PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks

Yizhou Sun[†] Jiawei Han[†] Xifeng Yan[‡] Philip S. Yu[§] Tianyi Wu[⋄]
[†] University of Illinois at Urbana-Champaign, Urbana, IL
[‡] University of California at Santa Barbara, Santa Barbara, CA
[§] University of Illinois at Chicago, Chicago, IL
[⋄] Microsoft Corporation, Redmond, WA
[†]{sun22, hanj}@illinois.edu [‡]xyan@cs.ucsb.edu [§]psyu@cs.uic.edu [⋄]tiwu@microsoft.com

ABSTRACT

Similarity search is a primitive operation in database and Web search engines. With the advent of large-scale heterogeneous information networks that consist of multi-typed, interconnected objects, such as the bibliographic networks and social media networks, it is important to study similarity search in such networks. Intuitively, two objects are similar if they are linked by many paths in the network. However, most existing similarity measures are defined for homogeneous networks. Different semantic meanings behind paths are not taken into consideration. Thus they cannot be directly applied to heterogeneous networks.

In this paper, we study similarity search that is defined among the same type of objects in heterogeneous networks. Moreover, by considering different linkage paths in a network, one could derive various similarity semantics. Therefore, we introduce the concept of *meta path-based similarity*, where a *meta path* is a path consisting of a sequence of relations defined between different object types (*i.e.,* structural paths at the meta level). No matter whether a user would like to explicitly specify a path combination given sufficient domain knowledge, or choose the best path by experimental trials, or simply provide training examples to learn it, meta path forms a common base for a network-based similarity search engine. In particular, under the meta path framework we define a novel similarity measure called *PathSim* that is able to find *peer objects* in the network (*e.g.,* find authors in the similar field and with similar reputation), which turns out to be more meaningful in many scenarios compared with random-walk based similarity measures. In order to support fast online query processing for PathSim queries, we develop an efficient solution that partially materializes short meta paths and then concatenates them online to compute top-k results. Experiments on real data sets demonstrate the effectiveness and efficiency of our proposed paradigm.

1. INTRODUCTION

Heterogeneous information networks are the logical networks involving multiple typed objects and multiple typed links denoting different relations, such as bibliographic networks, social media networks, and the knowledge network encoded in Wikipedia. It is important to provide effective search functions in such networks, where links play an essential role and attributes for objects are difficult to fully obtain. In particular, we are interested in providing similarity search functions for objects that are from the same type. For example, in a bibliographic network, a user may be interested in the (top-k) most similar authors for a given author, or the most similar venues for a given venue; in the Flickr network, a user may be interested in searching for the most similar pictures for a given picture, and so on.

Similarity search has been extensively studied for traditional categorical and numerical data types in relational data. There are also a few studies leveraging link information in networks. Most of these studies are focused on homogeneous networks or bipartite networks, such as personalized PageRank (P-PageRank) [9], SimRank [7] and SCAN [20]. However, these similarity measures are disregarding the subtlety of different types among objects and links. Adoption of such measures to heterogeneous networks has significant drawbacks: Objects of different types and links carry different semantic meanings, and it does not make sense to mix them together to measure the similarity without distinguishing their semantics.

Table 1: Top-4 similar venues for "DASFAA" under two meta paths

Rank	$CPAPC$ path	$CPTPC$ path
1	DASFAA	DASFAA
2	DEXA	Data Knowl. Eng.
3	WAIM	ACM Trans. DB Syst.
4	APWeb	Inf. Syst.

To distinguish the semantics among paths connecting two objects, we introduce a meta path-based similarity framework for objects of the same type in a heterogeneous network. A meta path is a sequence of relations between object types, which defines a new composite relation between its starting type and ending type. Consider a bibliographic network extracted from DBLP with four types of objects, namely, authors (A), papers (P), terms (T), and venues (C). Table 1 shows the top-4 most similar venues for a given venue, DASFAA, based on (a) the common authors shared by two venues, or (b) the common topics (i.e., terms) shared by two venues. These two scenarios are represented by two distinct meta paths: (a) $CPAPC$, denoting that the similarity is defined by the meta path "venue-paper-author-paper-venue", whereas (b) $CPTPC$, by the meta path "venue-paper-topic-paper-venue". A user can choose either (a) or (b) or their combination based on the preferred similarity semantics. According to Path (a), DASFAA is closer to DEXA, WAIM, and APWeb, *i.e.,* those that share many common authors, whereas according to Path (b), it is closer to Data

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Articles from this volume were invited to present their results at The 37th International Conference on Very Large Data Bases, August 29th - September 3rd 2011, Seattle, Washington.
Proceedings of the VLDB Endowment, Vol. 4, No. 7
Copyright 2011 VLDB Endowment 2150-8097/11/04... $ 10.00.

Knowl. Eng., ACM Trans. DB Syst., and Inf. Syst., *i.e.*, those that address many common topics. The meta path framework provides a powerful mechanism for a user to select an appropriate similarity semantics, by choosing a proper meta path, or learn it from a set of training examples of similar objects.

Under the proposed meta path-based similarity framework, there are multiple ways to define a similarity measure between two objects, based on concrete paths following a given meta path. One may adopt some existing similarity measures, such as (1) random walk used in P-PageRank, (2) pairwise random walk used in SimRank, or directly apply (3) P-PageRank and (4) SimRank on the extracted sub-network. However, these measures are biased to either highly visible objects (*i.e.*, objects associated with a large number of paths) or highly concentrated objects (*i.e.*, objects with a large percentage of paths going to a small set of objects). We propose a new similarity measure *PathSim*, which is able to capture the subtle semantics of similarity among peer objects in a network. In comparison, given a query object, PathSim can identify objects that not only are strongly connected but also share similar visibility in the network given the meta path. Table 2 presents in three measures the results of finding top-5 similar authors for "Anhai Doan", who is a well-established young researcher in the database field, under the meta path $APCPA$ (based on their shared venues), in the database and information system (DBIS) area. P-PageRank returns the most similar authors as those published substantially in the area, *i.e.*, highly ranked authors; SimRank returns a set of authors that are concentrated on a small number of venues shared with Doan; whereas PathSim returns Patel, Deshpande, Yang and Miller, who share very similar publication records and are also rising stars in the database field as Doan. Obviously, PathSim captures desired semantic similarity as peers in such networks.

Table 2: Top-5 similar authors for "AnHai Doan" in DBIS area

Rank	P-PageRank	SimRank	PathSim
1	AnHai Doan	AnHai Doan	AnHai Doan
2	Philip S. Yu	Douglas W. Cornell	Jignesh M. Patel
3	Jiawei Han	Adam Silberstein	Amol Deshpande
4	Hector Garcia-Molina	Samuel DeFazio	Jun Yang
5	Gerhard Weikum	Curt Ellmann	Renée J. Miller

Compared with P-PageRank and SimRank, the calculation for PathSim is much more efficient, as it is a local graph measure. But it still involves expensive matrix multiplication operations for top-k search functions, as we need to calculate the similarity between the query and every object of the same type in the network. In order to support fast online query processing for large-scale networks, we propose a methodology that partially materializes short length meta paths and then online concatenates them to derive longer meta path-based similarity. First, a baseline method (*PathSim-baseline*) is proposed, which computes the similarity between query object x and all the candidate objects y of the same type. Next, a co-clustering based pruning method (*PathSim-pruning*) is proposed, which prunes candidate objects that are not promising according to their similarity upper bounds.

The contributions of this paper are summarized as below.

1. It investigates *similarity search in heterogeneous information networks*, a new but increasingly important issue due to the proliferation of linked data and their broad applications.
2. It proposes a new framework of *meta path-based similarity* and a new definition of similarity measure, *PathSim*, that captures the subtle similarity semantics among peer objects in networks.
3. Computing PathSim is more efficient than computing P-PageRank and SimRank due to the usage of limited meta paths. Moreover, we provide an efficient *co-clustering-based computation framework* for fast query processing in large information networks.
4. Our experiments demonstrate the effectiveness of meta path-based similarity framework and the PathSim measure, in comparison with random walk-based measures, and the efficiency of PathSim search algorithms.

2. PROBLEM DEFINITION

In this section, we introduce a meta path-based similarity framework, a novel similarity measure under this framework, PathSim, and propose a PathSim-based top-k similarity search problem in information networks.

2.1 Heterogeneous Information Network

A heterogeneous information network is a special type of information network with the underneath data structure as a directed graph, which either contains multiple types of objects or multiple types of links.

DEFINITION 1. ***Information Network.*** *An* information network *is defined as a directed graph $G = (V, E)$ with an object type mapping function $\phi : V \rightarrow \mathcal{A}$ and a link type mapping function $\psi : E \rightarrow \mathcal{R}$, where each object $v \in V$ belongs to one particular object type $\phi(v) \in \mathcal{A}$, and each link $e \in E$ belongs to a particular relation $\psi(e) \in \mathcal{R}$.*

Different from the traditional network definition, we explicitly distinguish object types and relationship types in the network. Notice that, if a relation exists from type A to type B, denoted as $A\,R\,B$, the inverse relation R^{-1} holds naturally for $B\,R^{-1}\,A$. For most of the times, R and its inverse R^{-1} are not equal, unless the two types are the same and R is symmetric. When the types of objects $|\mathcal{A}| > 1$ or the types of relations $|\mathcal{R}| > 1$, the network is called **heterogeneous information network**; otherwise, it is a **homogeneous information network**.

EXAMPLE 1. *A **bibliographic information network** is a typical heterogeneous network, containing objects from four types of entities: papers (P), venues (i.e., conferences/journals) (C), authors (A), and terms (T). For each paper $p \in P$, it has links to a set of authors, a venue, a set of words as terms in the title, a set of citing papers, and a set of cited papers, and the link types are defined by these relations.*

Given a complex heterogeneous information network, it is necessary to provide its meta level (*i.e.*, schema-level) description for better understanding. Therefore, we propose the concept of network schema to describe the meta structure of a network.

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

The concept of network schema is similar to that of the ER (Entity-Relationship) model in database systems, but only captures the entity type and their binary relations, without considering the attributes for each entity type. Network schema serves as a template for a network, and tells how many types of objects there are in the network and where the possible links exist. Notice that although a relational database can often be transformed into an information network, the latter is much more general and can handle more unstructured and non-normalized data and links, and is also easier to deal with graph operations such as calculating the number of paths between two objects.

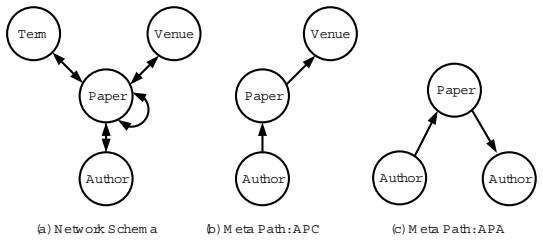


Figure 1: Bibliographic network schema and meta paths

EXAMPLE 2. ***Bibliographic network schema.*** *For a bibliographic network defined in Example 1, the network schema is shown in Fig. 1(a). Links exist between authors and papers denoting the* writing *or* written-by *relations, between venues and papers denoting the* publishing *or* published-in *relations, between papers and terms denoting* using *or* used-by *relations, and between papers, denoting* citing *or* cited-by *relations.*

2.2 Meta Path-based Similarity Framework

In a heterogeneous network, two objects can be connected via different paths. For example, two authors can be connected via "author-paper-author" path, "author-paper-venue-paper-author" path, and so on. Intuitively, the semantics underneath different paths imply different similarities. Formally, these paths are called *meta paths*, defined as follows.

DEFINITION 3. ***Meta path.*** *A* meta path $\mathcal{P}$ *is a path defined on the graph of network schema* $T_G = (\mathcal{A}, \mathcal{R})$*, and is denoted in the form of* $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \dots \xrightarrow{R_l} A_{l+1}$*, which defines a composite relation* $R = R_1 \circ R_2 \circ \dots \circ R_l$ *between type* A_1 *and* A_{l+1}*, where* $\circ$ *denotes the composition operator on relations.*

The **length** of $\mathcal{P}$ is the number of relations in $\mathcal{P}$. Further, we say a meta path is **symmetric** if the relation R defined by it is symmetric. For simplicity, we also use type names denoting the meta path if there exist no multiple relations between the same pair of types: $\mathcal{P} = (A_1 A_2 \dots A_{l+1})$. For example, in the DBLP network, the co-author relation can be described using the length-2 meta path $A \xrightarrow{\text{writing}} P \xrightarrow{\text{written-by}} A$, or short as APA if there is no ambiguity. We say a path $p = (a_1 a_2 \dots a_{l+1})$ between a_1 and a_{l+1} in network G follows the meta path $\mathcal{P}$, if $\forall i, \phi(a_i) = A_i$ and each link $e_i = \langle a_i a_{i+1} \rangle$ belongs to each relation R_i in $\mathcal{P}$. We call these paths as **path instances** of $\mathcal{P}$, which are denoted as $p \in \mathcal{P}$. A meta path $\mathcal{P}'$ is the **reverse meta path** of $\mathcal{P}$, if $\mathcal{P}'$ is the reverse path of $\mathcal{P}$ in T_G, which is denoted as $\mathcal{P}^{-1}$ and defines an inverse relation of the one defined by $\mathcal{P}$. Similarly, we define the **reverse path instance** of p as the reverse path of p in G, which is denoted as p^{-1}. Two meta paths $\mathcal{P}_1 = (A_1 A_2 \dots A_l)$ and $\mathcal{P}_2 = (A'_1 A'_2 \dots A'_k)$ are **concatenable** if and only if $A_l = A'_1$, and the concatenated path is written as $\mathcal{P} = (\mathcal{P}_1 \mathcal{P}_2)$, which equals to $(A_1 A_2 \dots A_l A'_2 \dots A'_k)$. A simple example of concatenation is: AP and PA can be concatenated to the meta path APA, which defines the co-author relation.

Analogously, a meta path in an information network corresponds to a feature in a traditional data set. Given a user-specified meta path, say $\mathcal{P} = (A_1 A_2 \dots A_l)$, several similarity measures can be defined for a pair of objects $x \in A_1$ and $y \in A_l$, according to the path instances between them following the meta path. We list several straightforward measures:

- Path count: the number of path instances p between x and y following $\mathcal{P}$: $s(x, y) = |\{p : p \in \mathcal{P}\}|$.
- Random walk: $s(x, y)$ is the probability of the random walk that starts form x and ends with y following meta path $\mathcal{P}$, which is the sum of the probabilities of all the path instances $p \in \mathcal{P}$ starting with x and ending with y, denoted as $Prob(p)$: $s(x, y) = \sum_{p \in \mathcal{P}} Prob(p)$.
- Pairwise random walk: for a meta path $\mathcal{P}$ that can be decomposed into two shorter meta paths with the same length $\mathcal{P} = (\mathcal{P}_1 \mathcal{P}_2)$, $s(x, y)$ is then the pairwise random walk probability starting from objects x and y and reaching the same middle object: $s(x, y) = \sum_{(p_1 p_2) \in (\mathcal{P}_1 \mathcal{P}_2)} Prob(p_1) Prob(p_2^{-1})$, where $Prob(p_1)$ and $Prob(p_2^{-1})$ are random walk probabilities of the two path instances.

In general, we can define a meta path-based similarity framework for the object x and object y as: $s(x, y) = \sum_{p \in \mathcal{P}} f(p)$, where $f(p)$ is some measure defined on the path instance p between x and y. Notice that, for P-PageRank and SimRank measure defined on homogeneous networks, they are weighted combinations of random walk measure or pairwise random walk measure over different meta paths in homogeneous networks, in which the meta paths are in the form of the concatenation of one relation with different lengths.

2.3 PathSim: A Novel Similarity Measure

Although there have been several similarity measures as presented above, they are biased to either highly visible objects or highly concentrated object but cannot capture the semantics of peer similarity. For example, the path count and random walk-based similarity always favor objects with large degrees, and the pairwise random walk-based similarity favors concentrated objects that the majority of the links goes to a small portion of objects. However, in many scenarios, finding similar objects in networks is to *find similar peers*, such as finding similar authors based on their field and reputation, finding similar actors based on their movie style and productivity, and finding similar products based on its function and popularity.

This motivated us to propose a new, meta path-based similarity measure, called *PathSim*, that captures the subtlety of peer similarity. The intuition behind it is that two similar peer objects should not only be strongly connected, but also share comparable visibility. As the relation of peer should be symmetric, we then confine PathSim merely on the symmetric meta paths. It is easy to see that, *round trip meta paths* with the form of $\mathcal{P} = (\mathcal{P}_l \mathcal{P}_l^{-1})$ are always symmetric.

DEFINITION 4. ***PathSim: A Meta path-based similarity measure.*** *Given a symmetric meta path* $\mathcal{P}$*, PathSim between two objects of the same type* x *and* y *is:*

$$s(x, y) = \frac{2 \times |\{p_{x \rightsquigarrow y} : p_{x \rightsquigarrow y} \in \mathcal{P}\}|}{|\{p_{x \rightsquigarrow x} : p_{x \rightsquigarrow x} \in \mathcal{P}\}| + |\{p_{y \rightsquigarrow y} : p_{y \rightsquigarrow y} \in \mathcal{P}\}|}$$

where $p_{x \rightsquigarrow y}$ *is a path instance between* x *and* y*,* $p_{x \rightsquigarrow x}$ *is that between* x *and* x*, and* $p_{y \rightsquigarrow y}$ *is that between* y *and* y*.*

This shows that given a meta path $\mathcal{P}$, $s(x, y)$ is defined in terms of two parts: (1) their connectivity defined by the number of paths between them following $\mathcal{P}$; and (2) the balance of their visibility, where the visibility is defined as the number of path instances between themselves. Notice that we do count multiple occurrences of a path instance as the weight of the path instance, which is the product of weights of all the links in the path instance. To see how this new measure works, we compare PathSim with a set of measures using a toy example to find peer authors, using meta path ACA.

EXAMPLE 3. ***Comparing a set of measures.*** *Table 3(a) shows a toy example of adjacency matrix W_{AC} between authors and venues in a network, denoting the number of papers published by each author in each venue. The query is to find the peer authors for "Mike". As "Bob" has exactly the same publication records as "Mike", it is expected to be the most similar peer. PathSim generates similarity scores: $s(Mike, Jim) = \frac{2\times(2\times50+1\times20)}{(2\times2+1\times1)+(50\times50+20\times20)} = 0.0826$, $s(Mike, Bob) = 1$, and so on; and the similarity scores derived by **P-PageRank**, **SimRank**, **random walk** (RW), and **pairwise random walk**(PRW) on the same meta path ACA, are also illustrated in Table 3(b). One can see that PathSim is the only measure giving the result that Bob and Mary are more similar to Mike than Jim is, in terms of peers, which follows human intuition.*

(a) Adjacency matrix W_{AC}

	SIGMOD	VLDB	ICDE	KDD
Mike	2	1	0	0
Jim	50	20	0	0
Mary	2	0	1	0
Bob	2	1	0	0
Ann	0	0	1	1

(b) Similarity between Mike and other authors

	Jim	Mary	Bob	Ann
P-PageRank	**0.3761**	0.0133	**0.0162**	0.0046
SimRank	**0.7156**	0.5724	**0.7125**	0.1844
RW	**0.8983**	0.0238	**0.0390**	0
PRW	**0.5714**	0.4444	**0.5556**	0
PathSim	0.0826	**0.8**	**1**	0

Table 3: Comparison of a set of similarity measures

We now introduce the calculation of PathSim between any two objects of the same type given a certain meta path.

DEFINITION 5. ***Commuting matrix.*** *Given a network $G = (V, E)$ and its network schema T_G,* a commuting matrix M for a meta path $\mathcal{P} = (A_1A_2 \dots A_l)$ *is defined as $M = W_{A_1A_2}W_{A_2A_3} \dots W_{A_{l-1}A_l}$, where $W_{A_iA_j}$ is the adjacency matrix between type A_i and type A_j. $M(i, j)$ represents the number of paths instances between object $x_i \in A_1$ and object $y_j \in A_l$ under meta path $\mathcal{P}$.*

For example, commuting matrix M for the meta path $\mathcal{P} = (APA)$ is a co-author matrix, with each element representing the number of co-authored papers for the pair of authors. Given a symmetric meta path $\mathcal{P}$, PathSim between two objects x_i and x_j from the same type can be calculated as $s(x_i, x_j) = \frac{2M_{ij}}{M_{ii}+M_{jj}}$, where M is the commuting matrix for the meta path $\mathcal{P}$, M_{ii} and M_{jj} are the visibility for x_i and x_j in the network given the meta path.

It is easy to see that the commuting matrix for the reverse meta path of $\mathcal{P}_l$, which is $\mathcal{P}_l^{-1}$, is the *transpose* of commuting matrix for $\mathcal{P}_l$. In this paper, we only consider the meta path in the round trip form of $\mathcal{P} = (\mathcal{P}_l\mathcal{P}_l^{-1})$, to guarantee its symmetry and therefore the symmetry of the PathSim measure. Notice that, if the meta path is length-2, the measure of PathSim is degenerated to a measure that compares the similarity of the neighbor sets of two objects, which is called Dice's coefficient [4]. By viewing PathSim in the meta path-based similarity framework, $f(p) = 2\frac{w(a_1,a_2)\dots w(a_{l-1},a_l)}{M_{ii}+M_{jj}}$, for any path instance p starting from x_i and ending with x_j following the meta path, where $w(a_i, a_j)$ is the weight for the link $\langle a_i, a_j\rangle$ defined in the adjacency matrix.

Some good properties of PathSim, such as symmetric, self-maximum and balance of visibility, are shown in Theorem 1. For the balance property, we can see that the larger difference of the visibility of the two objects, the smaller upper bound for their PathSim similarity.

THEOREM 1. ***Properties of PathSim:***

1. ***Symmetric**: $s(x_i, x_j) = s(x_j, x_i)$.*
2. ***Self-maximum**: $s(x_i, x_j) \in [0, 1]$, and $s(x_i, x_i) = 1$.*
3. ***Balance of Visibility**: $s(x_i, x_j) \le \frac{2}{\sqrt{M_{ii}/M_{jj}}+\sqrt{M_{jj}/M_{ii}}}$.*

PROOF. See Proof in the Appendix. □

Under the definition of PathSim, we formally define our top-k similarity search problem as follows.

DEFINITION 6. ***Top-k similarity search under PathSim.*** *Given an information network G and the network schema T_G, given a meta path $\mathcal{P} = (\mathcal{P}_l\mathcal{P}_l^{-1})$, where $\mathcal{P}_l = (A_1A_2 \dots A_l)$, the top-$k$ similarity search for an object $x_i \in A_1$ is to find sorted k objects in the same type A_1, such that $s(x_i, x_j) \ge s(x_i, x'_j)$, for any x'_j not in the returning list and x_j in the returning list, where $s(x_i, x_j)$ is defined as in Def. 4.*

Although using meta path-based similarity we can define similarity between two objects given *any* round trip meta paths, the following theorem tells us a *very long* meta path is not very meaningful. Indeed, due to the sparsity of real networks, objects that are similar may share no immediate neighbors, and longer meta paths will propagate similarities to remote neighborhoods. For example, as in the DBLP example, if we consider the meta path APA, only two authors that are co-authors have a non-zero similarity score; but if we consider longer meta paths like $APCPA$ or $APTPA$, authors will be considered to be similar if they have published papers in a similar set of venues or sharing a similar set of terms no matter whether they have co-authored. But how far should we keep going? The following theorem tells us that a very long meta path may be misleading. We now use $\mathcal{P}^k$ to denote a meta path repeating k times of the basic meta path pattern of $\mathcal{P}$, *e.g.*, $(ACA)^2 = (ACACA)$.

THEOREM 2. ***Property of PathSim under infinity length meta path.*** *Let meta path $\mathcal{P}^{(k)} = (\mathcal{P}_l\mathcal{P}_l^{-1})^k$, $M_{\mathcal{P}}$ be the commuting matrix for meta path $\mathcal{P}_l$, and $M^{(k)} = (M_{\mathcal{P}}M_{\mathcal{P}}^T)^k$ be the commuting matrix for $\mathcal{P}^{(k)}$, then by PathSim, the similarity between objects x_i and x_j as $k \to \infty$ is:*

$$\lim_{k\to\infty} s^{(k)}(i, j) = \frac{2\mathbf{r}(i)\mathbf{r}(j)}{\mathbf{r}(i)\mathbf{r}(i) + \mathbf{r}(j)\mathbf{r}(j)} = \frac{2}{\frac{\mathbf{r}(i)}{\mathbf{r}(j)} + \frac{\mathbf{r}(j)}{\mathbf{r}(i)}}$$

where $\mathbf{r}$ is the primary eigenvector of M, and $\mathbf{r}(i)$ is the i_{th} value of $\mathbf{r}$.

PROOF. See Proof in the Appendix. □

As primary eigenvectors can be used as authority ranking of objects [15], the similarity between two objects under an infinite meta path can be viewed as a measure defined on their rankings ($\mathbf{r}(i)$ is the ranking score for object x_i). Two objects with more similar ranking scores will have higher similarity (*e.g.*, SIGMOD will be similar to AAAI). Later experiments (Table 8) will show that this similarity, with the meaning of global ranking, is not that useful. Notice that, the convergence of PathSim with respect to path length is usually very fast and the length of 10 for networks of the scale of DBLP can almost achieve the effect of a meta path with an infinite length. Therefore, in this paper, we only aim at solving the top-k similarity search problem for a *relatively short* meta path.

Even for a relatively short length, it may still be inefficient in both time and space to materialize all the meta paths. Thus we propose in Section 3 materializing commuting matrices for short length meta paths, and concatenating them online to get longer ones for a given query.

3. ONLINE QUERY PROCESSING FOR SINGLE META PATH

This section is on efficient top-k *PathSim* similarity search for online queries, under a single meta path, with two algorithms proposed: *PathSim-baseline* and *PathSim-pruning*, both returning *exact* top-k results for the given query. The algorithm for multiple meta path combination with different weights is discussed in Appendix B. Note that the same methodology can be adopted by other meta path-based similarity measures, such as RW and PRW, by taking a different definition of commuting matrix accordingly.

While the definition of meta path-based similarity search is flexible to accommodate different queries, it requires expensive computations (matrix multiplications), which is not affordable for online query processing in large-scale information networks. One possible solution is to materialize all the meta paths within a given length. Unfortunately, it is time and space expensive to materialize all the possible meta paths. For example, in the DBLP network, the similarity matrix corresponding to a length-4 meta path, $APCPA$, for identifying similar authors publishing in common venues is a $710K \times 710K$ matrix, whose non-empty elements reaches $5G$, and requires storage size more than $40G$ (up to $4T$ for longer meta path between authors). Thus we propose the solution to partially materialize commuting matrices for short length meta paths, and concatenate them online to get longer ones for a given query, which returns search results in a reasonable response time while reduces the storage space significantly.

3.1 Single Meta Path Concatenation

Given a meta path $\mathcal{P} = (\mathcal{P}_l\mathcal{P}_l^{-1})$, where $\mathcal{P}_l = (A_1 \cdots A_l)$, the commuting matrix for path $\mathcal{P}_l$ is $M_{\mathcal{P}} = W_{A_1A_2}W_{A_2A_3}\cdots W_{A_{l-1}A_l}$, the commuting matrix for path $\mathcal{P}$ is $M = M_{\mathcal{P}}M_{\mathcal{P}}^T$. Let n be the number of objects in A_1. For a query object $x_i \in A_1$, if we compute the top-k most similar objects $x_j \in A_1$ for x_i on-the-fly, without materializing any intermediate results, computing M from scratch would be very expensive. On the other hand, if we have pre-computed and stored the commuting matrix $M = M_{\mathcal{P}}M_{\mathcal{P}}^T$, it would be a trivial problem to get the query results: We only need to locate the corresponding row in the matrix for the query x_i, rescale it using $(M_{ii} + M_{jj})/2$, and finally sort the new vector and return the top-k objects. However, fully materializing the commuting matrices for all possible meta paths is also impractical, since the space complexity ($O(n^2)$) would prevent us from storing M for every meta path. Instead of taking the above extreme, we partially materialize commuting matrix $M_{\mathcal{P}}^T$ for meta path $\mathcal{P}_l^{-1}$, and compute top-k results online by concatenating $\mathcal{P}_l$ and $\mathcal{P}_l^{-1}$ into $\mathcal{P}$ without full matrix multiplication.

We then examine the concatenation problem, *i.e.*, if the commuting matrix M for the full meta path $\mathcal{P}$ is not pre-computed and stored, but the commuting matrix $M_{\mathcal{P}}^T$ corresponding to the partial meta path $\mathcal{P}_l^{-1}$ has been pre-computed and stored. In this case, we assume the main diagonal of M, *i.e.*, $D = (M_{11}, \ldots, M_{nn})$, is pre-computed and stored. Since for $M_{ii} = M_{\mathcal{P}}(i,:)M_{\mathcal{P}}(i,:)^T$, the calculation only involves $M_{\mathcal{P}}(i,:)$ itself, and only $O(nd)$ in time and $O(n)$ in space are required, where d is the average number of non-zero elements in each row of $M_{\mathcal{P}}$ for each object. As the commuting matrices of $\mathcal{P}_l$ and $\mathcal{P}_l^{-1}$ are transpose to each other, we only need to store one of them in the sparse form. But from the efficiency point of view, we will keep both row index and column index for fast locating any rows and columns. In this study, we only consider concatenating the partial paths $\mathcal{P}_l$ and $\mathcal{P}_l^{-1}$ into the form $\mathcal{P} = \mathcal{P}_l\mathcal{P}_l^{-1}$ or $\mathcal{P} = \mathcal{P}_l^{-1}\mathcal{P}_l$. For example, given a pre-stored meta path APC, we are able to answer queries for meta paths $APCPA$ and $CPAPC$. For our DBLP network, to store commuting matrix for partial meta path APC only needs around $25M$ space, which is less than 0.1% of the space for materializing meta path $APCPA$. Other concatenation forms that may lead to different optimization methods are also possible (*e.g.*, concatenating several short meta paths). In the following discussion, we focus on the algorithms using the concatenation form $\mathcal{P} = \mathcal{P}_l\mathcal{P}_l^{-1}$.

3.2 Baseline

Suppose we know the commuting matrix $M_{\mathcal{P}}$ for path P_l, and the diagonal vector $D = (M_{ii})_{i=1}^n$, in order to get top-k objects $x_j \in A_1$ with the highest similarity for the query x_i, we need to compute $s(i,j)$ for all x_j. The straightforward baseline is: (1) first apply vector-matrix multiplication to get $M(i,:) = M_{\mathcal{P}}(i,:)M_{\mathcal{P}}^T$; (2) calculate $s(i,j) = \frac{2M(i,j)}{M(i,i)+M(j,j)}$ for all $x_j \in A_1$; and (3) sort $s(i,j)$ to return the top-k list in the final step. When n is very large, the vector-matrix computation will be too time consuming to check every possible object x_j. Therefore, we first select x_j's that are not orthogonal to x_i in the vector form, by following the links from x_i to find 2-step neighbors in commuting matrix $M_{\mathcal{P}}$, *i.e.*, $x_j \in CandidateSet = \{\bigcup_{y_k \in M_{\mathcal{P}}.neighbors(x_i)} M_{\mathcal{P}}^T.neighbors(y_k)\}$, where $M_{\mathcal{P}}.neighbors(x_i) = \{y_k | M_{\mathcal{P}}(x_i, y_k) \neq 0\}$, which can be easily obtained in the sparse matrix form of $M_{\mathcal{P}}$ that indexes both rows and columns. This will be much more efficient than pairwise comparison between the query and all the objects of that type. We call this baseline concatenation algorithm as *PathSim-baseline* (See Algorithm 2).

The *PathSim-baseline* algorithm, however, is still time consuming if the candidate set is very large. Although $M_{\mathcal{P}}$ can be relatively sparse given a short length meta path, after concatenation, M could be dense, *i.e.*, the $CandidateSet$ could be very large. Still, considering the query object and one candidate object represented by query vector and candidate vector, the dot product between them is proportional to the size of their non-zero elements. The time complexity for computing each candidate is $O(d)$ on average and $O(m)$ in the worst case, that is, $O(nm)$ in the worst case for all the candidates, where n is the row size of $M_{\mathcal{P}}$, *i.e.*, the number of objects in type A_1, and m the column size of $M_{\mathcal{P}}$, *i.e.*, the number of objects in type A_l, and d the average non-zero element for each object in $M_{\mathcal{P}}$. We now propose a co-clustering based top-k concatenation algorithm, by which non-promising target objects are dynamically filtered out to reduce the search space.

3.3 Co-Clustering Based Pruning

In the baseline algorithm, the computational costs involve two factors. First, the more candidates to check, the more time the algorithm will take; second, for each candidate, the dot product of query vector and candidate vector will at most involve m operations, where m is the vector length. The intuition to speed up the search is to prune unpromising candidate objects using simpler calculations. Based on the intuition, we propose a co-clustering (*i.e.*, clustering rows and columns of a matrix simultaneously) based path concatenation method, which first generates co-clusters of two types of objects for partial commuting matrix, then stores necessary statistics for each of the blocks corresponding to different co-cluster pairs, and then uses the block statistics to prune the search space. For better illustration, we call clusters of type A_1 as **target clusters**, since the objects in A_1 are the targets for the query; and call clusters of type A_l as **feature clusters**, since the objects in A_l serve as features to calculate the similarity between the query and the target objects. By partitioning A_1 into different target clusters, if a whole target cluster is not similar to the query, then all

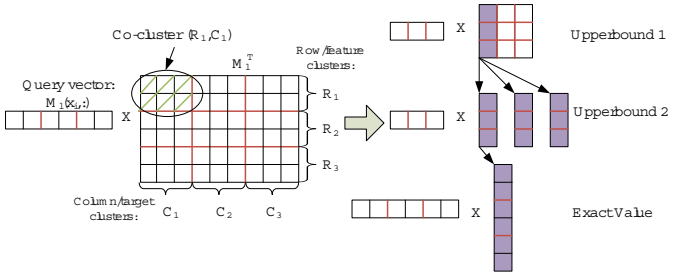


Figure 2: Illustration of Pruning Strategy

the objects in the target cluster are likely not in the final top-k lists and can be pruned. By partitioning A_l into different feature clusters, cheaper calculations on the dimension-reduced query vector and candidate vectors can be used to derive the similarity upper bounds.

This pruning idea is illustrated in Fig. 2 as follows. Given the partial commuting matrix M_l^T and its 3×3 co-clusters, and the query vector $M_l(x_i, :)$ for query object x_i, first the query vector is compressed into the aggregated query vector with the length of 3, and the upper bounds of the similarity between the query and all the 3 target clusters are calculated based on the aggregated query vector and aggregated cluster vectors; second, for each of the target clusters, if they cannot be pruned, calculate the upper bound of the similarity between the query and each of the 3 candidates within the cluster using aggregated vectors; third, if the candidates cannot be pruned, calculate the exact similarity using the non-aggregated query vector and candidate vectors.

The details of the co-clustering algorithm and the co-clustering based pruning algorithm, *PathSim-pruning* are introduced in Appendix A. Experiments show that *PathSim-Pruning* can significantly improve the query processing speed comparing with the baseline algorithm, without affecting the search quality.

4. EXPERIMENTS

For the experiments, we use the bibliographic network extracted from DBLP and the Flickr network to show the effectiveness of the PathSim measure and the efficiency (Appendix A.1) of the proposed algorithms.

4.1 Data Sets

We use the DBLP dataset downloaded in Nov. 2009. It contains over $710K$ authors, $1.2M$ papers, $5K$ venues (conferences/journals), and after removing stopwords in paper titles, we get around $70K$ terms that appear more than once. Our DBLP networks are built according to the network schema introduced in Example 2 except that there is no direct link between papers since DBLP provides very limited citation information. This dataset is referred as "full DBLP dataset". Two small subsets of the data (to alleviate the high computational costs of P-PageRank and SimRank) are used for the comparison with other similarity measures in effectiveness: (1) "DBIS dataset", which contains all the 464 venues and top-5000 authors from the database and information system area; and (2) "4-area dataset", which contains 20 venues and top-5000 authors from 4 areas: *database, data mining, machine learning* and *information retrieval* [17], and cluster labels are given for all the 20 venues and a subset of authors that include 1713 authors.

For additional case studies (Appendix C), we construct a Flickr network from a subset of the Flickr data, which contains four types of objects: images, users, tags, and groups. Links exist between images and users, images and tags, and images and groups. We use 10,000 images from 20 groups as well as their related 664 users and 10284 tags appearing more than once to construct the network.

Table 5: Top-15 query results accuracy for five similarity measures on "DBIS dataset" measured by nDCG

	P-PageRank	SimRank	RW	PRW	PathSim
Accuracy	0.5552	0.6289	0.7061	0.5284	**0.7446**

4.2 Effectiveness

1. Comparing *PathSim* with other measures

When a meta path $\mathcal{P} = (\mathcal{P}_l \mathcal{P}_l^{-1})$ is given, other measures such as random walk (RW) and pairwise random walk (PRW) can be applied on the same meta path, and P-PageRank and SimRank can be applied on the sub-network extracted from $\mathcal{P}$. For example, for the meta path $CPAPC$ (CAC in short) for finding venues sharing the same set of authors, the bipartite graph M_{CA}, derived from commuting matrix corresponding to CPA can be used in both P-PageRank and SimRank algorithms. In our experiments, the damping factor for P-PageRank is set as 0.9 and SimRank is set as 0.8.

First, a case study is shown in Table 4, which is applied on "DBIS dataset", under the meta path CAC. One can see that for query "PKDD" (short for "Principles and Practice of Knowledge Discovery in Databases", a European data mining conference), P-PageRank favors venues with higher visibility, such as KDD and several well-known venues; SimRank prefers concentrated venues (*i.e.*, a large portion of publications goes to a small set of authors) and returns many not well-known venues such as "Local Pattern Detection" and KDID; RW also favors highly visible objects such as KDD, but brings in fewer irrelevant venues due to that it utilizes merely one short meta path; PRW performs similar to SimRank, but brings in more not so well-known venues due to the short meta path it uses; whereas PathSim gives venues with similar area as well as similar reputation as PKDD, such as ICDM and SDM.

We then labeled top-15 results for 15 queries from venue type (SIGMOD, VLDB, ICDE, PODS, EDBT, DASFAA, KDD, ICDM, PKDD, SDM, PAKDD, WWW, SIGIR, TREC and APWeb) in "DBIS dataset", to test the quality of the ranking lists given by 5 measures. We label each result object with relevance score as three levels: 0–non-relevant, 1–some-relevant, and 2–very-relevant. Then we use the measure nDCG (Normalized Discounted Cumulative Gain, with the value between 0 and 1, the higher the better) [8] to evaluate the quality of a ranking algorithm by comparing its output ranking results with the labeled ones (See Table 5). The results show that PathSim gives the best ranking quality in terms of human intuition, which is also consistent with the previous case study.

Next, we study the performance of different single meta path-based similarity measures, including *PathSim*, RW, and PRW, in the task of clustering, where these measures use exactly the same information to determine the pairwise similarity between objects. Note the clustering problem is rather different from node-oriented similarity search but can still be used to roughly compare the sensitivity of the similarity measures. We use "4-area dataset" to evaluate the clustering performance, since this dataset naturally has 4 clusters, under the meta path CAC for venues and ACA for authors. We apply Normalized Cut [14] to the 3 similarity matrices, and use NMI (Normalized Mutual Information, with the value between 0 and 1, the higher the better) [15] to calculate the clustering accuracy for both venues and authors, and their weighted average accuracy over the two types. The average clustering accuracy results (based on 100 runs) for the venues-author network with different similarity measures are summarized in Table 6. It turns out that PathSim pro-

Table 4: Case study of five similarity measures on query"PKDD" on the "DBIS dataset"

Rank	P-PageRank	SimRank	RW	PRW	PathSim
1	PKDD	PKDD	PKDD	PKDD	PKDD
2	KDD	Local Pattern Detection	KDD	Local Pattern Detection	ICDM
3	ICDE	KDID	ICDM	DB Support for DM Appl.	SDM
4	VLDB	KDD	PAKDD	Constr.-Bsd. Min. & Induc. DB	PAKDD
5	SIGMOD	Large-Scale Paral. Data Min.	SDM	KDID	KDD
6	ICDM	SDM	TKDE	MCD	Data Min. Knowl. Disc.
7	TKDE	ICDM	SIGKDD Expl.	Pattern Detection and Discovery	SIGKDD Expl.
8	PAKDD	SIGKDD Expl.	ICDE	RSKD	Knowl. Inf. Syst.
9	SIGIR	Constr.-Bsd. Min. & Induc. DB	SEBD (Italian Sympo. on Adv. DB)	WImBI (Web Intell. Meets Brain Inf.)	J. Intell. Inf. Syst.
10	CIKM	TKDD	CIKM	Large-Scale Paral. Data Min.	KDID

Table 6: Clustering accuracy for single meta path-based similarity measures on "4-area dataset"

	RW	PRW	PathSim
Venue NMI	0.6159	**0.8198**	0.8116
Author NMI	0.6486	0.6364	**0.6501**
Weighted Avg. NMI	0.6485	0.6371	**0.6507**

Table 7: Top-10 most similar authors to "Christos Faloutsos" under different meta paths on "full DBLP dataset"

(a) Path: APA

Rank	Author
1	Christos Faloutsos
2	Spiros Papadimitriou
3	Jimeng Sun
4	Jia-Yu Pan
5	Agma J. M. Traina
6	Jure Leskovec
7	Caetano Traina Jr.
8	Hanghang Tong
9	Deepayan Chakrabarti
10	Flip Korn

(b) Path: $APCPA$

Rank	Author
1	Christos Faloutsos
2	Jiawei Han
3	Rakesh Agrawal
4	Jian Pei
5	Charu C. Aggarwal
6	H. V. Jagadish
7	Raghu Ramakrishnan
8	Nick Koudas
9	Surajit Chaudhuri
10	Divesh Srivastava

duces overall better performance in terms of weighted average of clustering accuracy in both types.

2. Semantic meanings of different meta paths

As we pointed out, different meta paths give different semantic meanings, which is one of the reasons that similarity definitions in homogeneous networks cannot be applied directly to heterogeneous networks. Besides the motivating example in the introduction section, Table 7 shows the author similarity under two scenarios for author Christos Faloutsos: *co-authoring papers* and *publishing papers in the same venues*, represented by the meta paths APA and $APCPA$ respectively. One can see that the first path returns co-authors who have strongest connections with Faloutsos (*e.g.*, students and close collaborators) in DBLP, whereas $APCPA$ returns those publishing papers in the most similar venues.

3. The impact of path length

The next interesting question is how the length of meta path impacts the similarity definition. Table 8 shows an example of venues similar to "SIGMOD" with three meta paths, using exactly the same basic meta path, but with different repeating times. These meta paths are $(CPAPC)^2$, $(CPAPC)^4$ and its infinity form (global ranking-based similarity). Notice that in $(CPAPC)^2$, two venues are similar if they share many similar authors who publish papers in *the same* venues; while in $(CPAPC)^4$, the similarity definition of those venues will be further relaxed, namely, two venues are similar if they share many similar authors who publish papers in *similar* venues. Since venue type only contains $5K$ venues, we are able to get the full materialization commuting matrix for $(CPAPC)^2$. $(CPAPC)^4$ is obtained using meta path concatenation from $(CPAPC)^2$. The results are summarized in Table 8, where longer path gradually bring in more remote neighbors, with higher similarity score, and finally it degenerates into global ranking comparison. Through this study, we can see that the meta path with relatively short length is good enough to measure similarity, and a long meta path may even reduce the quality.

Table 9: Impacts of length of meta path on clustering accuracy on the "4-area dataset"

	CAC	$(CAC)^2$	$(CAC)^3$
Venue NMI	**0.8116**	0.4603	0.4531
	ACA	$(ACA)^2$	$(ACA)^3$
Author NMI	**0.6501**	0.6091	0.5346

Table 9 shows that short meta paths produce better similarity measures in terms of clustering accuracy. We checked two other meta paths, namely $CPTPC$ and $APTPA$, which give the same conclusion.

5. RELATED WORK

Similarity measure has been widely studied in categorical, numerical, or mix-type data sets, such as cosine similarity defined on two vectors, Jaccard coefficient on two sets, and Euclidean distance on two numerical data points. Based on the traditional similarity measures, a recent study [19] proposes an efficient top-k similarity pair search algorithm, top-k-join, in relational database, which only considers similarity between tuples. Also widely studied are k nearest neighbor search in spatial data [10] and other high dimensional data [2], which aims at finding top-k nearest neighbors according to similarities defined on numerical features. However, these similarity definitions cannot be applied to networks.

Similarity measures defined on homogeneous networks emerged recently. Personalized PageRank [9] is an asymmetrical similarity measure that evaluates the probability starting from object x to visit object y by randomly walking on the network with restart. More discussions on how to scale the calculation for online queries are in [5, 18], *etc.*, and how to derive top-k answers efficiently is studied in [6]. SimRank [7] is a symmetric similarity measure defined on homogeneous networks, which can also be directly applied to bipartite networks. The intuition behind SimRank is propagating pairwise similarity to their neighboring pairs. Due to its computational complexity, there are many follow-up studies (*e.g.*, [11]) on speeding up such calculations. SCAN [20] measures similarity of two objects by comparing their immediate neighbor sets. ObjectRank [1] and PopRank [12] first noticed that heterogeneous relationships could affect the random walk, and assigned different propagation factors to each type of object relationship to either derive a revised version of P-PageRank (ObjectRank) or a global PageRank (PopRank). However, such solutions only give one particular combination of all the possible meta paths using the fixed weights determined by the damping factor and propagation factors

Table 8: Top-10 similar venues to "SIGMOD" under meta paths with different lengths on "full DBLP dataset"

(a) Path: $(CPAPC)^2$

Rank	Venue	Score
1	SIGMOD Conference	1
2	VLDB	0.981
3	ICDE	0.949
4	TKDE	0.650
5	SIGMOD Record	0.630
6	IEEE Data Eng. Bull.	0.530
7	PODS	0.467
8	ACM Trans. Database Syst.	0.429
9	EDBT	0.420
10	CIKM	0.410

(b) Path: $(CPAPC)^4$

Rank	Venue	Score
1	SIGMOD Conference	1
2	VLDB	0.997
3	ICDE	0.996
4	TKDE	0.787
5	SIGMOD Record	0.686
6	PODS	0.586
7	KDD	0.553
8	CIKM	0.540
9	IEEE Data Eng. Bull.	0.532
10	J. Comput. Syst. Sci	0.463

(c) Path: $(CPAPC)^\infty$

Rank	Venue	Score
1	SIGMOD Conference	1
2	AAAI	0.9999
3	ESA	0.9999
4	IEEE Trans. on Commun.	0.9999
5	STACS	0.9997
6	PODC	0.9996
7	NIPS	0.9993
8	Comput. Geom.	0.9992
9	ICC	0.9991
10	ICDE	0.9984

between different types. In our PathSim definition, users can freely specify the meta paths they are interested in and assign any weight to them. Random walk style similarity search is not adopted in PathSim, which overcomes the disadvantage of returning highly ranked objects rather than similar peers.

6. DISCUSSIONS

In this study, we assume that users know how to choose meta path. In practice, there are several ways for a user to select the best meta path or meta path combinations. First, a user can make a choice based on her interest and domain knowledge. Second, she can have several experimental trials, such as those done in Section 4, and choose the best one according to her intuition. Third, she can label a small portion of data according to specific applications. For example, one can label similar objects or rank them, and then train the best meta path(s) and their weights by some learning algorithms. By doing so, one can automatically choose appropriate meta paths as well as the associated weights, and make the similarity search adaptable to different application scenarios. The problem on how to choose and weight different meta paths is similar to the feature selection process in machine learning. In-depth study for a systematic solution is left as a future research task.

7. CONCLUSIONS

We have introduced a novel and practical notion of *meta path-based similarity for heterogeneous information networks*. We comparatively and systematically examine different semantics of similarity measures in such networks and introduce a new meta path-based similarity measure to find similar objects of the same type in such networks. Meta paths give users flexibility to choose different meta paths and their combinations based on their applications. Moreover, we propose a new similarity measure, PathSim, under this framework, which produces overall better similarity qualities than the existing measures. Since meta paths can be arbitrarily given, it is unrealistic to fully materialize all the possible similarity results given different meta paths and their combinations. However, online calculation requires matrix multiplication, which is time consuming especially when the vector and matrix are not sparse. Therefore, we proposed an efficient solution that partially materializes several short meta paths and then applies online concatenation and combination among paths to give the top-k results for a query. Experiments on real data sets show the effectiveness of the similarity measure and the efficiency of our method. The framework of meta path-based similarity search in networks can be enhanced in many ways, *e.g.,* weight learning for different meta paths, which may help provide accurate similarity measures in real systems and discover interesting relationships among objects.

8. REFERENCES

[1] A. Balmin, V. Hristidis, and Y. Papakonstantinou. Objectrank: authority-based keyword search in databases. In *VLDB'04*.

[2] S. Berchtold, B. Ertl, D. A. Keim, H.-P. Kriegel, and T. Seidl. Fast nearest neighbor search in high-dimensional space. In *ICDE'98*.

[3] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *KDD'03*.

[4] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[5] D. Fogaras, B. Rácz, K. Csalogány, and T. Sarlós. Towards scaling fully personalized pageRank: algorithms, lower bounds, and experiments. *Internet Math.*, 2(3), 2005.

[6] M. Gupta, A. Pathak, and S. Chakrabarti. Fast algorithms for topk personalized pagerank queries. In *WWW'08*.

[7] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *KDD'02*

[8] K. Jarvelin and J. Kekalainen. Cumulated gain-based evaluation of IR techniques. *ACM TOIS 20(4), 2002*.

[9] G. Jeh and J. Widom. Scaling personalized web search. In *WWW'03*.

[10] M. Kolahdouzan and C. Shahabi. Voronoi-based k nearest neighbor search for spatial network databases. In *VLDB'04*.

[11] D. Lizorkin, P. Velikhov, M. Grinev, and D. Turdakov. Accuracy estimate and optimization techniques for simrank computation. *PVLDB*, 1(1):422–433, 2008.

[12] Z. Nie, Y. Zhang, J.-R. Wen, and W.-Y. Ma. Object-level ranking: bringing order to web objects. In *WWW'05*.

[13] F. Pan, X. Zhang, and W. Wang. Crd: fast co-clustering on large datasets utilizing sampling-based matrix decomposition. In *SIGMOD'08*.

[14] J. Shi, and J. Malik Normalized cuts and image segmentation. *IEEE Trans. on PAMI, 22(8), 2000*.

[15] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu. Rankclus: integrating clustering with ranking for heterogeneous information network analysis. In *EDBT'09*.

[16] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS'01*.

[17] Y. Sun, J. Han, J. Gao, and Y. Yu. iTopicModel: Information Network-Integrated Topic Modeling. In *ICDM'09*.

[18] H. Tong, C. Faloutsos, J. Pan. Fast Random Walk with Restart and Its Applications. In *ICDM'06*.

[19] C. Xiao, W. Wang, X. Lin, and H. Shang. Top-k set similarity joins. In *ICDE'09*.

[20] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger. Scan: a structural clustering algorithm for networks. In *KDD'07*.

APPENDIX

In this Appendix, we present the technical details of the co-clustering based pruning algorithm, examine the case of multiple meta path combination, show an additional experiment on the Flickr network, and finally present the proofs of the theorems.

A. DETAILS OF CO-CLUSTERING BASED PRUNING ALGORITHM

1. Block-wise Commuting Matrix Materialization

The first problem is how to generate these clusters for each commuting matrix $M_{\mathcal{P}}$. Since one commuting matrix can be used for the concatenation into two longer meta paths, *i.e.*, $M_{\mathcal{P}}M_{\mathcal{P}}^T$ and $M_{\mathcal{P}}^T M_{\mathcal{P}}$, we hope to find co-clusters of feature cluster and target cluster, within which all values are similar to each other. We use a greedy KL-divergence based co-clustering method (summarized in Algorithm 1), which is similar to the information-theoretic co-clustering proposed in [3], but simplifies the feature space for each object by merely using the feature cluster information. For example, for $\mathcal{P}_l = (APC)$, we will use the conditional probability of author clusters appearing in some conference c, say $p(\hat{A}_u|c =$ "$VLDB$"), as the feature for conference c, use the conditional probability of author clusters in some conference cluster $\hat{C}_v$, say $p(\hat{A}_u|\hat{C}_v =$ "DB"), as the feature for conference cluster $\hat{C}_v$, and assign the conference to the conference cluster with the minimum KL-divergence. The adjustment is the same for author type given current conference clusters. The whole process is repeated for conference type and author type alternately, until the clusters do not change any more.

The time complexity of Algorithm 1 is $O(t(m+n)(UV))$, where t is the number of iterations, m and n are the number of objects for feature type and target type, U and V are the numbers of clusters for feature type and target type. Compared with the original $O(mn(U+V))$ algorithm in [3], it is much more efficient. Sampling-based variation algorithm such as in [13] can be applied for further faster co-clustering. In our experiment setting, we will select objects with higher degrees for the clustering, and assign those with smaller degrees to the existing clusters.

Algorithm 1 Greedy Co-Clustering Algorithm

Input: Commuting Matrix $M_{\mathcal{P}}^T$, number of feature clusters (row clusters) U, number of target clusters (column clusters) V
Output: row clusters $\{R_u\}_{u=1}^U$, column clusters $\{C_v\}_{v=1}^V$

```
//Initialization.
Randomly assign row objects into {R_u}_{u=1}^U;
Randomly assign column objects into {C_v}_{v=1}^V;
repeat
  //get center vector of each R_u:
  f(R_u) = 1/|R_u| Σ_{v=1}^V M_P^T(R_u, C_v);
  //Adjust row objects
  foreach object x_i in row objects do
    f(x_i) = Σ_{v=1}^V M_P^T(x_i, C_v);
    assign x_i into R_u, u = arg min_k KL(f(x_i)||f(R_u));
  end for
  //get center vector of each C_v:
  f(C_v) = 1/|C_v| Σ_{u=1}^U M_P^T(R_u, C_v)
  //Adjust column objects
  foreach object y_j in row objects do
    f(y_j) = Σ_{u=1}^U M_P(R_u, y_j);
    assign y_j into C_v, v = arg min_l KL(f(y_j)||f(C_v));
  end for
until {R_u}, {C_v} do not change significantly.
```

Once the clusters for each type of objects are obtained, the commuting matrix can be decomposed into disjoint blocks. To facilitate further concatenation on two meta paths for queries, necessary statistical information is stored for each block. For each block b denoted by row cluster R_u and column cluster C_v, we store:

1. Element sum of each block $T^{\{U\times V\}}$: $t_{uv} = \sum_{i\in R_u}\sum_{j\in C_v} M_{\mathcal{P}}^T(i,j)$;
2. Sum of row vectors (1-norm of each column vector) of each block $T_1^{\{U\times m\}}$: $t_{uv,1}(j) = \sum_{i\in R_u} M_{\mathcal{P}}^T(i,j)$, for $j \in C_v$;
3. Square root of sum of square of row vectors (2-norm of each column vector) of each block $TT_1^{\{U\times m\}}$: $t^2_{uv,1}(j) = \sqrt{\sum_{i\in R_u}(M_{\mathcal{P}}^T(i,j))^2}$, for $j \in C_v$;
4. Sum of column vectors (1-norm of each row vector) of each block $T_2^{\{n\times V\}}$: $t_{uv,2}(i) = \sum_{j\in C_v} M_{\mathcal{P}}^T(i,j)$, for $i \in R_u$;
5. Square root of sum of square of column vectors (2-norm of each row vector) of each block $TT_2^{\{n\times V\}}$: $t^2_{uv,2}(i) = \sqrt{\sum_{j\in C_v}(M_{\mathcal{P}}^T(i,j))^2}$, for $i \in R_u$.

2. Pruning Strategy in Path Concatenation

Now let's focus on how we can get top-k results efficiently for a query given the materialized block-wise commuting matrix. The intuition is that we first check the most promising target cluster, then if possible, prune the whole target cluster; if not, we first use simple calculations to decide whether we need to further calculate the similarity between the query and the candidate object, then compute the exact similarity value using more complex operations only for those needed.

THEOREM 3. ***Bounds for block-based similarity measure approximation.*** *Given a query object x, the query vector is $\mathbf{x} = M_{\mathcal{P}}(x,:)$. Let D be the diagonal vector of M, let $\hat{\mathbf{x}}_1$ be the compressed query vector given feature clusters $\{R_u\}_{u=1}^U$, where $\hat{\mathbf{x}}_1(u) = \max_{j\in R_u}\{\mathbf{x}(j)\}$, and let $\hat{\mathbf{x}}_2$ be the 2-norm query vector given feature clusters R_u, where $\hat{\mathbf{x}}_2(u) = \sqrt{\sum_{j\in R_u}\mathbf{x}(j)^2}$, the similarity between x and target cluster C_v, and the similarity between x and candidate $y \in C_v$ can be estimated using the following upper bounds:*

1. *upperbound 1:*
   $\forall y \in C_v, s(x,y) \le s(x,C_v) = \sum_{y\in C_v} s(x,y) \le \frac{2\hat{\mathbf{x}}_1^T T(:,v)}{D(x)+1}$;
2. *upperbound 2:*
   $\forall y \in C_v, s(x,y) \le \frac{2\hat{\mathbf{x}}_2^T TT_1(:,y)}{D(x)+D(y)}$.

PROOF. See Proof in the Appendix D. □

In Theorem 3, the upper bound for $s(x, C_v)$ can be used to find the most promising target clusters as well as to prune target clusters if it is smaller than the lowest similarity in the current top-k results. The upper bound for $s(x, y)$ can be used to prune target objects that are not promising, which only needs at most U times calculation, whereas the exact calculation needs at most m times calculation. Here, U is the number of feature clusters and m is the number of feature objects, *i.e.*, objects of type A_l.

The search strategy is to first sort the target clusters according to their upper bound of the similarity between the query x and the cluster C_v, *i.e.*, $s(x, C_v)$, in a decreasing order. The higher the similarity the more likely this cluster contains more similar objects to x. It is very critical to use the order to check the most promising target clusters first, by which the most desirable objects are retrieved at an early stage and the upper bounds then have stronger power to

prune the remaining candidates. When a new target cluster needs to be checked, the upper bound can be used to prune the whole target cluster and all the remaining target clusters, if it is smaller than the k-th value of the current top-k list. Next, when going to check the candidates within the target cluster, the upper bound between query object x and candidate y can be used to prune non-promising candidates if it is smaller than the current threshold. The algorithm *PathSim-pruning* is summarized in Algorithm 3. On Line 5, $min(S)$ is the lowest similarity in the current top-k result set S. Similar to *PathSim-baseline* (Algorithm 2), before the pruning steps, we still need to first derive the candidate set. Compared with the baseline algorithm, the pruning-based algorithm at most checks the same number of candidates with the overhead to calculate the upper bounds. In practice, a great number of candidates can be pruned, and therefore the performance can be enhanced.

Algorithm 2 (PathSim-Baseline) Vector-Matrix Multiplication Based Path Concatenation

Input: Query x_i, Commuting Matrix $M_{\mathcal{P}}$, Diagonal Vector D, top-k K
Output: Top-k List $SortList$

1: $CandidateSet = \emptyset$;
2: **foreach** $y_k \in M_{\mathcal{P}}.neighbors(x_i)$ **do**
3: **foreach** $x_j \in M_{\mathcal{P}}^T.neighbors(y_k)$ **do**
4: $CandidateSet = CandidateSet \cup \{x_j\}$;
5: **end for**
6: **end for**
7: $List = \emptyset$;
8: **foreach** $x_j \in CandidateSet$ **do**
9: $value = 2 * M_{\mathcal{P}}(i,:)M_{\mathcal{P}}(j,:)^T/(D(i) + D(j))$;
10: $List.update(x_j, value, K)$;
11: **end for**
12: $List.sort()$;
13: $SortList = List.topk(K)$;
14: return $SortList$;

Algorithm 3 (PathSim-Pruning) Cluster-based Top-k Search on Path Concatenation

Input: Query x_i, Commuting matrix $M_{\mathcal{P}}^T$, Feature clusters $\{R_u\}_{u=1}^U$, Target clusters $\{C_v\}_{v=1}^V$, Diagonal vector D, top-k K.
Output: Top-k list S.

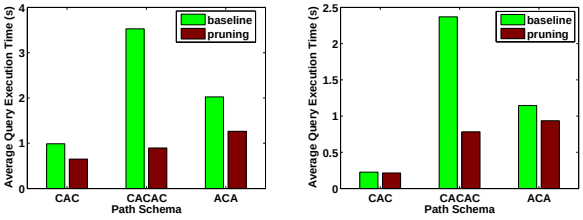
1: Set $CandidateSet = x_i.neighbors.neighbors$;
2: $S = \varnothing$;
3: Sort clusters in $\{C_v\}_{v=1}^V$ according to upper bound of $s(x_i, C_v)$;
4: **foreach** C_v with decreasing order **do**
5: **if** the upper bound of $s(x_i, C_v) < min(S)$ **then**
6: break;
7: **else**
8: **foreach** $x_j \in C_v$ and $x_j \in CandidateSet$ **do**
9: **if** the upper bound of $s(x_i, x_j) < min(S)$ **then**
10: continue;
11: **else**
12: $s(x_i, x_j) = \frac{2M_{\mathcal{P}}(x_i,:)(M_{\mathcal{P}}(x_j,:))^T}{D(x_i)+D(x_j)}$;
13: Insert x_j into S;
14: **end if**
15: **end for**
16: **end if**
17: **end for**
18: return S;

A.1 Efficiency Comparison

The time complexity for SimRank is $O(KN^2d^2)$, where K is the number of iterations, N is the total number of objects, and d is the average neighbor size; the time complexity for calculating P-PageRank for one query is $O(KNd)$, where K, N, d has the same meaning as in SimRank; whereas the time complexity for PathSim using $PathSim\text{-}baseline$ for single query is $O(nd)$, where $n < N$ is the number of objects in the target type, d is the average degree of objects in target type for partial commuting matrix $M_{\mathcal{P}_l}$. The time complexity for RW and PRW are the same as PathSim. We can see that similarity measure only using one meta path is much more efficient than those also using longer meta paths in the network (*e.g.,* SimRank and P-PageRank).

In this sub-section, two algorithms proposed in Section 3, *i.e.,* *PathSim-baseline* and *PathSim-pruning*, are compared, for efficiency study under different meta paths, namely, $CPAPC$, $(CPAPC)^2$ and $APCPA$ (denoted as CAC, $CACAC$ and ACA for short). For the co-clustering algorithm, the number of clusters for authors is set as 50, and that for conferences as 20. It is easy to see that the more clusters used, the more accurate the upper bounds would be, however the longer the calculation for the upper bounds would be. A trade-off should be made to decide the best number of clusters. Due to the limited space, we do not discuss the issue in this paper.

First, we check the impacts of the number of neighbors of the query on the execution time. Note, a query object with higher degree usually leads to larger number of neighbors. Therefore, two test query sets are selected based on their degrees to test the execution time for each meta path: one is of top-20 objects and the other is of 1001th-1020th objects according to their link degrees. We compare the performance of the two algorithms under three meta paths. Each query is executed 5 times and the output time is the total average execution time within each query set, and the results are summarized in Figure 3. From the results, one can see (1) *PathSim-pruning* is more efficient than *PathSim-baseline*; (2) the improvement rate depends on the meta path, the denser the corresponding commuting matrix, the higher rate *PathSim-pruning* can improve; and (3) the improvement rate also depends on the queries, the more neighbors of a query, the higher rate *PathSim-pruning* can improve. In Figure 4, we compare the efficiency under different top-k's ($k = 5, 10, 20$) for *PathSim-pruning* using query set 1. Intuitively, a smaller top-k has stronger pruning power, and thus needs less execution time, as demonstrated.



(a) Test Query Set 1 (1-20) (b) Test Query Set 2 (1001-1020)

Figure 3: *PathSim-baseline* vs. *PathSim-pruning* on "full DBLP dataset"

Now we compare the pruning power of *PathSim-pruning* vs. *PathSim-baseline* by considering two factors: the size of the neighbors of a query (Fig. 5) and the density of the partial commuting matrix $M_{\mathcal{P}}$ (Fig. 6). 500 queries are randomly chosen for two meta paths (CAC and $CACAC$), and the execution time is averaged with 10 runs. The results show that the execution time for *PathSim-baseline* is almost linear to the size of the candidate set, and the improvement rate for *PathSim-pruning* is larger for queries with more neighbors, which requires more calculation for exact dot product operation between a query vector and candidate vectors. Also, the denser that the commuting matrix corresponding to the partial meta path (M_{CPAPC} in comparison with M_{CPA}), the greater the prun-

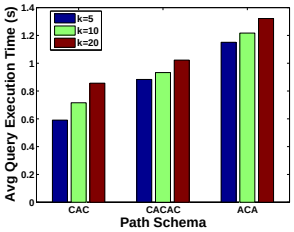
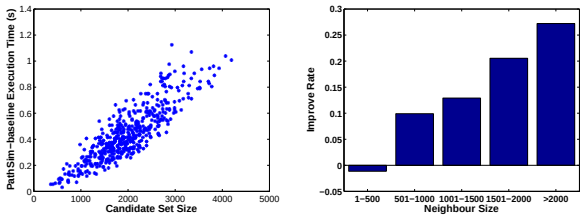


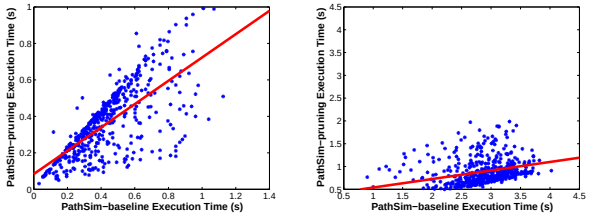
Figure 4: Average query execution time given different top-k's on "full DBLP dataset" using *PathSim-pruning*

ing power. The improvement rates are 18.23% and 68.04% for the two meta paths.



(a) Candidate size vs. baseline execution time (b) Neighbor size vs. improvement rate

Figure 5: Efficiency study for queries with different neighbor size under meta path CAC on "full DBLP dataset" based on 500 queries



(a) Meta path: CAC (b) Meta path: $CACAC$

Figure 6: Pruning power denoted by the slope of the red fitting line under two meta paths for type conference on "full DBLP dataset"

B. MULTIPLE META PATH COMBINATION

In Section 3, we presented algorithms for similarity search in single meta path. Now, we present a solution to combine multiple meta paths together. Formally, given r round trip meta paths from Type A back to Type A, $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_r$, and their corresponding commuting matrix $M_1, M_2, \ldots, M_r$, with weights $w_1, w_2, \ldots, w_r$ specified by users, the combined similarity between objects $x_i, x_j \in A$ are defined as: $s_{comb}(x_i, x_j) = \sum_{l=1}^{r} w_l s_l(x_i, x_j)$, where $s_l(x_i, x_j) = \frac{2M_l(i,j)}{M_l(i,i)+M_l(j,j)}$, as defined by PathSim in Definition 4.

EXAMPLE 4. *Following the motivating example in the introduction section, Table 10 shows the results of combining two meta paths $\mathcal{P}_1 = CPAPC$ and $\mathcal{P}_2 = CPTPC$ with different weights specified by w_1 and w_2, for query "DASFAA".* ∎

Table 10: "DASFAA" with multiple meta paths

Rank	w_1=0.2,w_2=0.8	w_1=0.5,w_2=0.5	w_1=0.8,w_2=0.2
1	DASFAA	DASFAA	DASFAA
2	Data Knowl. Eng.	DEXA	DEXA
3	CIKM	CIKM	WAIM
4	EDBT	Data Knowl. Eng.	CIKM
5	Inf. Syst.	EDBT	APWeb

Table 11: Clustering accuracy for PathSim for meta path combinations on "DBIS dataset"

w_1	0	0.2	0.4	0.6	0.8	1
w_2	1	0.8	0.6	0.4	0.2	0
$CAC; CTC$	0.7917	0.7936	**0.8299**	**0.8587**	0.8123	0.8116
$ACA; (ACA)^2$	0.6091	0.6219	0.6506	**0.6561**	**0.6508**	0.6501

The reason why we need to combine several meta paths is that, each meta path provides a unique angle (or a unique feature space) to view the similarity between objects, and the ground truth may be a cause of different factors. Some useful guidance of the weight assignment includes: longer path utilizes more remote relationships and thus should be assigned with a smaller weight, such as in P-PageRank and SimRank; and, meta path with more important relationships should be assigned with a higher weight. For automatically determining the weights, users could provide training examples of similar objects to learn the weights of different meta paths using machine learning algorithms.

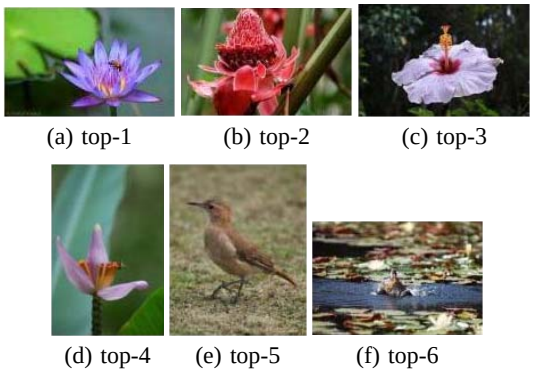
Since each meta path plays an independent role to decide the similarity, we can calculate top list for each of them and then combine the results together. The critical problem is how to determine if the remaining candidate objects are not going to appear in the final top-k list, and thus can be safely removed. The general idea for the search strategy is: (1) get top-k' objects for each meta path using single path top-k search algorithm, where k' should be larger than k (*e.g.*, in a order of $2k$, $4k$, $8k$, and so on); (2) check for each meta path, whether current top-k' objects can guarantee higher similarity than the remaining ones, if not, expanding the top-k' list by recalculating the single meta path top-k' list with a bigger k' (*e.g.*, $k' = 2k'$); (3) repeat (2) until all the candidates generated for each meta path can guarantee they are in the final list; and (4) get the exact similarity score for each candidate and return top-k of the candidates. One possible upper bound for remaining objects other than those in the current top-k lists can be calculated as $upper_k = \sum_l w_l * TopKList[l].min$, where $TopKList[l].min$ stands for the lowest similarity score of Top-k for the l_{th} meta path. The FA and TA methods [16] could also be applied here, if the full ranking list is ready for each meta path using *PathSim-baseline*.

We now study the accuracy of combined meta paths using the "four-area dataset", evaluated by the clustering performance given the similarity matrix. First, two meta paths for the type conference, namely, CAC and CTC (short for $CPAPC$ and $CPTPC$), are selected and their linear combinations with different weights are considered. Second, two meta paths with the same basic path but different lengths, namely ACA and $(ACA)^2$, are selected and their linear combinations with different weights are considered. The clustering accuracy measured by NMI for conferences and authors is shown in Table 11, which shows that the combination of multiple meta paths can produce better similarity than the single meta path in terms of clustering accuracy.

C. CASE STUDY ON FLICKR NETWORK

In this case study, we show that one can merely use links in the network rather than any content information to retrieve similar im-

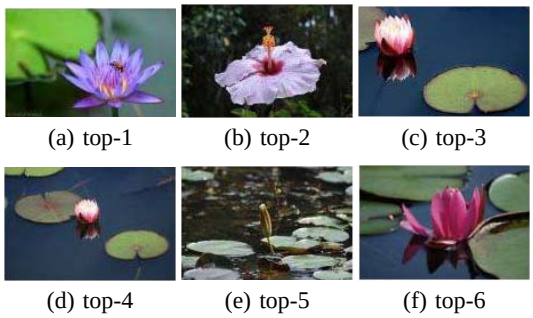
ages for a query image. Let "I" represent image, "T" tags that associated with each image, and "G" groups that each image belongs to. Two meta paths are used and compared. The first is ITI, which means common tags are used by two images at evaluation of their similarity. The results are shown in Fig. 7. The second meta path is $ITIGITI$, which means tags similarities are further measured by their shared groups, and two images could be similar even they do not share many exact same tags as long as these tags are used by many images of the same groups. One can see that the second meta path gives better results than the first one as shown in Fig. 8, where the first image is the input query. This is likely due to that the latter meta path provides additional information related to image groups, and thus improves the similarity measure between images.



(a) top-1 (b) top-2 (c) top-3

(d) top-4 (e) top-5 (f) top-6

Figure 7: Top-6 images in Flickr network under meta path ITI



(a) top-1 (b) top-2 (c) top-3

(d) top-4 (e) top-5 (f) top-6

Figure 8: Top-6 images in Flickr network under meta path $ITIGITI$

Discussion. The Flickr network is an interesting example that goes beyond the relational data. Our running example of bibliographic network can be viewed as a network constructed from relational data. So, naturally, it leads to two questions: (1) one may wonder whether the meta path-based top-k similarity search can be applied to relational databases. The answer to this question is "Yes", if we treat data from multiple relations as information networks. For example, to find the students most similar to a given student, one can view multiple relations as interconnected information networks and meta-paths to be selected can be based on the course taken, venues of the publications, advisors, or their weighted combinations. (2) One may also wonder whether the meta path-based similarity search can go far beyond the network formed based on relational data. This case study on the Flickr network shows that the analysis of heterogeneous information networks can go far beyond typical relational data since this network consists of links connecting photos with bags of terms and groups. There are many networks that cannot be constructed from relational data. For example, a news/blog network contains links among themes, categories, time, locations, authors, terms, pictures, and so on, beyond relational data. Similarity search in such networks can be readily handled under the framework presented in this study.

D. PROOFS OF THEOREMS

Here are the proofs of the theorems introduced in the previous sections.

Theorem 1: Properties of PathSim.

PROOF. (1) $s(x_i, x_j) = \frac{2M_{ij}}{M_{ii}+M_{jj}} = \frac{2M_{ji}}{M_{ii}+M_{jj}} = s(x_j, x_i)$, since $M_{ij} = M_{\mathcal{P}}(i,:) \cdot M_l(j,:) = M_l(j,:) \cdot M_l(i,:) = M_{ji}$, where $\cdot$ means the dot product of two vectors.

(2) Let $M_l(i,:) = (a_1, a_2, \ldots, a_p)$, $M_l(j,:) = (b_1, b_2, \ldots, b_p)$, easy to see a_k, b_k are nonnegative for all $1 \le k \le p$, then $M_{ij} = \sum_{k=1}^{p} a_k b_k \ge 0$, $M_{ii} = \sum_{k=1}^{p} a_k^2 > 0$ (no dangling object), and $M_{jj} = \sum_{k=1}^{p} b_k^2 > 0$, therefore $s(x_i, x_j) \ge 0$; also, $\sum_{k=1}^{p} a_k^2 + \sum_{k=1}^{p} b_k^2 \ge 2\sum_{k=1}^{p} a_k b_k$, with equality holding when $a_k = b_k$ for every k, therefore $s(x_i, x_j) \le 1$, and $s(x_i, x_i) = 1$.

(3) $M_{ij} = \sum_k a_k b_k \le \sqrt{\sum_k a_k^2 \sum_k b_k^2} = \sqrt{M_{ii}M_{jj}}$ (by Cauchy-Schwarz inequality), then $s(x_i, x_j) \le \frac{2}{\sqrt{M_{ii}/M_{jj}}+\sqrt{M_{jj}/M_{ii}}}$. □

Theorem 2: Property of PathSim under infinity length meta path.

PROOF. Since $M = (M_{\mathcal{P}} M_{\mathcal{P}}^T)$ is real symmetric, it can be decomposed as $M = PDP^T$, where D is a diagonal matrix with the values of eigenvalues of M, P is an orthogonal matrix composed of eigenvectors corresponding to eigenvalues in D. Let $\mathbf{r}$ be the first column in P, then $M^k = PD^kP^T$. Let $s_{ij}^{(k)} = \frac{2M^k(i,j)}{M^k(i,i)+M^k(j,j)}$, λ_1 be the largest eigenvalue of M, then $s_{ij}^{(k)} = \frac{2(PD^kP^T/\lambda_1^k)(i,j)}{(PD^kP^T(i,i)+PD^kP^T(j,j))/\lambda_1^k}$, and $\lim_{k\to\infty} s_{ij}^{(k)} = \frac{2\mathbf{r}(i)\mathbf{r}(j)}{\mathbf{r}(i)\mathbf{r}(i)+\mathbf{r}(j)\mathbf{r}(j)}$. □

Theorem 3: Bounds for block-based approximate similarity measure.

PROOF. 1. $\sum_{y\in C_v} s(x,y) = \sum_{y\in C_v} \frac{2\mathbf{x}^T\mathbf{y}}{D(x)+D(y)} \le \frac{2\mathbf{x}^T\sum_{y\in C_v}\mathbf{y}}{D(x)+1}$ $= \sum_u \frac{2\mathbf{x}(R_u)^T\sum_{y\in C_v}\mathbf{y}(R_u)}{D(x)+1} \le \sum_u \frac{2\hat{\mathbf{x}}_1(u)T(u,v)}{D(x)+1} = \frac{2\hat{\mathbf{x}}_1^T T(:,v)}{D(x)+1}$, since according to Holder's Inequality, $\mathbf{a}^T\mathbf{b} \le ||\mathbf{a}||_\infty ||\mathbf{b}||_1$.

2. $s(x,y) = \frac{2\mathbf{x}^T\mathbf{y}}{D(x)+D(y)} = \frac{2\sum_u \mathbf{x}(R_u)^T\mathbf{y}(R_u)}{D(x)+D(y)}$. Since $\mathbf{a}^T\mathbf{b} \le ||\mathbf{a}||_2||\mathbf{b}||_2$ according to Cauchy-Schwarz inequality, then the above formula $\le \frac{2\sum_u \hat{\mathbf{x}}_2(u)TT_1(u,y)}{D(x)+D(y)} = \frac{2\hat{\mathbf{x}}_2^T TT_1(:,y)}{D(x)+D(y)}$. □

E. ACKNOWLEDGEMENT

The authors would like to thank Lu Liu, Fabio Fumarola, Yintao Yu and Ziyu Guan for their valuable comments and suggestions.

The work was supported in part by U.S. National Science Foundation grants IIS-09-05215, the U.S. Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053 (NS-CTA), and MIAS, a DHS-IDS Center for Multimodal Information Access and Synthesis at UIUC. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.