Meta Structure: Computing Relevance in Large Heterogeneous Information Networks

Zhipeng Huang, Yudian Zheng, Reynold Cheng, Yizhou Sun †, Nikos Mamoulis, Xiang Li
The University of Hong Kong, †Northeastern University
{zphuang, ydzheng2, ckcheng, nikos, xli2}@cs.hku.hk, †yzsun@ccs.neu.edu

ABSTRACT

A heterogeneous information network (HIN) is a graph model in which objects and edges are annotated with types. Large and complex databases, such as YAGO and DBLP, can be modeled as HINs. A fundamental problem in HINs is the computation of closeness, or *relevance*, between two HIN objects. Relevance measures can be used in various applications, including entity resolution, recommendation, and information retrieval. Several studies have investigated the use of HIN information for relevance computation, however, most of them only utilize simple structure, such as path, to measure the similarity between objects. In this paper, we propose to use *meta structure*, which is a directed acyclic graph of object types with edge types connecting in between, to measure the proximity between objects. The strength of meta structure is that it can describe complex relationship between two HIN objects (e.g., two papers in DBLP share the same authors and topics). We develop three relevance measures based on meta structure. Due to the computational complexity of these measures, we further design an algorithm with data structures proposed to support their evaluation. Our extensive experiments on YAGO and DBLP show that meta structure-based relevance is more effective than state-of-the-art approaches, and can be efficiently computed.

1. INTRODUCTION

Heterogeneous information networks (HINs), such as DBLP [8], YAGO [15], DBpedia [1] and Freebase [2], have recently received a lot of attention. These graph data sources contain a vast number of inter-related facts, and they are used to facilitate the discovery of interesting knowledge [5, 7, 12, 13]. Figure 1 illustrates an HIN, which describes the relationship among entities of different types (e.g., author, paper, venue and topic). For example, `Jiawei Han` (a_2) has *written* a `VLDB` paper ($p_{2,2}$), which *mentions* the topic "`efficient`" (t_3).

Given two HIN objects a and b, the evaluation of their *relevance* is of fundamental importance. This quantifies the degree of closeness between a and b. In Figure 1, `Jian Pei` (a_1) and `Jiawei Han` (a_2) have a high relevance score, since they have both published papers with keyword "`mining`" in the same venue (`KDD`). Relevance finds its applications in information retrieval, recommendation, and clustering [18, 22]: a researcher can retrieve papers that have high relevance in terms of topics and venues in DBLP; in YAGO, relevance facilitates the extraction of actors who are close to a given director. As another example, in entity resolution applications, duplicated HIN object pairs having high relevance scores (e.g., two different objects in an HIN referring to the same real-world person) can be identified and removed from the HIN.

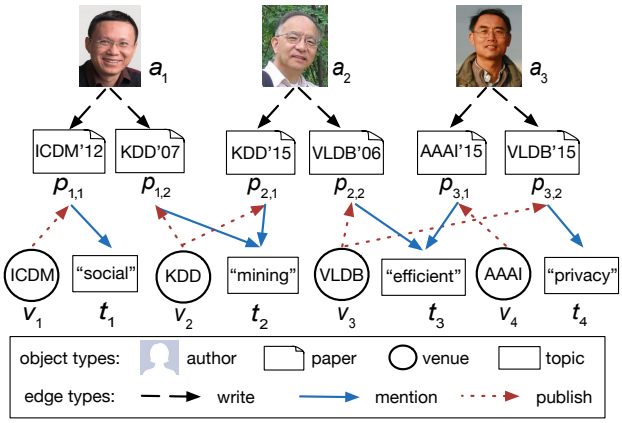


Figure 1: Illustrating an HIN.

Prior works. To measure the relevance between two graph objects, neighborhood-based measures such as *common neighbors* and *Jaccard's coefficient* were proposed [9]. Other graph-theoretic measures that are based on random walks between objects include *Personalized PageRank* [3] and *SimRank* [6]. These measures do not consider object and edge type information in an HIN. To handle this information, the concept of *meta paths* has been recently proposed [7, 18]. A meta path is a sequence of object types with edge types in between. Figure 2(b) illustrates a meta path $\mathcal{P}_1$, which states that two authors (A_1 and A_2) are related by their publications in the same venue (V). Another meta path $\mathcal{P}_2$ says that two authors have written papers containing the same topic (T). Based on a meta path, several relevance measures, such as *PathCount*, *PathSim*, and *Path Constrained Random Walk (PCRW)* [7, 18] have been proposed. These measures have been shown to be better than those that do not consider object and edge type information.

Meta structures. We propose a novel concept, named *meta structure*, to depict the relationship of two graph objects. This is essentially a directed acyclic graph of object and edge types. Figure 2(b) illustrates a meta structure $\mathcal{S}$, which depicts that two authors are relevant if they have published papers in the same venue, and have also mentioned the same topic. A meta path (e.g., $\mathcal{P}_1$ or

KDD '16, August 13-17, 2016, San Francisco, CA, USA

DOI: http://dx.doi.org/10.1145/2939672.2939815

$\mathcal{P}_2$) is a special case of a meta structure. However, a meta path fails to capture such complex relationship that can be conveniently expressed by a meta structure (e.g., $\mathcal{S}$). Our experiments also show that meta structures are more effective than meta paths.

We provide a sound definition for meta structure. This is not straightforward, since a meta structure can be complex. We then present three relevance measures based on meta structures. These measures vary in the way that the relevance is computed. Given a meta structure $\mathcal{S}$, the *StructCount* evaluates the number of subgraphs that matches $\mathcal{S}$; the *Structure Constrained Subgraph Expansion (SCSE)* simulates the process of subgraph expansion restricted on $\mathcal{S}$; the *Biased Structure Constrained Subgraph Expansion (BSCSE)* is a generalization of *StructCount* and *SCSE*.

A challenge of these new measures is their high computational cost. In general, evaluating these measures requires a subgraph matching operation over an HIN. In a typical HIN (e.g., YAGO) that contains millions of objects and edges, this can be very expensive. Moreover, an application (e.g., clustering) may require computing relevance for many object pairs. Hence, it is important to ensure that these relevance measures can be efficiently evaluated. To tackle this challenge, we design a recursive traversal algorithm with two data structures (called *Compressed-ETree* and *i-LTable*) to improve the efficiency of relevance computation.

To validate our approaches, we have performed extensive experiments on YAGO and DBLP. The results show that our three meta structure measures are more effective in expressing relevance than meta path based approaches. Our algorithms also enable meta structure relevance to be computed efficiently on large graphs, yielding similar runtime cost to meta path measures.

The rest of this paper is as follows. We describe the HIN model and summarize existing meta path based approaches in Section 2. We introduce the meta structure in Section 3. We then define relevance measures based on meta structures in Section 4. We develop a recursive algorithm and two data structures to facilitate computing relevance measures in Section 5. Section 6 presents our experiment results. We conclude our study in Section 7.

2. HIN AND META PATHS

Let us now review the HIN model in Section 2.1. We then summarize existing meta path approaches in Section 2.2.

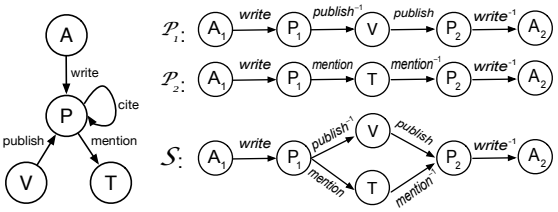
2.1 The HIN model

A *Heterogeneous Information Network (HIN)*, proposed in [18], is a directed graph $G = (V, E)$ with an object type mapping function $\phi : V \rightarrow \mathcal{L}$ and a link type mapping function $\psi : E \rightarrow \mathcal{R}$, where each object $v \in V$ belongs to an object type $\phi(v) \in \mathcal{L}$, and each link $e \in E$ belongs to a link type $\psi(e) \in \mathcal{R}$.

Figure 1 illustrates an HIN, which is also a bibliography network. A paper object can link (or be linked) to its authors, a venue and its related topics. Note that multiple edges of distinct types between two objects may exist.

DEFINITION 1. ***HIN Schema [18].*** *Given an HIN $G = (V, E)$ with mappings $\phi : V \rightarrow \mathcal{L}$ and $\psi : E \rightarrow \mathcal{R}$, its schema T_G is a directed graph defined over object types $\mathcal{L}$ and link types $\mathcal{R}$, i.e., $T_G = (\mathcal{L}, \mathcal{R})$.*

The HIN schema expresses all allowable link types between object types. Figure 2(a) shows the schema of the HIN defined in Figure 1, where the nodes A, P, T and V correspond to author, paper, topic, and venue, respectively. There are also different edge types in the schema, such as '*publish*' and '*write*'.



(a) Schema (b) Meta Path, Meta Structure

Figure 2: Schema, Meta Path, and Meta Structure.

Table 1: Relevance of Author Pairs.

Pair	Meta Path Measures			Meta Structure Measures		
	PathCount	PathSim	PCRW	StructCount	SCSE	BSCSE
a_2, a_1	2	0.5	0.25	1	0.25	0.5
a_2, a_3	2	0.5	0.25	0	0	0

2.2 Meta Paths

A *meta path* [18], denoted by $\mathcal{P}$, is essentially a path defined on an HIN schema T_G, with the types of *source object* and *target object* on both ends of the path. For example, based on the schema in Figure 2(a), a meta path $APVPA$ ($\mathcal{P}_1$ in Figure 2(b)) describes the relationship of two authors (source and target objects) who have published papers at the same venue. An *instance* of the meta path in the HIN of Figure 1 is $a_1 \rightarrow p_{1,2} \rightarrow t_2 \rightarrow p_{2,1} \rightarrow a_2$. Here we use lower-case letters (e.g., v_1) to denote objects in an HIN and upper-case letters (e.g., V) to denote object types.

Given a source object $o_s \in V$, a target object $o_t \in V$ and a meta path $\mathcal{P}$, meta path relevance measures have been proposed to evaluate the relevance between o_s and o_t:

• **PathCount [18]**: the number of meta path instances of $\mathcal{P}$ connecting o_s and o_t.

• **PathSim [18]**: a normalized version of PathCount, whose value is between 0 and 1.

• **PCRW [7]**: the probability that a random walk restricted on $\mathcal{P}$ starting from o_s would arrive at o_t.

Researchers have recently studied the use of meta paths in search and mining tasks, including top-k search [18], link prediction [16, 17, 20], clustering [4, 19], and recommendation [10, 11, 21]. As pointed out in [18], meta paths can be provided by experts who are familiar with the HIN schema. More recently, a meta path discovery algorithm has been proposed [12], where users provide example instances of source and target objects, based on which meta paths are derived automatically.

Drawbacks of meta paths. Although meta paths have been shown to be useful in different applications, they can only express simple relationship between source and target objects. As illustrated in Figure 2(b), a complex relationship ($\mathcal{S}$) between two authors cannot be captured by a path between them. To solve this problem, a straightforward way is to decompose S into two meta paths (i.e., $\mathcal{P}_1$ and $\mathcal{P}_2$). The relevance functions of two given author objects are computed for $\mathcal{P}_1$ and $\mathcal{P}_2$ separately, then the relevance based on $\mathcal{S}$ is a linear combination of the relevances based on $\mathcal{P}_1$ and $\mathcal{P}_2$ [7, 10, 12]. However, this simple approach overlooks the problem that some nodes in S (e.g., P_1) are shared by two or more edges; decomposing S into two separate meta paths results in a loss of this information. In this example, the node P_1 in S refers to a single paper. However, when S is decomposed, the corresponding P_1 nodes in meta paths $\mathcal{P}_1$ and $\mathcal{P}_2$ can mean different papers. This can yield *inaccurate* relevance results. As shown in Table 1, using

the linear combination approach, existing meta path measures regard pairs (a_2, a_1) and (a_2, a_3) to have the same relevance score. In fact, (1) a_1 and a_2 have papers (KDD'07 and KDD'15) both mentioning "mining" and published in the KDD venue; and (2) none of the papers of a_2 and a_3 are published in the same venue *and* have the same topic. Hence, (a_2, a_1) should have a higher relevance score than (a_2, a_3). The linear combination approach fails to recognize these differences, and mistakenly gives the same relevance for (a_2, a_1) and (a_2, a_3). This calls for a better measure to handle such complex relationship, as discussed next.

3. META STRUCTURES

The meta structure, designed to capture complex relationship between two HIN objects, is defined as follows.

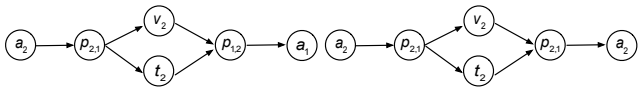
DEFINITION 2. ***Meta Structure***. *A meta structure $\mathcal{S}$ is a directed acyclic graph (DAG) with a single source node n_s (i.e., with in-degree 0) and a single sink (target) node n_t (i.e., with out-degree 0), defined on an HIN schema $T_G = (\mathcal{L}, \mathcal{R})$. Formally, $\mathcal{S} = (N, M, n_s, n_t)$, where N is a set of nodes and M is a set of edges. For any node $x \in N$, $x \in \mathcal{L}$; for any link $(x, y) \in M$, $(x, y) \in \mathcal{R}$.*

An example meta structure S is shown in Figure 2(b). We can see that $\mathcal{S}$ is a DAG, with source node $n_s = A_1$ (in-degree 0) and target node $n_t = A_2$ (out-degree 0).

In Definition 2, a meta structure has a single source node and a single target node. Otherwise, there exists at least one node v such that there is no path from n_s to n_t that goes through v. Since v does not affect the relationship between n_s and n_t, v can be removed from $\mathcal{S}$.

DEFINITION 3. ***Instance of Meta Structure***. *Given an HIN G and meta structure $\mathcal{S} = (N, M, n_s, n_t)$, an instance s of meta structure $\mathcal{S}$ on G is a subgraph of G, denoted by $s = (N_s, M_s)$, such that there exists a mapping for s, $h_s : N_s \to N$ satisfying the following constraints:*

- *Object Correspondence: for any object $v \in N_s$, its object type $\phi(v) = h_s(v)$;*
- *Link Correspondence: for any link $(u, v) \in (\notin) M_s$, we have $(h_s(u), h_s(v)) \in (\notin) M$.*



(a) Instance s_1 (b) Instance s_2

Figure 3: Instances of Meta Structure for Figure 2(b).

Figure 3 illustrates two instances of meta structure $\mathcal{S}$ in Figure 2(b), where $o_s = a_2$ for both cases.

Constructing a meta structure. In this paper, we assume that the meta structure is given. We outline several possible solutions that can be used to define a meta structure; their details are left for future work.

- Develop a Graphical User Interface (GUI) that provides drawing tools to allow meta structures to be conveniently specified.
- Use an existing graph query. For example, SPAQRL [14] is a RDF language that allows query graphs to be expressed. Since a meta structure can also be specified as a query graph, SPAQRL can be used to represent a meta structure. Meta structure relevance computation operations can also be defined on SPARQL.
- Synthesize meta paths. A meta structure can also be constructed by synthesizing existing meta paths. For example, from two meta paths $\mathcal{P}_1$ and $\mathcal{P}_2$ in Figure 2(b), we can form meta structure $\mathcal{S}$ by combining the common nodes A_1, P_1, P_2, A_2 in $\mathcal{P}_1$ and $\mathcal{P}_2$.

The above solutions assume that the user has some knowledge about the HIN schema and meta structures. Once these meta structures are defined, they can be stored in the system for non-expert users to choose. Recently, an *example-based* algorithm has been recently developed in [12], where a user first provides some example pairs of relevant source and target objects. The algorithm then discovers possible meta paths that best explain the relationship between the example pairs. It would be interesting to investigate how this method can be extended to support automatic discovery of meta structures.

3.1 Meta Structure Based Relevance

Given an HIN $G = (V, E)$ and a meta structure $\mathcal{S}$, we define the relevance function for a source object $o_s \in V$ and a target object $o_t \in V$ as follows:

$$s(o_s, o_t \mid \mathcal{S}) = \sum\nolimits_{s \in \mathcal{S}} f(o_s, o_t \mid s), \tag{1}$$

where $f(o_s, o_t \mid s)$ is a relevance measure defined on some instance s of meta structure that conforms to $\mathcal{S}$. Here, we use $s \in \mathcal{S}$ to denote the set of all instances of $\mathcal{S}$ on G.

For example, given HIN G in Figure 1, and meta structure $\mathcal{S}$ in Figure 2(b), two possible instances of $\mathcal{S}$ are shown in Figure 3. Let us define $f(o_s, o_t \mid s) = 1$ if and only if $h_s(o_s) = n_s$ and $h_s(o_t) = n_t$. Then, $s(a_2, a_1 \mid \mathcal{S}) = 1$ and $s(a_2, a_3 \mid \mathcal{S}) = 0$.

We can now define the *Relevance Search Problem* that we intend to study in this paper.

DEFINITION 4. ***Relevance Search Problem***. *Given an HIN $G = (V, E)$, a meta structure $\mathcal{S} = \{N, M, n_s, n_t\}$, a relevance measure $f(\cdot)$, and a source object $o_s \in V$, return a ranked list of target objects in decreasing order of $s(o_s, o_t \mid \mathcal{S})$, such that for any o_t in the list, $s(o_s, o_t \mid \mathcal{S}) > 0$.*

The relevance search problem is prevalent in many applications, such as information retrieval and recommendation. For example, an author can use meta structure $\mathcal{S}$ in Figure 2(b) to find out a list of potential co-authors. In our experiments, we also study this problem in the context of entity resolution, ranking, and clustering. We remark that a useful variant of this problem is to return the top-k target objects (i.e., those whose relevance scores are among the k highest), where k is specified by the user.

4. MEASURES ON META STRUCTURE

In this section, we show how the relevance measures $f(o_s, o_t \mid s)$ based on a meta structure instance s can be defined. Specifically, we first define two meta structure-based relevance measures, *StructCount* and *Structure Constrained Subgraph Expansion (SCSE)*. Then, we propose a variant of *SCSE* named *Biased Structure Constrained Subgraph Expansion (BSCSE)*, which is a generalization of *StructCount* and *SCSE*. Finally we analyze the recursive tree of *BSCSE* in detail and give an explicit definition of $f(o_s, o_t \mid s)$ for *BSCSE*.

4.1 StructCount

A straightforward relevance measure is to count the number of meta structure instances in the graph that have o_s (o_t) as source (target) node:

DEFINITION 5. ***StructCount***. *Given an HIN $G = (V, E)$, a meta structure $\mathcal{S} = (N, M, n_s, n_t)$, a source object o_s and a target object o_t, the value of StructCount is defined as the number of*

instances of $s \in \mathcal{S}$, such that o_s and o_t are mapped to n_s and n_t in $\mathcal{S}$, respectively. Recall the mapping function $h_s(\cdot)$ defined in Definition 3. Formally, for the relevance measure f of StructCount, $f(o_s, o_t \mid s) = 1$ if there exists a mapping function h_s for s, such that $h_s(o_s) = n_s$ and $h_s(o_t) = n_t$.

Take the HIN G in Figure 1 and the meta structure $\mathcal{S}$ in Figure 2(b) as an example. If we set $o_s = a_2$ and $o_t = a_1$, then the StructCount of $\mathcal{S}$ on G is 1, i.e., StructCount$(a_2, a_1 \mid \mathcal{S}) = 1$. The reason is that there is only one instance, i.e., s_1 in Figure 3 that correctly maps a_2 to A_1 and a_1 to A_2.

We can directly use StructCount to measure relevance on HINs. However, just as PathCount in meta path-based framework, the value of StructCount is not bounded. This biases highly visible objects (i.e., objects with higher degrees tend to have larger StructCount values). This could be useful when we favor popular objects, but in some applications where we favor highly relevant objects instead of popular ones, such as co-author recommendation, StructCount is not suitable.

4.2 SCSE

The fact that StructCount is a biased measure motivates us to define another relevance measure, named *Structure Constrained Subgraph Expansion (SCSE)*. Intuitively, SCSE models the probability that the source object o_s would expand to an instance of $\mathcal{S}$ that covers the target object o_t. As the value of SCSE is between 0 and 1, it removes the bias of highly visible nodes.

Before defining SCSE, we first need to define a concept of layer for meta structure.

DEFINITION 6. ***Layer of Meta Structure***. *Given a meta structure $\mathcal{S} = (N, M, n_s, n_t)$, we can partition its nodes w.r.t. their topological order in $\mathcal{S}$. Specifically, we denote by $\mathcal{S}[i] \subseteq N$ as the nodes of the i-th layer, and by $\mathcal{S}[i : j]$ $(1 \le i \le j)$ as the nodes from the i-th to the j-th layer. We denote by $d_{\mathcal{S}}$ the number of layers, thus $\mathcal{S}[1 : d_{\mathcal{S}}] = N$. Note that $\mathcal{S}[\cdot]$ is a partition of nodes in N, thus for any $i \neq j$, $\mathcal{S}[i] \cap \mathcal{S}[j] = \emptyset$.*

For example, the meta structure $\mathcal{S}$ in Figure 2(b) has $d_{\mathcal{S}} = 5$ layers. That is, $\mathcal{S}[i]$ for $1 \le i \le 5$ are $\{A_1\}$, $\{P_1\}$, $\{V, T\}$, $\{P_2\}$ and $\{A_2\}$, respectively.

Given an HIN G and a meta structure $\mathcal{S}$, starting from a source object $o_s \in V$, we can generate all possible instances $s \in \mathcal{S}$ following the layers of $\mathcal{S}$. For example, given an HIN G (Figure 1) and a meta structure $\mathcal{S}$ (Figure 2(b)), starting from an instance $o_s = a_2$, we can generate all the instances of $s \in \mathcal{S}$ on G by recursively expanding subgraph of G as shown in Figure 4.

In order to define the process of subgraph expansion, we denote by $\sigma(g, i \mid \mathcal{S}, G)$ the $(i+1)$-th layer's instances expanded from $g \in \mathcal{S}[1 : i]$ on G. For example, if g is the graph 3(a) in Figure 4, then $\sigma(g, 3 \mid \mathcal{S}, G)$ is a set containing the graphs 4(a) and 4(b) because they are instances of $\mathcal{S}[1 : 4]$ expanded from g.

Based on these notations, we now turn to a more unbiased measure, defined below.

DEFINITION 7. ***Structure Constrained Subgraph Expansion (SCSE)***. *Given an HIN $G = (V, E)$, a meta structure $\mathcal{S}$, a source object $o_s \in V$ and a target object $o_t \in V$, the SCSE of a i-th layer subgraph $g \subseteq G$ is defined recursively as follows:*

$$SCSE(g, i \mid \mathcal{S}, o_t) = \frac{\sum_{g' \in \sigma(g, i \mid \mathcal{S}, G)} SCSE(g', i+1 \mid \mathcal{S}, o_t)}{\mid \sigma(g, i \mid \mathcal{S}, G) \mid},$$

where the base case is the instance at layer $d_{\mathcal{S}}$. $SCSE(g, d_{\mathcal{S}} \mid \mathcal{S}, o_t) = 1$ if and only if there exists a mapping function h_g for g such that $h_g(o_t) = n_t$. We are interested in $SCSE(o_s, 1 \mid \mathcal{S}, o_t)$.

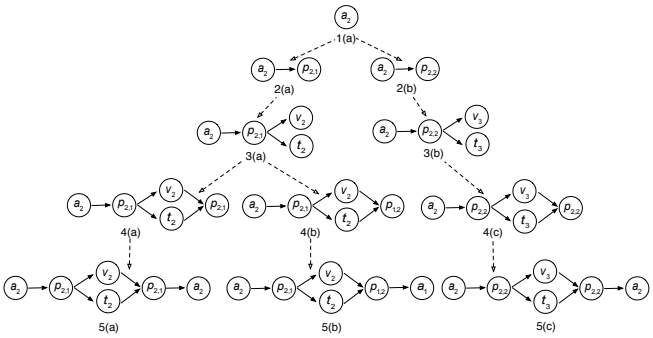


Figure 4: An Example ETree.

For example, given the HIN G of Figure 1, meta structure $\mathcal{S}$ in Figure 2(b), and $o_s = a_2$, $o_t = a_1$, starting from o_s, we show the process of subgraph expansion in Figure 4. In the last layer, i.e., the base case, only 5(b) correctly maps a_1 to A_2 (5(a) and 5(c) do not have a_1). In the first layer, we derive the value of our interest $SCSE(s_2, 1 \mid \mathcal{S}, a_1) = \frac{1/2+0}{2} = \frac{1}{4}$.

We can see that SCSE models the probability that an initial subgraph of G (i.e., o_s) would expand to an instance of $\mathcal{S}$ covering o_t. Obviously, the value of SCSE is between 0 and 1, so it can remove the bias to highly visible objects.

4.3 BSCSE: A Unified Measure

From the definitions above, we observe that both StructCount and SCSE restrict search to subgraphs that can *strictly* match the meta structure. For example, StructCount measures the absolute number of such subgraphs, while SCSE applies graph expansion from source object o_s to an instance covering the target object o_t. Each measure has its own pros and cons. To make the best of both measures and combine them in a unified framework, we propose a variant of SCSE, named *Biased Structure Constrained Subgraph Expansion (BSCSE)*, defined as follows.

DEFINITION 8. ***Biased Structure Constrained Subgraph Expansion(BSCSE)***. *Given an HIN $G = (V, E)$, a meta structure $\mathcal{S}$, a source object $o_s \in V$ and a target object $o_t \in V$, the BSCSE of a i-th layer subgraph $g \subseteq G$ is defined recursively as follows:*

$$BSCSE(g, i \mid \mathcal{S}, o_t) = \frac{\sum_{g' \in \sigma(g, i \mid \mathcal{S}, G)} BSCSE(g', i+1 \mid \mathcal{S}, o_t)}{\mid \sigma(g, i \mid \mathcal{S}, G) \mid^{\alpha}},$$

where for the base case, i.e., $i = d_{\mathcal{S}}$, we have $BSCSE(g, d_{\mathcal{S}} \mid \mathcal{S}, o_t) = 1$ if and only if there exists a mapping function h_g for g such that $h_g(o_t) = n_t$. We are interested in $BSCSE(o_s, 1 \mid \mathcal{S}, o_t)$.

Note that $\alpha \in [0, 1]$ is a bias factor to balance the weight between StructCount and SCSE: (1) a smaller α cares more about the number of subgraphs that match the meta structure (if $\alpha = 0$, BSCSE reduces to StructCount); (2) a larger α focuses more on the possibility that a random expansion can cover the target object (if $\alpha = 1$, BSCSE reduces to SCSE). On the other hand, as we have combined StructCount and SCSE into a unified BSCSE framework, we can focus on the computation of BSCSE only.

4.4 ETree

In this subsection, we analyze the expanding process of BSCSE, and give an explicit expression of the relevance measure $f(o_s, o_t \mid s)$ for BSCSE.

As we can see in Definition 8, the computation of BSCSE simulates the process of subgraph expansion. If we track the expansion

path from the original source object o_s to an instance $s \in \mathcal{S}$, we can get a recursive tree of subgraph expansion. We define this recursive tree **ETree**, as follows:

DEFINITION 9. ***ETree***. *Given an HIN G, a meta structure $\mathcal{S}$ and a source object o_s, the structure ETree is denoted as $ETree = (T, L, w)$, where*
- *T: the tree node set, where each node is a subgraph of G;*
- *L: the edge set;*
- *w: a function $w(\cdot)$ that maps a tree node $v \in T$ to its weight $w(v)$. The weight is defined based on v's parent u, i.e., $(u, v) \in L$. It considers (1) u's weight $w(u)$, and (2) the #children of u, i.e., $|\{v'|(u, v') \in T\}|$. Specifically, we have*

$$w(v) = \begin{cases} 1 & \text{if } v = o_s, \\ \frac{w(u)}{|\{v'|(u,v') \in T\}|^{\alpha}} & \text{otherwise.} \end{cases}$$

For example, given G in Figure 1 and $\mathcal{S}$ in Figure 2(b), the ETree that starts from a_2 is shown in Figure 4. We can see that the root is a_2, and each edge links a subgraph to one of its one layer expansion w.r.t. $\mathcal{S}$. For example, a_2 can either expand to $\{a_2, p_{2,1}\}$ or $\{a_2, p_{2,2}\}$ w.r.t. $\mathcal{S}$, and their weights are both $1/2^{\alpha}$. The leaf nodes (with depth $d_{\mathcal{S}}$) contain all instances of $\mathcal{S}$ starting from a_2.

Next, we analyze two properties of ETree, related to its height (Property 1) and node (Property 2), which help to express our value of interest, i.e., $BSCSE(o_s, 1 \mid \mathcal{S}, o_t)$ (Theorem 1).

PROPERTY 1. *The height of ETree is at most $d_{\mathcal{S}} - 1$.*

PROOF. The root of ETree is the source object o_s at the first layer of $\mathcal{S}$. Suppose $g_1 = o_s, g_2, \cdots, g_{\mathcal{S}-1}, g_k = v$ is a path from o_s to a leave nodes v, each step means a one layer expansion of subgraph. We have at most $d_{\mathcal{S}} - 1$ one layer expansions from o_s to v. Thus, the height of ETree is at most $d_{\mathcal{S}} - 1$. □

PROPERTY 2. *Each node of ETree at depth d is an instance of $\mathcal{S}[1 : d + 1]$ and each instance s of $\mathcal{S}[1 : d + 1]$ with $h_s(o_s) = n_s$ must be a node of ETree at depth d.*

PROOF BY INDUCTION. When $d = 0$, the root o_s is only an instance of $\mathcal{S}[1 : 1]$ with $h(o_s) = n_s$. Suppose Property 2 holds for $d = k$. Assume that u is a node of ETree at depth $k + 1$ and its parent node is v. Then, $u \in \sigma(v, k + 1 \mid \mathcal{S}, G)$ as $(v, u) \in L$, so u must be an instance of $\mathcal{S}[1 : k + 2]$. On the other hand, $\forall s \in \mathcal{S}[1 : k + 2]$, $s' = s - \{v \in s \mid h_s(v) \in \mathcal{S}[k + 2]\}$ must be an instance of $\mathcal{S}[k + 1]$, which means s is a one layer expansion of s'. So s is a node of ETree at depth $k + 1$. □

THEOREM 1. *Given a meta structure $\mathcal{S} = (N, M, n_s, n_t)$, a souce object $o_s \in V$ and a target object $o_t \in V$,*

$$BSCSE(o_s, 1 \mid \mathcal{S}, o_t) = \sum_{s \in \mathcal{S},\ h_s(o_t) = n_t} w(s). \tag{2}$$

PROOF. Suppose s is an instance of $\mathcal{S}$ and $g_1 = o_s$, g_2, …, $g_{d_{\mathcal{S}}} = s$ is the path of ETree from o_s to s. According to the recursive definition of $BSCSE$,

$$f(o_s, o_t \mid s) = \prod_{i=1}^{d_{\mathcal{S}}-1} \frac{1}{|\ \sigma(g_i, i \mid \mathcal{S}, G)\ |^{\alpha}}, \text{ if } h_s(o_t) = n_t.$$

According to Properties 1 and 2, s must be a leaf node at depth $d_{\mathcal{S}} - 1$, and P must be a path of ETree from root o_s to s. According to the definition of w, $w(s) = \prod_{i=1}^{d_{\mathcal{S}}-1} \frac{1}{|\ \sigma(g_i, i \mid \mathcal{S}, G)\ |^{\alpha}}$, then we can finally derive

$$BSCSE(o_s, 1 \mid \mathcal{S}, o_t) = \sum_{s \in \mathcal{S},\ h_s(o_t) = n_t} w(s).$$

□

Based on the proof of Theorem 1, we know that the relevance measure $f(o_s, o_t \mid s)$ for BSCSE is:

$$f(o_s, o_t \mid s) = \begin{cases} w(s) & \text{if } h_s(o_t) = n_t, \\ 0 & \text{otherwise.} \end{cases}$$

Take the HIN in Figure 1 for example, we show the relevance values for two pairs of authors with our three measures in Table 1. We can see that our three meta structure relevance measures can better handle complex relationship, i.e., the relevance score of (a_2, a_1) is larger than (a_2, a_3). This is because the meta structure can make use of the information of common nodes in different meta paths.

5. COMPUTING BSCSE

As we know, BSCSE is a generalization of StructCount and SCSE. Thus in this section, we study how to efficiently perform relevance search with BSCSE (also works for StructCount and SCSE) w.r.t. a source object o_s, based on a given $\mathcal{S}$. We first propose a traversal algorithm on ETree (Section 5.1), and then further improve its efficiency by proposing two optimizations (Section 5.2).

5.1 Traversal Algorithm

In order to calculate $BSCSE(o_s, 1\ \mathcal{S}, G)$, an initial idea is to visit all the leaf nodes of ETree and accumulate the weights of all $s \in \mathcal{S}$ for which $h_s(o_t) = n_t$. Based on this, we develop a recursive algorithm, called *Traversal* (Algorithm 1) to compute BSCSE. It first checks whether the base case is caught, i.e., if g is already an instance of $\mathcal{S}$. In this case, the instance g with its weight w are returned (steps 1-2). The rest of the algorithm consists of two phases. The first phase (steps 3-11) calculates the set $\sigma(g, layer \mid \mathcal{S}, G)$ and the second phase (steps 12-17) recursively calls the algorithm for each $g' \in \sigma(g, layer \mid \mathcal{S}, G)$ and accumulates the results.

In the first phase, for each node n at the $(i + 1)$-th layer of $\mathcal{S}$, we consider all nodes n' such that $(n', n) \in M$, check its instance object $g[n']$ and calculate possible instance objects w.r.t. node n (steps 6-8). Then, we calculate the instance objects w.r.t. n that satisfy all the dependency constrains (step 9). Finally, we compute the Cartesian product over the possible instances of each node at layer $(i + 1)$, and derive the set of possible expansions σ (step 11).

In the second phase, we first record the weight w' for layer $(i + 1)$ according to Definition 9 (step 12). Then, for each possible expansion, we expand the subgraph g to g' (step 15) and recursively call the algorithm on the expanded subgraph g' to get all instances of $\mathcal{S}$ and their corresponding weights (step 16).

For example, suppose we are traversing the ETree in Figure 4 based on the meta structure $\mathcal{S}$ in Figure 2(b). We set layer as 3 and g as the graph in 3(b). In the first phase, there is only one meta node n at the 4th layer, i.e., $n = P_2$, and n depends on two nodes, i.e., V and T. Then, we can see that $g[V] = v_3$, and it has two neighbors $p_{2,2}$ and $p_{3,2}$; $g[T] = t_3$, and it has two neighbors $p_{2,2}$ and $p_{3,1}$. We get $C = \{\{p_{2,2}, p_{3,2}\}, \{p_{2,2}, p_{3,1}\}\}$, and $Ins[P_2] = \bigcap C = \{p_{2,2}\}$, which means that there is only one possible instance object for P_2. At the second phase, we have $w' = w$ because there is only one possible expansion. We then compute the expanded subgraph $g' = g \cup \{p_{2,2}\}$, and recursive call $Traversal(G, \mathcal{S}, g', w', 4)$.

5.2 Optimizations

We propose two optimizations on the traversal algorithm to boost the efficiency. First, we devise a compressed representation of ETree to reduce the redundancy. Then, we propose an index structure to further accelerate the process of online query.

Algorithm 1: Traversal Algorithm

Input: HIN G, meta structure $\mathcal{S}$, subgraph g, weight w and $layer$ id.
Output: all possible instances of $\mathcal{S}$ and their weights.

1 **if** $layer == d_{\mathcal{S}}$ **then**
2 **return** $\{< g, w >\}$;
3 Initialize $Ins[\cdot]$;
4 **for** $n \in \mathcal{S}[layer + 1]$ **do**
5 $C \leftarrow \emptyset$;
6 **for** $(n', n) \in M$ **do**
7 $F \leftarrow \{v \mid \psi(g[n'], v) = (n', n)\}$;
8 $C \leftarrow C \cup \{F\}$;
9 $E \leftarrow \bigcap C$;
10 $Ins[n] \leftarrow E$;
11 $\sigma \leftarrow \prod_{n \in Ins} Ins[n]$;
12 $w' = \frac{w}{|\sigma|^{\alpha}}$;
13 $rtn \leftarrow \emptyset$;
14 **for** $combination \in \sigma$ **do**
15 $g' \leftarrow g \cup combination$;
16 $\mathcal{I} \leftarrow Traversal(G, \mathcal{S}, g', w', layer + 1)$;
17 $rtn \leftarrow rtn \cup \mathcal{I}$;
18 **return** rtn;

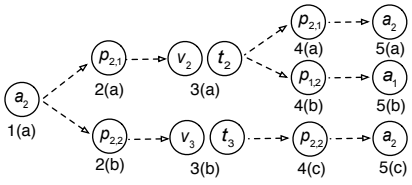


Figure 5: A Compressed-ETree.

Table 2: 3-LTable.

key	*value*
$<v_1, t_1>$	$<a_1, 1.0>$
$<v_2, t_2>$	$<a_1, 0.5>$
	$<a_2, 0.5>$
$<v_3, t_3>$	$<a_2, 1.0>$
$<v_3, t_4>$	$<a_3, 1.0>$
$<v_4, t_3>$	$<a_3, 1.0>$

Compressed-ETree

According to Property 2, at an internal node v of ETree with depth d, we need to maintain an instance of $\mathcal{S}[1 : d + 1]$. However, to further expand v, we do not necessarily need the information of the whole instance. Instead, we just need to maintain a subset of v on which the layers of $\mathcal{S}$ after d have dependencies. For example, in graph 2(a) (Figure 4), we do not need to maintain the whole graph; instead, $\{p_{2,1}\}$ is enough to represent it as the rest of meta structure only depends on node P_1.

Based on this idea, we develop a compressed structure, called *Compressed-ETree*, which is shown in Figure 5. We can see that it is more concise compared to ETree (Figure 4). Thus by traversing Compressed-ETree instead of ETree, we can reduce the computation cost and required space for each tree node.

To derive the Compressed-ETree , intuitively we have to pre-compute and maintain the necessary nodes for each layer of meta structure, which we call the *dependency set*. We use a map structure to store the nodes that need to be maintained at each layer. The details are shown in Algorithm 2. Specially, for each node n' in $\mathcal{S}$, we first get the maximal layer that the node can reach, i.e., depending on n' (step 3). Then we add n' to the corresponding layers of D (steps 4-5). After all nodes have been added to D, we can get the set of nodes we need to maintain at layer i in $D[i]$. Take $\mathcal{S}$ in Figure 2(b) as an example; the dependency set $D[i]$ for $i = 1$ to 5 is $\{A_1\}$, $\{P_1\}$, $\{V, T\}$, $\{P_2\}$ and $\{A_2\}$, respectively.[1]

By considering the dependency set $D[*]$, we can improve the performance by traversing Compressed-ETree instead of ETree. The algorithm is slightly different from Algorithm 1. At step 16, instead of calling it recursively on a whole subgraph g', we can just maintain a subset of g' which is in the dependency set $D[layer + 1]$.

[1]Note that $D[i]$ is not necessarily equal to $\mathcal{S}[i]$, as we do not require that each edge must point to the node in the next layer.

Algorithm 2: Pre-compute Dependencies

Input: meta structure $\mathcal{S}$.
Output: $D[*]$, where $D[i]$ is a set of nodes we need to maintain in Compressed-ETree at layer i.

1 Initialize $D[i] = \emptyset$ for $i = 1, 2, \ldots, d_{\mathcal{S}}$;
2 **for** $n' \in N$ **do**
3 $d \leftarrow max_{(n', n) \in M} layer(n)$;
4 **for** $i = layer(n')$ *to* d **do**
5 $D[i] = D[i] \cup \{n'\}$;
6 **return** D;

i-LTable

Compressed-ETree can reduce the computation for each node of ETree, but it still has the same number of tree nodes. Especially there is much redundancy in the computation if we have a batch of queries to answer. For example, when computing BSCSE for two source objects a_2 and a_1 in Figure 1, we have to traverse two Compressed-ETrees for a_2 and a_1, respectively. When traversing one for a_2 (Figure 5), we visit a sub-tree with 3(a) as root; meanwhile, we would visit the same sub-tree while traversing the other Compressed-ETree for a_1. This is because the last two layers of the meta structure $\mathcal{S}[4, 5]$ only depend on $\mathcal{S}[3]$ (instead of $\mathcal{S}[1, 3]$).

By considering this idea, we propose a novel data structure called *i-LTable*, which stores all leaf nodes for a sub-tree of the Compressed-ETree in advance. Once we traverse to the i-th layer, we can get the information of leaf nodes directly from the i-LTable, which saves the search time from the $(i + 1)$-th to the last layer.

Given an $\mathcal{S}$, the i-LTable w.r.t. layer i is a data structure that maps each node instance v of Compressed-ETree at layer i to all the node instances in the last layer (with v as an ancestor). To be specific, the keys of i-LTable are the instances of the stored nodes in $D[i]$, and the values are the distributions of weights over all possible target objects. Given $\mathcal{S}$ in Figure 2(b), consider the Compressed-ETree in Figure 5, the corresponding 3-LTable is shown in Table 2. For example, as $D[3] = \{V, T\}$, and the target node $n_t = A_2$, the keys of 3-LTable are pairs of venues and topics and the values are distributions of weights on authors.

Next we study how to build an i-LTable for a given meta structure $\mathcal{S}$ offline. First, we address the selection of i, i.e., which layer the i-LTable should be built on, and then we deal with how to build indexes offline and conduct queries online.

Choosing An Appropriate i. If we have built i-LTable on the i-th layer, then we only need to search the Compressed-ETree for the top i layers. Intuitively the choice of i is a trade-off between time and space. For a smaller i, the number of nodes that needs to be visited is smaller, resulting in efficient processing. However, the number of reachable target objects is large, resulting in larger space requirement. We next list three heuristic methods on how to select i: (1) *MinKey*: choose i with minimal number of possible key values; (2) *Half*: choose $i = \frac{1}{2} \cdot d_{\mathcal{S}}$; (3) *Min*: choose a minimal i with space budget constraint.

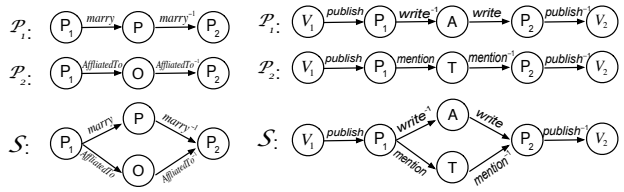
Building Indexes Offline. After choosing an i, we can start to build the i-LTable, and the details are shown in Algorithm 3. After retrieving the nodes in $D[i]$, we can construct i-LTable by traversing the sub-trees of Compressed-ETree for each possible key.

Online Query Processing. Once we have built i-LTable, we can speed up the process of online query. The algorithm is similar to

Algorithm 3: Building i-LTable

Input: HIN G, meta structure $\mathcal{S}$, dependency set D, layer i.
Output: i-LTable for $\mathcal{S}$.

```
initialize i-LTable;
for key ∈ ∏_{n∈D[i]} {v ∈ V | φ(v) = n} do
    i-LTable[ key ] ← Traversal(G, S, key, 1.0, i);
return i-LTable;
```



(a) YAGO (b) DBLP

Figure 6: Meta Paths and Meta Structures Used in Experiments.

Algorithm 1, except that it only needs to traverse the Compressed-ETree for the top i layers. Then the results can be retrieved directly from i-LTable instead of recursively searching the sub-trees.

6. EXPERIMENTS

We now discuss the experiment results. Section 6.1 describes the experiment setup. We then examine the effectiveness (Section 6.2) and efficiency (Section 6.3) of different relevance measures.

6.1 Setup

We examine two HIN datasets, namely YAGO and DBLP.

YAGO [15] is a large-scale knowledge graph derived from Wikipedia, WordNet and GeoNames. We use its "CORE Facts", i.e., YAGO-Core [12], which consists of 4 million facts (edges) of 125 types, made from 2.1 million objects. These entities have 365,000 types.

DBLP is a bibliographic network. It contains four types of objects, i.e., paper, author, venue and topic. We use a subset of DBLP, i.e., DBLP-4-Area [12], containing 5,237 papers, 5,915 authors, 18 venues, and 4,479 topics from 4 areas: database, data mining, machine learning and information retrieval. These objects are connected by 51,377 edges.

We compare our relevance metrics (i.e., StructCount, SCSE, and BSCSE) with three representative meta path measures (i.e., PathCount [18], PCRW [7], and PathSim [18]). These measures employ the meta paths and structures shown in Figure 6. We implement the experiments in C++ on an 8GB memory Mac OS X machine.

6.2 Effectiveness

We compare the quality of relevance measures in three applications: entity resolution (Section 6.2.1), ranking (Section 6.2.2) and clustering (Section 6.2.3). We then study the properties of meta structures in Section 6.2.4 and Section 6.2.5.

6.2.1 Entity Resolution

We first perform an *entity resolution* (ER) task to find pairs of objects in YAGO that refer to the same entity. For example, the two objects *Barack_Obama* and *Presidency_Of_Barack_Obama* refer to the same person. Identifying such pairs helps to "clean" an HIN by deduplicating its entries.

We manually label a small subset of data. We look for (human) object pairs that both have marriage relationship to an object. In total, we get 3020 such pairs, containing 4518 different persons. We consider these as our test data, and manually label their ground truth. We got 44 positive samples (i.e., each object pair refers to the same person), while the remaining 2976 ones are negative.

We use the meta structure $\mathcal{S}$ and two meta paths ($\mathcal{P}_1$, $\mathcal{P}_2$) in Figure 6(a) to compute the relevance. For each person (out of 4518 ones), we set it as the source object in $\mathcal{S}$, $\mathcal{P}_1$, $\mathcal{P}_2$, and use them to find target objects, which can be duplicates. Then, we get all the (target) persons such that the relevance value with respect to the source object is larger than zero. The larger the relevance, the more likely the object pairs refer to the same person. For each relevance measure, we vary the threshold for the relevance values of all returned pairs and plot the Precision-Recall Curve. We then compute the the area under the curve, i.e., AUC.

The AUC values for different metrics are shown in Table 3. Observe that meta structure based measures are more effective than the meta path ones. This is because $\mathcal{S}$ is more expressive than a single meta path (i.e., ($\mathcal{P}_1$ or $\mathcal{P}_2$). Here, $\mathcal{S}$ limits the results to those persons who are married to the same person and affiliated to the same organization, which cannot be represented by $\mathcal{P}_1$ or $\mathcal{P}_2$ alone.

We then study the effectiveness of the linear combination of the two meta paths. The relevance is computed as $s = \beta \cdot s_1 + (1 - \beta) \cdot s_2$, where s_1 and s_2 is the relevance derived by $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively. As shown in Figure 7(a), a linear combination of two meta paths is better than using $\mathcal{P}_1$ or $\mathcal{P}_2$ alone. However, as it does not consider the common nodes in the meta paths (i.e., nodes P_1 and P_2), its AUC value, based on the the optimal β, is just 0.2920, and is still worse than SCSE (i.e., 0.5640) (Table 3).

We also examine how parameter α influences the effectiveness of BSCSE. As shown in Figure 7(b), its AUC is stable for a wide range of α values. When $\alpha = 1$, BSCSE has the best result. This is consistent with our expectation, because entity resolution favors highly relevant objects, instead of popular ones.

We next show the top-10 relevant pairs for PCRW and SCSE in Table 4 (for PCRW, we use a linear combination with optimal β, achieving the best AUC value). The pairs in bold are negative samples. We see that PCRW has three negative samples in the top-10 pairs. For example, the reason that Sally Hayfron and Grace Mugabe appear in the result is that they have been married to the same person, and as the weight for the meta path $\mathcal{P}_1$ dominates the other ($\mathcal{P}_2$), this pair has a high score even though these two persons do not satisfy $\mathcal{P}_2$. This explains why meta path-based measures have lower AUC values than meta structure-based ones.

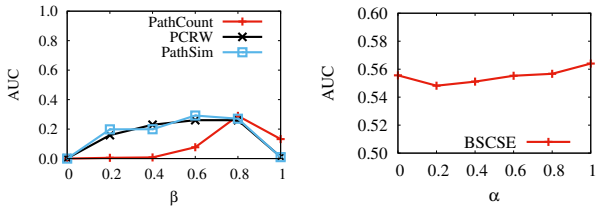
6.2.2 Ranking Quality

In our second effectiveness experiment, we perform a task of relevance ranking as follows. We first label the relevance of each pair of venues in DBLP using three levels: 0 for 'non-relevant', 1 for 'somewhat-relevant' and 2 for 'very-relevant'. We consider both the level and the scope of the venues while labeling. For example, the relevance score for SIGMOD and VLDB is 2 as they are highly relevant. We use the meta structure $\mathcal{S}$ and the two meta paths $\mathcal{P}_1$, $\mathcal{P}_2$ shown in Figure 6(b). Then, we evaluate the quality of the returned ranked list w.r.t. different measures using Normalized Discounted Cumulative Gain (nDCG), which is a commonly used measure in ranking quality, and the larger, the better.

The results are shown in Table 3. We can observe that the first meta path $\mathcal{P}_1 = VPAPV$ yields better results than the second meta path $\mathcal{P}_2 = VPTPV$ on all the three meta path-based measures. However, meta structure-based measures perform better than meta path-based measures on the whole.

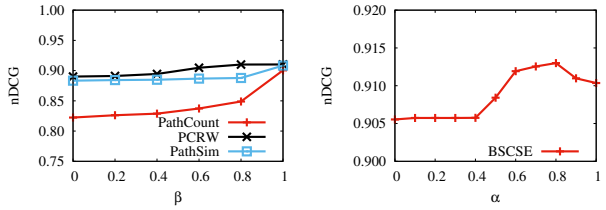
Table 3: Qualities on Three Experiments: Entity Resolution-ER (Section 6.2.1), Ranking (Section 6.2.2), and Clustering (Section 6.2.3).

Experiment	Metric	$\mathcal{P}_1$			$\mathcal{P}_2$			Linear Combination (Optimal β)			$\mathcal{S}$ (BSCSE*: Optimal α)		
		PathCount	PCRW	PathSim	PathCount	PCRW	PathSim	PathCount	PCRW	PathSim	StructCount	SCSE	BSCSE*
ER	AUC	0.1324	0.0120	0.0097	0.0003	0.0014	0.0002	0.2898	0.2606	0.2920	0.5556	**0.5640**	**0.5640**
Ranking	nDCG	0.9004	0.9047	0.9083	0.8224	0.8901	0.8834	0.9004	0.9100	0.9083	0.9056	0.9104	**0.9130**
Clustering	NMI	0.4932	0.6866	0.6780	0.3595	0.6866	0.5157	0.4932	0.6866	0.6780	0.3202	**0.8065**	**0.8065**
	Purity	2.75	3.50	3.00	2.50	3.50	2.75	2.75	3.5	3.5	2.25	**3.50**	**3.50**



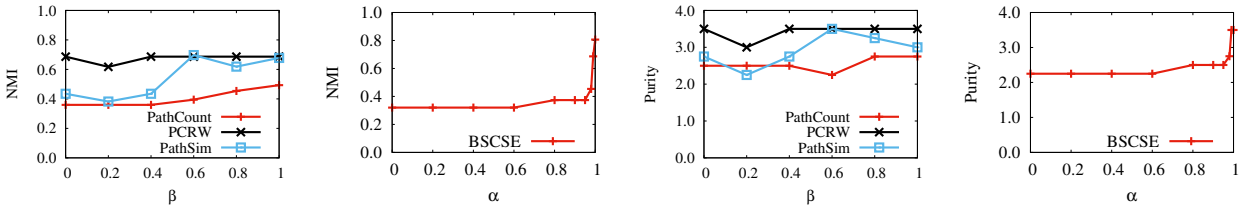
(a) Varying β (Meta Path Measures) (b) Varying α (BSCSE)

Figure 7: Varying Parameters on Different Measures (Entity Resolution).



(a) Varying β (Meta Path Measures) (b) Varying α (BSCSE)

Figure 8: Varying Parameters on Different Measures (Ranking).



(a) Varying β (Meta Path Measures) (b) Varying α (BSCSE) (c) Varying β (Meta Path Measures) (d) Varying α (BSCSE)

Figure 9: Varying Parameters on Different Measures (Clustering), with Metrics NMI (a)(b) and Purity (c)(d).

Table 4: Top-10 Relevant Pairs in YAGO.

Rank	PCRW	SCSE
1	Presidency of Corazon Aquino, Corazon Aquino	Ronald Reagan, Presidency of Ronald Reagan
2	Corazon Aquino, Presidency of Corazon Aquino	Rudy Giuliani, Political positions of Rudy Giuliani
3	Sally Ponce Enrile, Salvacion Sally Santiago Ponce Enril	Political positions of Rudy Giuliani, Rudy Giuliani
4	Presidency of Cristina Fernandez de Kirchner, Cristina Fernandez de Kirchner	Presidency of Corazon Aquino, Corazon Aquino
5	**Sally Hayfron, Grace Mugabe**	Presidency of Nestor Kirchner, Nestor Kirchner
6	**Edu Manzano, Ralph Recto**	Presidency of C. F. de Kirchner, C. F. de Kirchner
7	Gloria Macapagal Arroyo, Presidency of Gloria Macapagal Arroyo	Presidency of Ronald Reagan, Ronald Reagan
8	Presidency of Fidel V. Ramos, Fidel V. Ramos	Rise of Neville Chamberlain, Neville Chamberlain
9	Presidency of Gloria Macapagal Arroyo, Gloria Macapagal Arroyo	Outerbridge Horsey (senator), Outerbridge Horsey
10	**Marguerite of Lorraine, Marie de Bourbon**	Vice Presidency of Al Gore, Al Gore

We also compare with a linear combination of the two meta paths. We vary the weight $\beta \in [0, 1]$ to trade-off $\mathcal{P}_1, \mathcal{P}_2$, and record the nDCG values of the ranking results. The results are shown in Figure 8(a). Among meta path-based measures, PCRW performs better than PathCount and PathSim. We can see that the quality gets better as β increases. This means that the linear combination of two meta paths cannot get better results than $\mathcal{P}_1$ itself.

We further study how parameter α influences the ranking quality of BSCSE. We vary $\alpha \in [0, 1]$ and observe the quality of returned ranked list. As shown in Figure 8(b), BSCSE achieves the best nDVG value when $\alpha = 0.8$. In Table 3 we see BSCSE with optimal α (0.8) outperforms the linear combination of meta paths.

6.2.3 *Clustering Quality*

Similar to the experiment above, given the same meta structure and meta paths in Figure 6(b), in order to further evaluate the quality of relevance values between venues, we perform a task of clustering the venues in DBLP. To be specific, we apply K-means on the derived relevance matrixes w.r.t. different measures. We use two evaluation metrics, Normalized Mutual Information (NMI) and Purity (both the larger, the better). The results are shown in Table 3. We can see that SCSE has the best performance over all measures.

We further compare with a linear combination of these two meta paths. We vary the weight $\beta \in [0, 1]$ to trade-off $\mathcal{P}_1, \mathcal{P}_2$, and record the clustering accuracy. The results are shown in Figures 9(a)(c). It can be seen that PCRW performs better than PathCount and PathSim, and its performance does not vary much with β. Again, from Table 3 we observe that a linear combination of two meta paths cannot get better results than $\mathcal{P}_1$ itself. No matter what weight we give, the clustering accuracy of meta path-based measures is no better than SCSE.

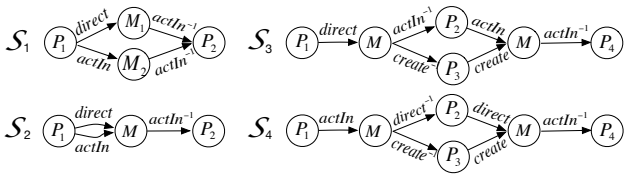


Figure 10: Meta Structures with Different Semantic Meaning.

Table 5: Top-5 Relevant Actors to *Clint Eastwood* with Different $\mathcal{S}$.

$\mathcal{S}_1$	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$
Clint Eastwood	Clint Eastwood	Clint Eastwood	Clint Eastwood
Sondra Locke	Sondra Locke	Matt Damon	Shirley MacLaine
Gene Hackman	Meryl Streep	Chief Dan George	Robert Duvall
Laura Linney	Jessica Walter	Cecile de France	Richard Burton
Marcia Gay Harden	John Larch	Sondra Locke	Fred Ward

We also study how α influences BSCSE in the task of clustering. The results are shown in Figures 9(b)(d). We can see that the clustering accuracy gets better with a larger α. When $\alpha = 1$, we have the best clustering accuracy.

We observe that in different tasks (e.g., ranking and clustering), BSCSE achieves the best performance at different values of α. This leads to the question of how to set α. We can set $\alpha = 1$ for simplicity as SCSE (i.e., $\alpha = 1$) has pretty good performances over all the tasks we perform. On the other hand, α can be set as a user input, or can be tuned with training data in the experiments.

6.2.4 Semantics of Meta Structures

Different meta structures imply different meanings. We perform a case study on YAGO to show that, with different meta structures, we can find totally different top-k results w.r.t. different relations. Specifically, we query a famous actor and director *Clint Eastwood* in YAGO with four different meta structures in Figure 10 to find top-5 relevant actors to him.

We make some analysis based on the observations of the top-5 results in Table 5: (1) Sondra Locke ranks very high in the results of $\mathcal{S}_1$ and $\mathcal{S}_2$, but has low relevance in the results of $\mathcal{S}_3$ and $\mathcal{S}_4$. This is because $\mathcal{S}_1$ and $\mathcal{S}_2$ are shorter, and they tend to find out actors who directly collaborated with Eastwood, e.g., Sondra Locke. On the other hand, $\mathcal{S}_3$ and $\mathcal{S}_4$ are longer, so they tend to find out famous actors like Matt Damon and Shirley MacLaine. (2) Matt Damon ranks high with $\mathcal{S}_3$ because he collaborated a lot with the actors and creators who had participated in the films directed by Eastwood. (3) Similarly, Shirley MacLaine ranks high with $\mathcal{S}_4$ because he collaborated a lot with the directors and creators who had participated in the films Eastwood acted in.

We can conclude that, with different meta structure, the top-5 results are different. Although $\mathcal{S}_1$ and $\mathcal{S}_2$ have the same length, $\mathcal{S}_1$ and $\mathcal{S}_2$ are different as $\mathcal{S}_2$ only consider those films with Eastwood being the director and actor at the same time, while $\mathcal{S}_1$ has looser constraint. Although $\mathcal{S}_3$ and $\mathcal{S}_4$ both have $d_{\mathcal{S}} = 5$, $\mathcal{S}_3$ only considers those films he directs and $\mathcal{S}_4$ only considers those films that he acts in. We want to show that, as meta structure is more complex than meta path, a user can use meta structures with subtle differences to express different relevances.

6.2.5 Effect of Meta Structure Sizes

We study the impact of different sizes of a meta structure. Especially, we study whether the following hold: does larger size (i.e., $d_{\mathcal{S}}$) leads to better quality for a meta structure?

To test the effectiveness of different sizes, we use concatenations of the meta structure $\mathcal{S}$ in Figure 6(b), i.e., $\mathcal{S}$, $\mathcal{S}^2$ and $\mathcal{S}^4$. Intuitively, with $\mathcal{S}$, two venues are relevant if they share the same authors and the same topics. However, with $\mathcal{S}^2$ and $\mathcal{S}^4$, the relevance becomes more subtle as the meta structures involve remote objects. When the size tends to infinity (∞), the top-k results tend to be the global result. We also compare the ranking quality (i.e., nDCG) similar to Section 6.2.2. It is shown in Table 6 that a meta structure with larger size gives worse ranking result.

Table 6: Influence of $d_{\mathcal{S}}$ on Ranking Quality.

Measure	$StructCount$			$SCSE$		
	$\mathcal{S}$	$\mathcal{S}^2$	$\mathcal{S}^4$	$\mathcal{S}$	$\mathcal{S}^2$	$\mathcal{S}^4$
nDCG	0.9055	0.7767	0.7332	0.9104	0.8026	0.7933

6.3 Efficiency

We perform two experiments to study the efficiency of the algorithm and two optimization techniques proposed in Section 5. For ease of presentation, we denote as follows:

- Traversal+: the Traversal algorithm with Compressed-ETree optimization (without index);
- Traversal++: the Traversal+ with index built on it.

In this section, we first compare the executing time of Traversal, and Traversal+ with meta path measures. Then, we study the impact of different indexes (i.e., i) in Traversal++.

6.3.1 Comparison with Meta Path Measures

We start by comparing the runtime of BSCSE with that of meta path measures. On DBLP, we ran 18 queries using the meta structure and meta paths in Figure 6(b), setting source objects as different venues. In addition, we ran 1000 queries starting from randomly selected authors using the meta structure and meta paths in Figure 2(b). On YAGO, we ran queries over 1000 randomly selected persons based on the meta structure and meta paths in Figure 6(a). We record the average executing time of each bundle of queries as shown in Table 7. We can see that meta path-based measures have different runtime performances for different meta paths. For example, a $\mathcal{P}_2$ query for venues takes 20 times more than a $\mathcal{P}_1$ query for all the three meta path-based measures. Observe that BSCSE is not worse than meta path-based measures in terms of efficiency. In addition, the Compressed-ETree optimization can slightly boost the efficiency as it can reduce the redundancy in the representation.

To further explain this phenomenon, we analyze the average number of instances by the different meta structures and meta paths. As shown in Table 8, the number of instances is proportional to the executing time. We can also see that the number of instances of meta structures are small because they are more restrictive compared to meta paths.

6.3.2 Effect of i-LTable

We show the time for building the i-LTable offline for different values of i in Figure 11(a). We can see that, as i increases, the time for building the i-LTable decreases. Particularly, if we select $i = \frac{1}{2}d_{\mathcal{S}} = 3$, we need 10s for building i-LTable.

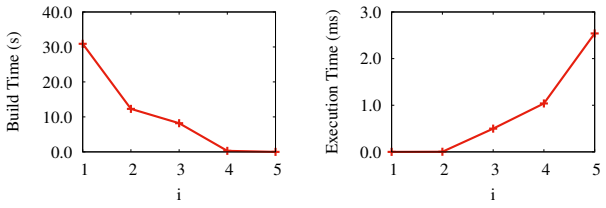
Figure 11(b) shows the time for online queries using the i-LTable for different values of i ($i = 5$ means we do not use i-LTable as $d_{\mathcal{S}} = 5$). We can see that an i-LTable greatly reduces the cost of online queries. Particularly, if we select $i = \frac{1}{2}d_{\mathcal{S}} = 3$, the Traversal++ algorithm needs only 0.5ms compared to 2.45ms required by Traversal+.

Table 7: Execution Time for Different Measures.

$\mathcal{S}$ (time unit)	$\mathcal{P}_1$			$\mathcal{P}_2$			$\mathcal{S}$	
	PathCount	PCRW	PathSim	PathCount	PCRW	PathSim	Traversal	Traversal+
venue (s)	0.055	0.065	0.054	1.187	1.181	1.188	0.528	0.516
author (10^{-2}s)	3.06	2.88	2.95	1.80	1.70	1.71	2.54	2.45
person (10^{-3}s)	2.533	2.454	2.163	7106	7086	7426	3.629	3.629

Table 8: Number of Instances.

	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{S}$
venue	5150.7	118893.2	7254.8
author	5949.0	3602.3	766.6
person	1.615	3610.3	1.259



(a) Build Time with i (b) Execution Time with i

Figure 11: Influence of i on Build and Execution Time.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we introduce a notion of meta structure, which is a powerful extension of meta path. Based on meta structure, we introduce a relevance framework on heterogeneous information networks, which can express complex relevance of two objects. In particular, we define two relevance measures under this framework, i.e., StructCount and SCSE. SCSE simulates the process of subgraph expansion, and it can reduce the bias to highly visible objects. Moreover, we define a unified measure named BSCSE, which combines StructCount and SCSE into the same framework. For efficiently computing BSCSE, we propose a recursive algorithm along with two optimizations (Compressed-ETree and i-LTable) to boost the efficiency. Experiments on real datasets demonstrate the effectiveness and efficiency of our methods.

In the future, we will examine methods for automatically learning meta structures from the knowledge base. We will also study the use of meta structure in different applications, such as citation recommendation and paper reviewer assignment.

Acknowledgments

We would like to thank the reviewers for their invaluable comments. Reynold Cheng, Zhipeng Huang, and Yudian Zheng were supported by the Research Grant Council of Hong Kong (RGC GRF project 17205115), Nikos Mamoulis was supported by RGC GRF project 715413E, and Yizhou Sun was supported by NSF CAREER 1453800, Northeastern TIER 1, and Yahoo! ACE Award.

8. REFERENCES

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: a nucleus for a web of open data. In *ISWC*, pages 722–735. Springer-Verlag, 2007.

[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250, 2008.

[3] S. Chakrabarti. Dynamic personalized pagerank in entity-relation graphs. In *WWW*, pages 571–580, 2007.

[4] J. Chen, W. Dai, Y. Sun, and J. Dy. Clustering and ranking in heterogeneous information networks via gamma-poisson model. *NTm*, 1000:1.

[5] N. Jayaram, M. Gupta, A. Khan, C. Li, X. Yan, and R. Elmasri. Gqbe: Querying knowledge graphs by example entity tuples. In *ICDE*, pages 1250–1253. IEEE, 2014.

[6] G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. In *KDD*, pages 538–543, 2002.

[7] N. Lao and W. W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67, 2010.

[8] M. Ley. Dblp computer science bibliography. 2005.

[9] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *J. Assoc. Inf. Sci. Technol.*, 58(7), 2007.

[10] X. Liu, Y. Yu, C. Guo, and Y. Sun. Meta-path-based ranking with pseudo relevance feedback on heterogeneous graph for citation recommendation. In *CIKM*, pages 121–130, 2014.

[11] X. Liu, Y. Yu, C. Guo, Y. Sun, and L. Gao. Full-text based context-rich heterogeneous network mining approach for citation recommendation. In *JCDL*, pages 361–370, 2014.

[12] C. Meng, R. Cheng, S. Maniu, P. Senellart, and W. Zhang. Discovering meta-paths in large heterogeneous information networks. In *WWW*, pages 754–764, 2015.

[13] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. Exemplar queries: Give me an example of what you need. *PVLDB*, 7(5):365–376, 2014.

[14] E. Prud'Hommeaux, A. Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.

[15] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, 2007.

[16] Y. Sun, R. Barber, M. Gupta, C. C. Aggarwal, and J. Han. Co-author relationship prediction in heterogeneous bibliographic networks. In *ASONAM*, pages 121–128, 2011.

[17] Y. Sun, J. Han, C. C. Aggarwal, and N. V. Chawla. When will it happen?: relationship prediction in heterogeneous information networks. In *WSDM*, pages 663–672, 2012.

[18] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. In *PVLDB*, pages 992–1003, 2011.

[19] 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. *TKDD*, 7(3):11, 2013.

[20] Y. Yang, N. Chawla, Y. Sun, and J. Hani. Predicting links in multi-relational and heterogeneous networks. In *ICDM*, pages 755–764, 2012.

[21] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han. Personalized entity recommendation: A heterogeneous information network approach. In *WSDM*, pages 283–292, 2014.

[22] X. Yu, X. Ren, Y. Sun, B. Sturt, U. Khandelwal, Q. Gu, B. Norick, and J. Han. Recommendation in heterogeneous information networks with implicit user feedback. In *RecSys*, pages 347–350, 2013.