mathcal{P}_L})), \\ UB(R_{t,s}(PRE_{\mathcal{P}_R})) \end{cases} \\
UB(PC^{new}_{s,t}(C_{\mathcal{P}_L,\mathcal{P}_R})) &= Strength(\mathcal{P}_L) \times Strength(\mathcal{P}_R) \\
&\quad \times \min \begin{cases} UB(MNI_{s,t}(PRE_{\mathcal{P}_L})), \\ UB(MNI_{t,s}(PRE_{\mathcal{P}_R})) \end{cases}
\end{aligned}$$

Proof. In this proof, we only focus on these two bound functions $UB(Penalty(C_{\mathcal{P}_L,\mathcal{P}_R}))$ and $UB(R_{s,t}(C_{\mathcal{P}_L,\mathcal{P}_R}))$, the similar concept can be further applied for $UB(PC^{new}_{s,t}(C_{\mathcal{P}_L,\mathcal{P}_R}))$.

As shown in Figure 11, the meta paths in $C_{\mathcal{P}_L,\mathcal{P}_R}$ contain $\mathcal{P}_L$ and $\mathcal{P}_R$ as prefix from s and t respectively. Therefore, for every meta path $\mathcal{P}$ in $C_{\mathcal{P}_L,\mathcal{P}_R}$, we have:

(1) the meta path $\mathcal{P}$ is longer than the sum of length of $\mathcal{P}_L$ and $\mathcal{P}_R$. As such, we have the above $UB(Penalty(C_{\mathcal{P}_L,\mathcal{P}_R}))$ (cf. Table 1).

(2) $C_{\mathcal{P}_L,\mathcal{P}_R} \subseteq PRE_{\mathcal{P}_L}$. As such, we have:

$$UB(R_{s,t}(C_{\mathcal{P}_L,\mathcal{P}_R})) \leq UB(R_{s,t}(PRE_{\mathcal{P}_L}))$$

Similarly, we have:

$$UB(R_{s,t}(C_{\mathcal{P}_L,\mathcal{P}_R})) \leq UB(R_{t,s}(PRE_{\mathcal{P}_R}))$$

By combining these two upper bounds, we have the above $UB(R_{s,t}(C_{\mathcal{P}_L,\mathcal{P}_R}))$. □

Based on Lemma 1, we can achieve the following upper bound for importance function in the set $C_{\mathcal{P}_L,\mathcal{P}_R}$.

Lemma 2. *The upper bound for importance function in the set $C_{\mathcal{P}_L,\mathcal{P}_R}$ is denoted as $UB(I_{s,t}(C_{\mathcal{P}_L,\mathcal{P}_R}))$, where:*

$$\begin{aligned}
&UB(I_{s,t}(C_{\mathcal{P}_L,\mathcal{P}_R})) \\
= &\min \begin{cases} UB(I_{s,t}(PRE_{\mathcal{P}_L})) \times Strength(\mathcal{P}_R) \times Penalty(\mathcal{P}_R), \\ UB(I_{t,s}(PRE_{\mathcal{P}_R})) \times Strength(\mathcal{P}_L) \times Penalty(\mathcal{P}_L) \end{cases}
\end{aligned}$$

Proof. We first prove this inequality $UB(I_{s,t}(C_{\mathcal{P}_L,\mathcal{P}_R})) \leq UB(I_{s,t}(PRE_{\mathcal{P}_L})) \times Strength(\mathcal{P}_R) \times Penalty(\mathcal{P}_R)$ and we can easily extend to the other part. The upper bound for the importance function (cf. Equation 7) in the set $C_{\mathcal{P}_L,\mathcal{P}_R}$ is:

$$\begin{aligned}
&UB(I_{s,t}(C_{\mathcal{P}_L,\mathcal{P}_R})) \\
= &\ UB(Penalty(C_{\mathcal{P}_L,\mathcal{P}_R})) \times UB(R_{s,t}(C_{\mathcal{P}_L,\mathcal{P}_R})) \times \\
&\ UB(PC^{new}_{s,t}(C_{\mathcal{P}_L,\mathcal{P}_R}))
\end{aligned}$$

By Lemma 1, we have:

$$\begin{aligned} & UB(I_{s,t}(C_{\mathcal{P}_L,\mathcal{P}_R})) \\ \leq\ & Penalty(\mathcal{P}_L) \times Penalty(\mathcal{P}_R) \times UB(R_{s,t}(PRE_{\mathcal{P}_L})) \times \\ & Strength(\mathcal{P}_L) \times Strength(\mathcal{P}_R) \times UB(MNI_{s,t}(PRE_{\mathcal{P}_L})) \end{aligned}$$

Recall from the upper bound function (cf. Equation 8) for the set $PRE_{\mathcal{P}}$ of meta paths and the upper bound (cf. Sections 6.2-6.4) for each component. We can conclude $UB(I_{s,t}(C_{\mathcal{P}_L,\mathcal{P}_R})) \leq UB(I_{s,t}(PRE_{\mathcal{P}_L})) \times Strength(\mathcal{P}_R) \times Penalty(\mathcal{P}_R)$. □

Based on our upper bound in Lemma 2, we can achieve tighter bound compared with Equation 14. With simple modification of Algorithm 1, we can easily combine this new upper bound function (for bidirectional searching method) with multi-step framework. For details, please refer to https://github.com/littlepig2013/topk-meta-path-query.

7.3 Case Study on the Tightness of Bound Functions

In order to demonstrate the tightness of our newly developed bound function (cf. Lemma 2) for bidirectional search, i.e., Bi-search, compared with the bound function for unidirectional search (cf. Equation 8), i.e., Uni-search, and the basic bound function for bi-directional search (cf. Equation 14), i.e., Bi-search$_{\text{basic}}$, we conduct the following case study on these three bound functions in the ACM dataset [20]. First, we pick a (s,t)-pair, with s as Phillip Yu and t as Jiawei Han. Then, we issue the top-k meta path query on this (s,t)-pair. After that, we plot the upper bound values of these methods vesus the number of iterations. Observe from Figure 12, since our bound function for bidirectional search (Bi-search) are tighter than other bound functions, Bi-search can terminate earlier than Uni-search and Bi-search$_{\text{basic}}$. This case study also justifies our newly developed bound function (cf. Lemma 2) can enable fast bidirectional search, compared with the basic bidirectional search (Bi-search$_{\text{basic}}$), with loose bound function (cf. Equation 14).

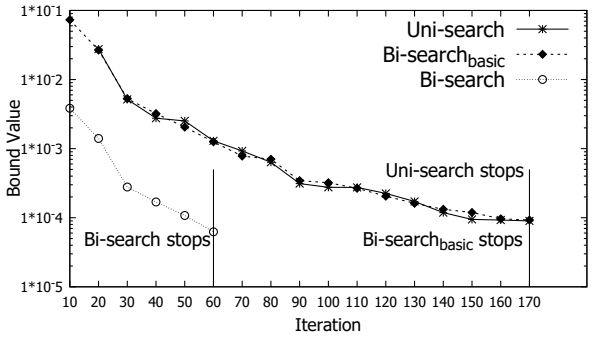


Fig. 12: Tightness of different bound functions

8 Experimental Evaluation

We first introduce the experimental setting in Section 8.1. Later, we evaluate the effectiveness (accuracy) of different importance functions in Section 8.2. After that, we demonstrate the efficiency improvement of bidirectional searching in Section 8.3. Lastly, we incorporate our importance function MNIS to the state-of-the-art embedding method, a.k.a. metapath2vec [4], and compare the effectiveness performance in the node classification task. We implement all the methods in C++ (compile all the methods with 4.6.3-version gcc) and conduct all the experiments in Ubuntu 12.04.01 with 8-core Intel i7-3770 processors.

8.1 Experimental Setting

In our experimental study, we compare different state-of-the-art importance functions with our developed importance function (cf. Section 5.3) for finding top-k meta paths between source node s and target node t, as shown in Table 2.

TABLE 2: Importance Functions

Method	Importance function	Ref.
SMP	$\frac{1}{\lvert\mathcal{P}\rvert}$	[23]
SLV1	$\frac{Strength(\mathcal{P})}{\lvert\mathcal{P}\rvert}$	[27]
SLV2	$e^{Strength(\mathcal{P})-\lvert\mathcal{P}\rvert}$	[21]
BPCRW	$BPCRW_{s,t}(\mathcal{P})$	[14]
MNIS	$I_{s,t}(\mathcal{P})$	ours

In this paper, we use two widely known datasets, DBLP and ACM, for conducting all the experiments.

DBLP [8]: DBLP is a bibliographic network, which involves 18 major conferences: AAAI, CIKM, ECIR, ECML, EDBT, ICDE, ICDM, ICML, IJCAI, KDD, PAKDD, PKDD, PODS, SDM, SIGIR, SIGMOD, VLDB and WWW. Some authors are labeled by their major research area, for example: database, data mining, information retrieval and artificial intelligence. This dataset contains 5.9K authors, 4.4K topics and 5.2K papers, with a total number of 51K links. The schema can be found in Figure 2a. We use A, P, V, T to represent author, paper, venue, topic respectively.

ACM [20]: The ACM dataset contains 14 representative computer science conferences: KDD, SIGMOD, WWW, SIGIR, CIKM, SODA, STOC, SOSP, SPAA, SIGCOMM, MobiCOM, ICML, COLT and VLDB. These conferences contain 196 venue proceedings. Besides, this dataset includes 12K papers, 17K authors, 1.8K affiliations and 1.5K frequent terms. Each paper is labeled by a subject, which is extracted for labeling in the following experiment. After we remove the subject nodes, there are 1096K links remaining in this dataset. The network schema of ACM dataset is shown in Figure 13. We use A, F, V, P, T, C to represent author, affiliation, venue, paper, term, conference respectively.

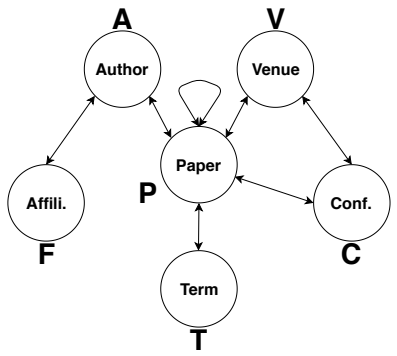


Fig. 13: ACM Schema

YAGO [8]: The YAGO dataset is a large-scale knowledge graph, which contains 2.1 million objects (vertices) and four million facts (edges). The total number of entity-types (nodes in the network schema) is 365000. We follow [8] and look for the human object pairs which

have the similar semantics (e.g., Barack_Obama and Presidency_Of_Barack_Obama), so as to clean the HIN, by deduplicating its entries.

8.2 Effectiveness

To measure the effectiveness (accuracy) of different methods, we first extract the labels of all the nodes with type Paper, type Author from the ACM and DBLP datasets respectively. In DBLP and ACM datasets, we denote the (s,t)-pair with two same labels as positive pair. Otherwise, we term it as negative pair. Then, we sample 100 positive pairs (denoted as the set POS) and 100 negative pairs (denoted as the set NEG) from the HIN. After that, we conduct the top-k meta path query for each $(s,t) \in POS$, where we denote $p_{s,t}^{(i)}$ and $n_{s,t}^{(i)}$ as the numbers of positive and negative pairs that can be connected through the retrieved i^{th} meta path ($1 \leq i \leq k$) in POS and NEG, respectively. Then, we measure the accuracy with the following equation.

$$Accuracy@k = \frac{1}{k}\sum_{i=1}^{k}\left(\frac{1}{100}\sum_{(s,t)\in POS}\frac{p_{(s,t)}^{(i)}}{p_{(s,t)}^{(i)}+n_{(s,t)}^{(i)}}\right) \quad (15)$$

In YAGO dataset, since there are 57 human object pairs that have the similar semantics, we adopt all these positive pairs in our experiment, and then we sample 150 negative pairs in this dataset. We follow the above procedure to conduct our experiment. Instead, we change 100 to 57 in Equation 15.

In the first experiment, we compare the effectiveness of different penalty functions $Penalty(\mathcal{P})$ (cf. Table 1) in our MNIS importance function. As shown in Figure 14, once the β changes from 0.1 to 0.4, the accuracy increases as some longer but important meta paths can be discovered. However, if β is too large, the accuracy does not increase as the extremely long meta paths are normally not accurate. By default, we choose $\beta = 0.2$ in later experiments.

In the second experiment, we compare the accuracy with our method MNIS and other methods. Since our method considers more components, e.g., rarity and path count, we can discover more important meta paths. As such, our method can provide more accurate result (cf. Figure 15). In the following, we provide the case study to justify this reason.

Case study: We proceed to conduct the case study to demonstrate why our importance function $I_{s,t}(\mathcal{P})$ can be better than existing importance functions (cf. Table 2). We consider the top-k meta path query, using different importance functions, between two author nodes, Jiawei Han and Phillip Yu, in the ACM dataset. Observe from Table 3, both SMP, SLV1 and SLV2 provide the high rank for short meta paths, since these importance functions mainly use the length or strength-based penalty functions. However, as mentioned in Section 1, some short meta paths, e.g., $A \rightarrow F \leftarrow A$, are not as important as some longer meta paths, e.g., $A \rightarrow P \rightarrow P \leftarrow A$. BPCRW is the variant of the path count function (cf. Section 5.2). However, this function is totally biased to the number of path counts, which can provide the high rank for the same meta path each time. For example: each paper can mention 78 terms and each term can be mentioned by 511 papers on average in ACM dataset. Therefore, no matter which (s,t)-pair (with node type A) is considered, the meta path $A \rightarrow P \rightarrow T \leftarrow P \leftarrow A$ is always chosen each time. In contrast, our importance function is unbiased to rank the meta-paths by integrating all other components, e.g., rarity and strength functions.

8.3 Efficiency improvement via bidirectional searching

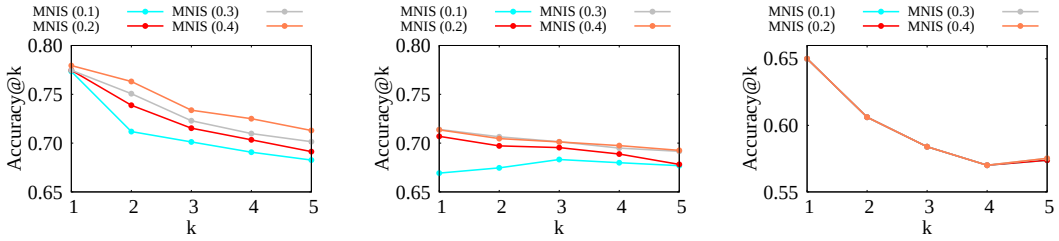
We proceed to conduct efficiency experiment on DBLP, ACM datasets. In this experiment, we randomly sample 100 (s,t)-pairs and record the average response time of the top-k meta path query, using different importance functions (cf. Table 2). For YAGO dataset, we use the 57 human object pairs with the same semantic to conduct this efficiency experiment.

First, we compare the response time between the unidirectional searching (cf. Figure 9a), which was used in our preliminary work [28], and the bidirectional searching (cf. Figure 9b). Besides, we record the maximum length among explored meta paths for different importance functions using unidirectional and bidirectional searching. In Figure 16, l_{max} represents the maximum length using unidirectional searching while l_L and l_R represent the maxmum length explored in the left and right side, respectively, in bidirectonal searching. Observe from Figure 16, we notice that the maximum length among explored meta paths is reduced by half using bidirectional searching and thus the evaluation time of different importance functions under the multi-step framework is reduced by at least 5x to one-order-of-magnitude if we adopt the bidirectional searching method (cf. Figure 17). Since MNIS considers many different components (cf. Equation 7), e.g., rarity, path count and penalty functions, compared with other importance functions, the response time is normally longer. However, given the higher accuracy performance (cf. Figure 15), it is still acceptable to use MNIS as the importance function. As a remark, we omit the evaluation time of BPCRW function in Figure 17 as this importance function cannot be supported in multi-step framework. However, the evaluation time of BPCRW function is more than 1000 seconds, which is the slowest one compared with other importance functions in Figure 17.

In our second experiment, we compare how the parameter β (of penalty function $Penalty(\mathcal{P})$) affects the query performance of our MNIS importance function. Once the β increases, there is less penalty on long meta paths, which induces a large search space for the multi-step searching algorithm. Therefore, it indicates why we have longer response time for larger β in Figure 18.

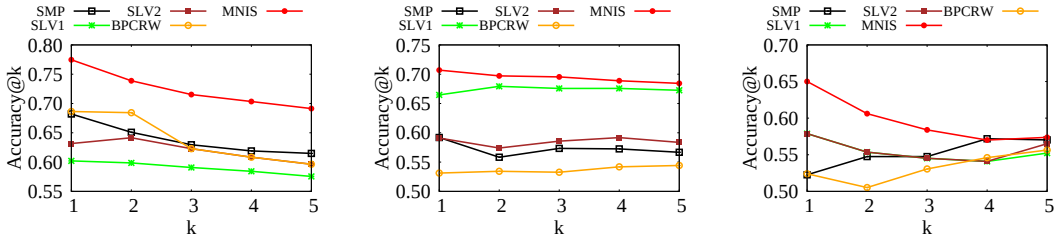
8.4 Case Study: MNIS-based metapath2vec

In existing work, Dong et al. [4], [5] have proposed the state-of-the-art representation learning method, called metapath2vec, which converts each node in an HIN into d-dimensional embedded vector, based on applying the random walk in the HIN with respect to a given meta path. Since these embedded vectors can capture the semantic meaning of different nodes, they show that metapath2vec can provide better performance in the heterogeneous academic network for node classification task [4], [5]. However, metapath2vec relies on a specified and fixed meta path to generate the random walk for each node, in which this



(a) DBLP (b) ACM (c) YAGO

Fig. 14: Accuracy@k of our MNIS importance function, varying the parameter β

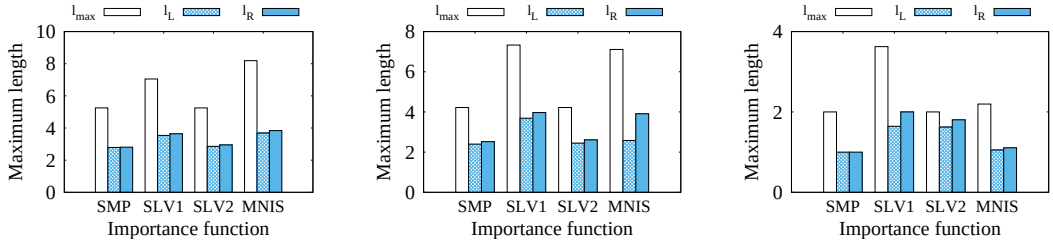


(a) DBLP (b) ACM (c) YAGO

Fig. 15: Accuracy@k of different importance functions

TABLE 3: Top-5 meta paths with different importance functions (cf. Table 2)

Rank	SMP	SLV1/ SLV2	BPCRW	MNIS
1	$A \to F \leftarrow A$	$A \to P \leftarrow A$	$A \to P \to T \leftarrow P \leftarrow A$	$A \to P \leftarrow A$
2	$A \to P \leftarrow A$	$A \to F \leftarrow A$	$A \to P \leftarrow A \to P \leftarrow A$	$A \to P \leftarrow P \leftarrow A$
3	$A \to P \to P \leftarrow A$	$A \to P \to P \leftarrow A$	$A \to P \leftarrow C \to P \leftarrow A$	$A \to P \to P \leftarrow A$
4	$A \to P \leftarrow P \leftarrow A$	$A \to P \leftarrow P \leftarrow A$	$A \to P \to P \leftarrow P \leftarrow A$	$A \to P \leftarrow V \to P \leftarrow A$
5	$A \to P \leftarrow P \to P \leftarrow A$	$A \to P \leftarrow P \to P \leftarrow A$	$A \to P \leftarrow P \to P \leftarrow A$	$A \to F \leftarrow A$



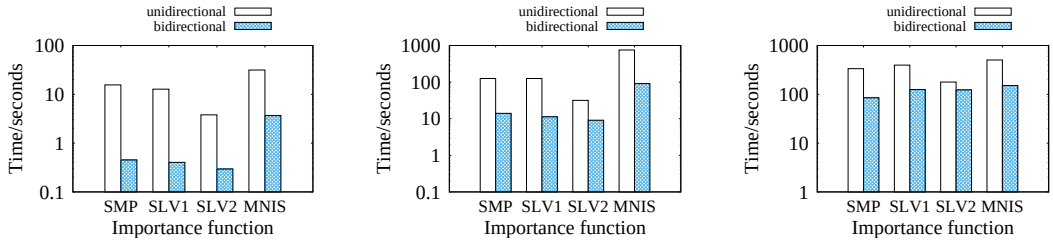
(a) DBLP (b) ACM (c) YAGO

Fig. 16: Maximum length among explored meta paths for different importance functions in multi-step framework (we fix $k = 3$), using unidirectional/bidirectional searching

meta path is hard to be properly selected without domain knowledge [5], [14]. In addition, different meta paths can be suitable for different nodes to perform random walk. To overcome these two drawbacks of metapath2vec method, we propose to integrate our MNIS importance function (cf. Equation 7) into this embedding method, in which we term this approach as MNIS-based metapath2vec, and perform the case study for the node classification task in two heterogeneous academic networks, which are DBLP and ACM. The MNIS-based metapath2vec method is summarized as follows.

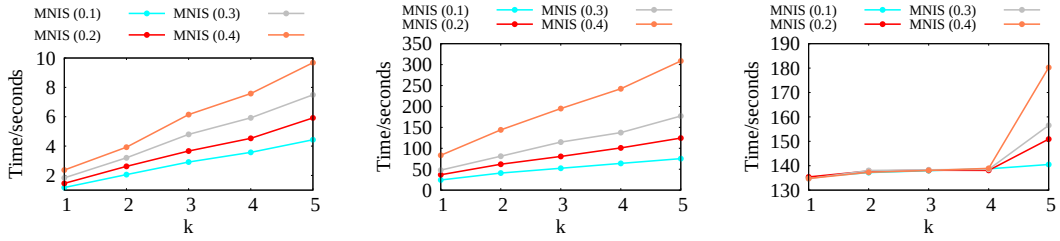
1) For each node v with label $L(v)$, e.g., "database", "data mining", in a given HIN, e.g., academic networks, we randomly sample M nodes, i.e., v_1, v_2,... and v_M, which have the same labels as $L(v)$.
2) We perform the top-k meta path query (cf. Problem 1), based on the MNIS importance function (cf. Equation 7), for each pair (v, v_i), where $1 \leq i \leq M$. Therefore, we retrieve $M \times k$ meta paths in total.
3) We aggregate the MNIS importance function values for the same meta paths in those retrieved $M \times k$ meta paths.
4) The random walk for each node v (used in the embedding method [4], [5]) follows the top-$\mathcal{L}$ meta paths, which have the top-$\mathcal{L}$ highest aggregate values (from Step 3).

In this case study, we compare the effectiveness between the MNIS-based metapath2vec method and the metapath2vec method [4], [5] with different representative metapaths, which are summarized in Table 4, in four commonly used machine learning models, including K-nearest neighbor similarity search (K-NN, with $K = 10$ in [4], [5]), random forest (RF), Naive Bayes (NB), and multilayer



(a) DBLP (b) ACM (c) YGAO

Fig. 17: Response time for different importance functions in multi-step framework (we fix $k = 3$), using unidirectional/bidirectional searching



(a) DBLP (b) ACM (c) YGAO

Fig. 18: Response time for our importance function MNIS under different β (from 0.1 to 0.4), varying the parameter k

perceptron (MLP) models. We randomly split all labelled nodes into two sets, in which 70% and 30% nodes are used for training and testing the model respectively.

TABLE 4: Five representative meta paths for metapath2vec method [4], [5] in both DBLP and ACM datasets

Meta path	DBLP	ACM
$\mathcal{M}_1$	$A \to P \leftarrow A$	$P \to P$
$\mathcal{M}_2$	$A \to P \to P \leftarrow A$	$P \to A \leftarrow P$
$\mathcal{M}_3$	$A \to P \to T \leftarrow P \leftarrow A$	$P \to C \leftarrow P$
$\mathcal{M}_4$	$A \to P \to V \leftarrow P \leftarrow A$	$P \to V \leftarrow P$
$\mathcal{M}_5$	$A \to P \to A \leftarrow P \leftarrow A$	$P \to T \leftarrow P$

We adopt the AUC scores[5] (both macro-AUC and weighted-AUC) to measure the accuracy [15]. In addition, we also fix the parameters $M = 10$, $k = 5$ and $\mathcal{L} = 3$ for the MNIS-based metapath2vec method. Tables 5 and 6 summarize the results for different methods in DBLP and ACM datasets respectively. Here, once we choose the inappropriate meta paths for the metapath2vec method, e.g., $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$ in DBLP dataset and $\mathcal{M}_2$, and $\mathcal{M}_5$ in ACM dataset, this method can achieve the low accuracy. However, since MNIS-based metapath2vec method can automatically select the suitable meta paths for random walk in each node v based on our MNIS importance function, which can provide better embeddings, these methods can normally achieve higher accuracy without the need of domain experts. Observe that incorporating the MNIS with the metapath2vec method can provide the highest accuracy in different machine learning models, compared with other methods.

9 Conclusion

In this paper, we study the problem of discovering top-k important meta paths between two nodes in an HIN. We propose an importance function, called MNIS, for a meta path based on its frequency and rarity, and then utilize an efficient top-k searching algorithm, called multi-step searching, which can work for not only the new importance function but also other existing importance functions. Extensive effectiveness experiments on DBLP, ACM and YAGO datasets show that our proposed importance function outperforms state-of-the-art methods. In addition, we further incorporate our MNIS importance function into the state-of-the-art embedding method, a.k.a. metapath2vec [4], [5], which improves the effectiveness for the node classification task without specifying the meta path in advance by the domain experts. To speed up the multi-step searching framework, we further integrate this framework with bidirectional searching method. Our experimental results show that bidirectional searching method can outperform unidirectional searching method by at least 5x to one-order-of-magnitude for all importance functions. In the future, we plan to utilize the importance function for other machine learning applications in HINs, e.g., link prediction and recommendation.

10 Appendix

10.1 Tighter lb for $UB(\mathcal{R}_{s,t}(PRE_{\mathcal{P}}))$ (Equation 10)

Recall from Equation 10, the upper bound function $UB(\mathcal{R}_{s,t}(PRE_{\mathcal{P}}))$ depends on the lower bound value lb. In Section 6.3, we state that one trivial bound value is $lb = 1$. However, this bound function is not tight in general since $|\{(u, v) \in SIM_{s,t}, u \xrightarrow{\mathcal{P}} v\}|$ can be larger than 1.

As such, we proceed to discuss how to achieve tighter lower bound lb. Consider all meta paths $\widehat{\mathcal{P}}$ with the prefix $\mathcal{P}_1 = A \xrightarrow{\text{write}} P$, i.e. $PRE_{\mathcal{P}_1}$ (cf. Figure 7b). No matter what the target t is, as long as $s \xrightarrow{\widehat{\mathcal{P}}} t$ in Figure 7a, we have $u_1 \xrightarrow{\widehat{\mathcal{P}}} t$. For example: let $t = u_4$, we have $s \xrightarrow{\mathcal{P}_5} t$. We

5. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html

TABLE 5: Macro-AUC and weighted-AUC scores for metapath2vec/MNIS-based metapath2vec methods in DBLP dataset

	Macro-AUC						Weighted-AUC					
	metapath2vec					MNIS-based	metapath2vec					MNIS-based
Model	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$	$\mathcal{M}_5$	metapath2vec	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$	$\mathcal{M}_5$	metapath2vec
K-NN	0.6492	0.6107	0.757	0.8874	0.7585	**0.895**	0.6506	0.6111	0.7585	0.8864	0.7168	**0.8946**
RF	0.6328	0.5819	0.7255	0.8795	0.684	**0.8889**	0.6346	0.5824	0.7225	0.8791	0.684	**0.8887**
NB	0.6129	0.5724	0.7946	0.8881	0.6452	**0.8938**	0.6153	0.5708	0.7941	0.8875	0.6474	**0.8934**
MLP	0.7293	0.5912	0.8244	0.8977	0.7516	**0.8986**	0.7305	0.5928	0.8234	0.8973	0.7517	**0.8982**

TABLE 6: Macro-AUC and weighted-AUC scores for metapath2vec/MNIS-based metapath2vec methods in ACM dataset

	Macro-AUC						Weighted-AUC					
	metapath2vec					MNIS-based	metapath2vec					MNIS-based
Model	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$	$\mathcal{M}_5$	metapath2vec	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$	$\mathcal{M}_5$	metapath2vec
K-NN	0.5173	0.4998	0.5181	0.5221	0.5028	**0.532**	0.6024	0.4957	0.6252	0.6438	0.525	**0.7167**
NB	0.5215	0.5	0.5146	0.5333	0.5011	**0.5407**	0.5895	0.5	0.5697	0.5627	0.5273	**0.6608**
RF	0.5075	0.5001	0.5192	0.512	0.5001	**0.5233**	0.5758	0.501	0.6458	0.5935	0.5014	**0.6957**
MLP	0.5116	0.4999	0.5186	0.5215	0.5019	**0.528**	0.5999	0.4986	0.6399	0.651	0.5176	**0.723**

can also notice $u_1 \xrightarrow{\mathcal{P}_5} t$. The main reason is both u_1 and s link to the same node u_2 with the same edge type "write". Therefore, during the searching process, once it searches for $\mathcal{P}_1$ in Figure 7b and moves from s to u_2 in Figure 7a, we can find all these links to u_2. Therefore,

$$\begin{aligned}\{(u,v) \in SIM_{s,t}, u \xrightarrow{\mathcal{P}} v\} &\supseteq \{(u,t) \in SIM_{s,t}, u \xrightarrow{\mathcal{P}} v\} \\ &\supseteq \{(s,t),(u_1,t)\}\end{aligned}$$

We can conclude for all $\mathcal{P}$ with prefix $\mathcal{P}_1$, the lower bound $lb = 2 > 1$, if $s \xrightarrow{\mathcal{P}} t$. We can extend this lower bound for other cases, e.g., $\mathcal{P}_2$ in the similar way.

10.2 Derivation of $UB(MNI_{s,t}(PRE_{\mathcal{P}}))$ (Equation 12)

Recall from Section 6.4, one valid upper bound for $UB(MNI_{s,t}(PRE_{\mathcal{P}}))$ is $\max_{\widehat{\mathcal{P}} \in PRE_{\mathcal{P}}} MNI_{s,t}(\widehat{\mathcal{P}})$. Now, we proceed to derive the upper bound for $\max_{\widehat{\mathcal{P}} \in PRE_{\mathcal{P}}} MNI_{s,t}(\widehat{\mathcal{P}})$, using the property of Equation 5.

$$\begin{aligned}&\max_{\widehat{\mathcal{P}} \in PRE_{\mathcal{P}}} MNI_{s,t}(\widehat{\mathcal{P}}) \\ =\ &\max_{\widehat{\mathcal{P}} \in PRE_{\mathcal{P}}} \min_{\substack{1<i\le l \\ a_1=s, a_{l+1}=t}} |\{a_i | p \in \widehat{\mathcal{P}}, p=(a_1,a_2,...,a_{l+1})\}| \\ \le\ &\max_{\widehat{\mathcal{P}} \in PRE_{\mathcal{P}}} \min_{\substack{1<i\le l \\ a_1=s}} |\{a_i | p \in \widehat{\mathcal{P}}, p=(a_1,a_2,...,a_{l+1})\}|\end{aligned}$$

Since $\mathcal{P}$ is the prefix of all $\widehat{\mathcal{P}} \in PRE_{\mathcal{P}}$, we have:

$$\begin{aligned}&\max_{\widehat{\mathcal{P}} \in PRE_{\mathcal{P}}} MNI_{s,t}(\widehat{\mathcal{P}}) \\ \le\ &\min_{\substack{1<i\le |P| \\ a_1=s}} |\{a_i | p \in \mathcal{P}, p=(a_1,a_2,...,a_{|P|+1})\}|\end{aligned}$$

10.3 Derivation of $UB(Strength(\mathcal{P}))$ (Equation 13)

Observe from Equation 1, we notice that:

$$OD(A \xrightarrow{R} B) \ge 1, ID(A \xrightarrow{R} B) \ge 1$$

Therefore, we have:

$$Strength(A \xrightarrow{R} B) \le 1$$

Since $\mathcal{P}$ is the prefix of all meta paths $\widehat{\mathcal{P}}$ in $PRE_{\mathcal{P}}$, i.e. $\mathcal{P}$ contains smaller set of relations compared with all $\widehat{\mathcal{P}}$, we have:

$$\max_{\widehat{\mathcal{P}} \in PRE_{\mathcal{P}}} Strength(\widehat{\mathcal{P}}) \le Strength(\mathcal{P})$$

ACKNOWLEDGEMENT

This work was supported by the Research Grants Council of Hong Kong (RGC Projects HKU 17229116, 106150091, and 17205115), the University of Hong Kong (Projects 104004572, 102009508, and 104004129), and the Innovation and Technology Commission of Hong Kong (ITF project MRP/029/18).

REFERENCES

[1] M. W. Barley, P. J. Riddle, C. Linares López, S. Dobson, and I. Pohl, "GBFHS: A generalized breadth-first heuristic search algorithm," in *SOCS*, 2018, pp. 28–36. [Online]. Available: https://aaai.org/ocs/index.php/SOCS/SOCS18/paper/view/17965

[2] L. Chang, X. Lin, L. Qin, J. X. Yu, and J. Pei, "Efficiently computing top-k shortest path join," in *Extending Database Technology*, 2015.

[3] N. Developers, "Neo4j," *Graph NoSQL Database [online]*, 2012.

[4] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *SIGKDD*, 2017, pp. 135–144. [Online]. Available: https://doi.org/10.1145/3097983.3098036

[5] Y. Dong, Z. Hu, K. Wang, Y. Sun, and J. Tang, "Heterogeneous network representation learning," in *IJCAI*, 2020, pp. 4861–4867. [Online]. Available: https://doi.org/10.24963/ijcai.2020/677

[6] J. Gao, H. Qiu, X. Jiang, T. Wang, and D. Yang, "Fast top-k simple shortest paths discovery in graphs," in *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, 2010, pp. 509–518.

[7] R. C. Holte, A. Felner, G. Sharon, N. R. Sturtevant, and J. Chen, "MM: A bidirectional search algorithm that is guaranteed to meet in the middle," *Artif. Intell.*, vol. 252, pp. 232–266, 2017. [Online]. Available: https://doi.org/10.1016/j.artint.2017.05.004

[8] Z. Huang, Y. Zheng, R. Cheng, Y. Sun, N. Mamoulis, and X. Li, "Meta structure: Computing relevance in large heterogeneous information networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1595–1604.

[9] N. Lao and W. W. Cohen, "Relational retrieval using a combination of path-constrained random walks," *Machine learning*, vol. 81, no. 1, pp. 53–67, 2010.

[10] M. Ley, "Dblp: some lessons learned," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1493–1500, 2009.

[11] G. Liu and K. Ramakrishnan, "A* prune: an algorithm for finding k shortest paths subject to multiple constraints," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. IEEE, 2001, pp. 743–749.

[12] H. Liu, C. Jin, B. Yang *et al.*, "Finding top-k shortest paths with diversity," *IEEE Transactions on Knowledge and Data Engineering*, 2017.

[13] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: a system for large-scale graph processing," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 2010, pp. 135–146.

[14] C. Meng, R. Cheng, S. Maniu, P. Senellart, and W. Zhang, "Discovering meta-paths in large heterogeneous information networks," in *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015, pp. 754–764.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[16] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242, 2003, pp. 133–142.

[17] T. Seidl and H. Kriegel, "Optimal multi-step k-nearest neighbor search," in *SIGMOD*, 1998, pp. 154–165. [Online]. Available: https://doi.org/10.1145/276304.276319

[18] Z. Shang and J. X. Yu, "Auto-approximation of graph computing," *Proceedings of the VLDB Endowment*, vol. 7, no. 14, pp. 1833–1844, 2014.

[19] B. Shi and T. Weninger, "Mining interesting meta-paths from complex heterogeneous information networks," in *Data Mining Workshop (ICDMW), 2014 IEEE International Conference on*. IEEE, 2014, pp. 488–495.

[20] C. Shi, X. Kong, Y. Huang, S. Y. Philip, and B. Wu, "Hetesim: A general framework for relevance measure in heterogeneous networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 10, pp. 2479–2492, 2014.

[21] C. Shi, C. Zhou, X. Kong, P. S. Yu, G. Liu, and B. Wang, "Heterecom: a semantic-based recommendation system in heterogeneous networks," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 1552–1555.

[22] Y. Sun and J. Han, "Mining heterogeneous information networks: a structural analysis approach," *Acm Sigkdd Explorations Newsletter*, vol. 14, no. 2, pp. 20–28, 2013.

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

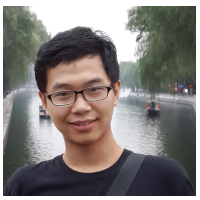
[24] Y. Sun, B. Norick, J. Han, X. Yan, P. S. Yu, and X. Yu, "Pathselclus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 7, no. 3, p. 11, 2013.

[25] X. Wei, Z. Liu, L. Sun, and P. S. Yu, "Unsupervised meta-path reduction on heterogeneous information networks," *arXiv preprint arXiv:1810.12503*, 2018.

[26] J. Y. Yen, "Finding the k shortest loopless paths in a network," *management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[27] T. Zhu, Z. Peng, S. Wang, S. Y. Philip, and X. Hong, "Measuring the relevance of different-typed objects in weighted signed heterogeneous information networks," in *Computer Supported Cooperative Work in Design (CSCWD), 2017 IEEE 21st International Conference on*. IEEE, 2017, pp. 556–561.

[28] Z. Zhu, R. Cheng, L. Do, Z. Huang, and H. Zhang, "Evaluating top-k meta path queries on large heterogeneous information networks," in *ICDM*, 2018, pp. 1470–1475. [Online]. Available: https://doi.org/10.1109/ICDM.2018.00204



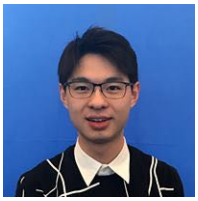
Zichen Zhu received the bachelor's degree in software engineering from Tsinghua University in 2017. He is currently an MPhil candidate in the University of Hong Kong. His research interests include data management systems, access methods, and storage systems.

Tsz Nam Chan received the bachelor's degree in electronic and information engineering and the PhD degree in computer science from the Hong Kong Polytechnic University in 2014 and 2019 respectively. He worked as the post-doctoral researcher in The University of Hong Kong from Sep 2018 to Aug 2020. He is currently a research assistant professor in the Hong Kong Baptist University. His research interests include multidimensional similarity search, pattern matching and kernel methods for machine learning. He is a member of IEEE.

Reynold Cheng is an Associate Professor of the Department of Computer Science in the University of Hong Kong (HKU). He obtained his PhD from Department of Computer Science of Purdue University in 2005. Dr. Cheng was granted an Outstanding Young Researcher Award 2011-12 by HKU.

Loc Do received the PhD degree in Information Systems from Singapore Management University in 2017. He is currently working at Alibaba, China. His research interests include text mining, information retrieval and transfer learning.

Zhipeng Huang received his B.Sc. degree in Machine Intelligence in 2011 from Peking University, China. He is now a PhD student at the Department of Computer Science, University of Hong Kong. His research focuses on the mining of heterogeneous information networks.



Haoci Zhang received his bachelor's degree from Tsinghua University in 2017 and his M.S. degree from Columbia University in 2019. He is currently a software engineer at Facebook Inc. His research interests include database, data visualization and data mining.