Fast and Flexible Top-*k* Similarity Search on Large Networks

JING ZHANG, Tsinghua University, Renmin University of China
JIE TANG and CONG MA, Tsinghua University
HANGHANG TONG, Arizona State University
YU JING and JUANZI LI, Tsinghua University
WALTER LUYTEN and MARIE-FRANCINE MOENS, KU Leuven

Similarity search is a fundamental problem in network analysis and can be applied in many applications, such as collaborator recommendation in coauthor networks, friend recommendation in social networks, and relation prediction in medical information networks. In this article, we propose a sampling-based method using random paths to estimate the similarities based on both common neighbors and structural contexts efficiently in very large homogeneous or heterogeneous information networks. We give a theoretical guarantee that the sampling size depends on the error-bound ε, the confidence level $(1 - \delta)$, and the path length T of each random walk. We perform an extensive empirical study on a Tencent microblogging network of 1,000,000,000 edges. We show that our algorithm can return top-k similar vertices for any vertex in a network 300× faster than the state-of-the-art methods. We develop a prototype system of recommending similar authors to demonstrate the effectiveness of our method.

CCS Concepts: • **Information systems** → **Data mining**;

Additional Key Words and Pharses: Vertex similarity, similarity search, social network, random path, heterogeneous information network

ACM Reference Format:
Jing Zhang, Jie Tang, Cong Ma, Hanghang Tong, Yu Jing, Juanzi Li, Walter Luyten, and Marie-Francine Moens. 2017. Fast and Flexible Top-*k* Similarity Search on Large Networks. *ACM Trans. Inf. Syst.* 36, 2, Article 13 (August 2017), 30 pages.
http://dx.doi.org/10.1145/3086695

1 INTRODUCTION

Similarity search is a fundamental problem in network analysis and can be applied in many applications, such as collaborator recommendation in coauthor networks, friend recommendation in social networks, and drug-protein relation prediction in biological information networks.

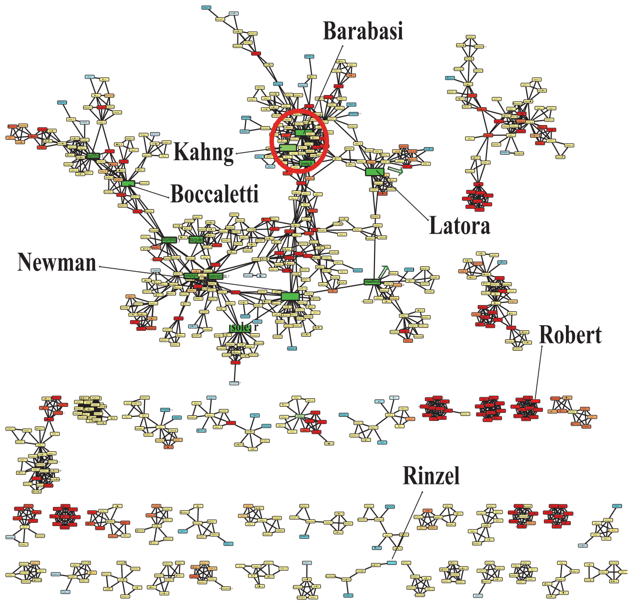
Authors' addresses: J. Zhang, Department of Computer Science and Technology, Tsinghua University, and Information School, Renmin University of China; email: zhang-jing@ruc.edu.cn; J. Tang (corresponding author), C. Ma, Y. Jing, and J. Li, Department of Computer Science and Technology, Tsinghua University, Beijing, China, 100084; emails: mac11@mails.tsinghua.edu.cn, {jietang, yujing5b5d, lijuanzi}@tsinghua.edu.cn; H. Tong, School of Computing, Informatics, and Decision Systems Engineering, ASU; email: hanghang.tong@asu.edu; W. Luyten and M.-F. Moens, Katholieke Universiteit Leuven, Leuven, Belgium; emails: walter.luyten@med.kuleuven.be, sien.moens@cs.kuleuven.be.

http://dx.doi.org/10.1145/3086695

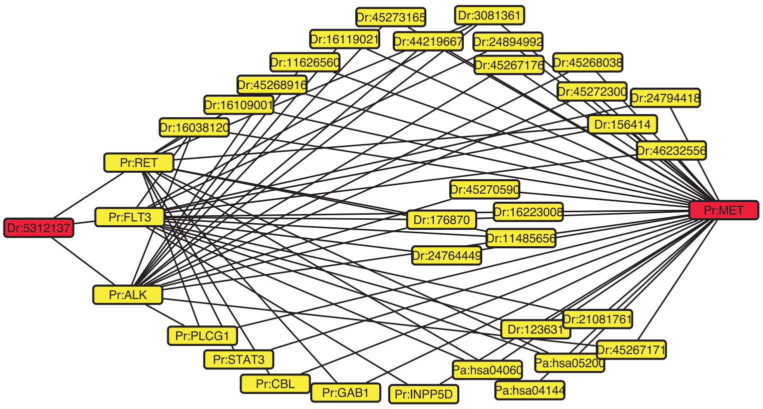
The problem has been extensively studied. In general, existing research work follows two basic principles. The first one is that two vertices are considered similar if they have many direct or indirect common neighbors in a network. For example, in a scientific coauthor network shown in Figure 1(a), we can say that the authors in the red circle are similar to Barabási, because they have more collaborations with Barabási than others. We category the methods following the first principle into neighborhood similarity. The Jaccard index [23] and cosine similarity [4] are two basic metrics of neighborhood similarity. However, they estimate the similarity in a local fashion. Although some work, such as SimRank [24] and VertexSim [34], use the entire network to compute similarity, they are essentially based on the transitivity of similarity in the network. Meanwhile, some researchers extend the neighborhood similarity to multityped neighborhood similarity based on different relation types in a heterogeneous information network [50]. The typed similarities provide extra semantic explanations for similarities. For example, in a medical information network shown in Figure 1(b), we can say that the drug named *5312137* is closely related with the protein named MET because they are connected by many paths of different semantics. Here we use "related" instead of "similar" because the type of the source vertex and the target vertex may not be the same in a heterogeneous information network. The second principle is that two vertices are considered equivalent if they play the same structural role—this can be further quantified by degree, closeness centrality, betweenness, and other network centrality metrics [15]. For example, in Figure 1(a), we can say that the authors denoted in green are similar to Barabási because they are all in the center places of certain subnetworks. Similarly, the authors denoted in red are similar to Robert because they are all in some tight-knit groups, and those denoted in blue are similar to Rinzel because they are in periphery places. We categorize the methods following the second principle into structure similarity. For example, RoleSim calculates the similarity between disconnected vertices by changing the initialization of SimRank [27], and ReFeX [20] is a feature-based method to calculate vertex similarity by defining a vector of features for each vertex.

When networks get larger, the efficiency issue has become one of the biggest challenges in similarity search, as most existing similarity methods in networks are iterative and have a high computation cost. For example, SimRank results in a complexity of $O(I|V|^2\bar{d}^2)$, where $|V|$ is the number of vertices in a network, $\bar{d}$ is the average degree of all vertices, and I is the number of iterations to perform the SimRank algorithm. It is clearly infeasible to apply SimRank to large-scale networks. For example, in our experiments, when dealing with a network with 500,000 edges, even the fast (top-k) version of SimRank [33] requires more than 5 days to complete the computation for all vertices (as shown later in Table 2). Although much effort has been made to improve the computational efficiency, such as fast random walk with restart (RWR) [16, 52] and fast SimRank [32, 33], they usually only focus on improving the efficiency of neighborhood similarity search in homogeneous information networks while ignoring other kinds of similarity search. Thus, in this article, we aim at designing a similarity method that is flexible enough to measure both neighborhood similarity and structure similarity quickly in large homogeneous or heterogeneous information networks.

To achieve the goal of neighborhood similarity search, we define a new similarity metric referred to as path similarity. The basic idea behind this is that two vertices have a high similarity if they frequently appear on the same paths. Then we propose a sampling-based method, referred to as Panther, based on a novel idea of *random path* to estimate path similarity. Specifically, given a network, we perform R random walks, each starting from a randomly picked vertex and walking T steps. The path similarities are calculated efficiently based on the generated paths and the inverted index from vertex to paths. We provide theoretical proof that the sample size, $R = \frac{c}{\varepsilon^2}(\log_2 \binom{T+1}{2} + 1 + \ln \frac{1}{\delta})$, only depends on the path length T of each random walk for a given error-bound ε and confidence level $1 - \delta$. Then we extend Panther to Panther$_m$ and Panther$_v$. Specially, Panther$_m$

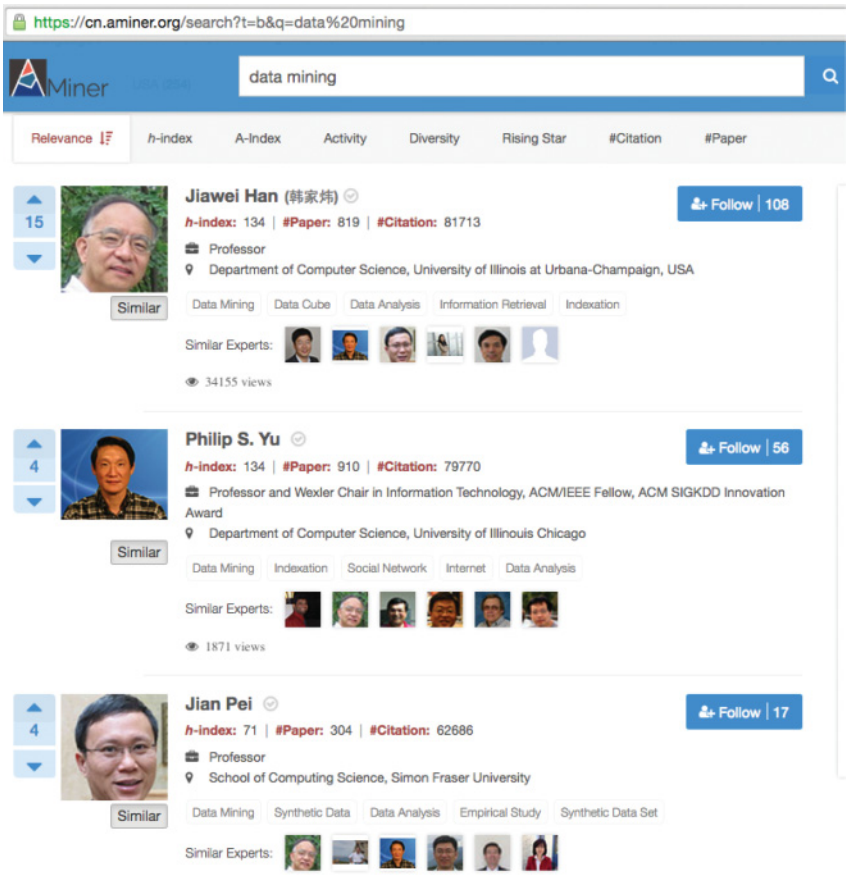


(a) Homogeneous information network.



(b) Heterogeneous information network.

Fig. 1. Case studies of similar nodes in a homogeneous information network (a) and a heterogeneous information network (b). We use a scientific coauthor network [38] as the case of a homogeneous information network. In the network, authors who have close collaborations with a given author can be treated as authors similar to the given author. In another viewpoint, similar authors can be also treated as those in positions similar to that of the given author. For example, the authors in a position similar to Barabási are denoted in green, those similar to Robert are in red, and those similar to Rinzel are in blue. We use a medical network as the case of a heterogeneous information network. In the network, the proteins with many paths connected with a drug can be treated as related proteins to the given drug. The prefix "Dr" in the vertex label denotes drug, "Pr" denotes protein, and "Pa" denotes pathway.


https://cn.aminer.org/search?t=b&q=data%20mining

15

Jiawei Han (韩家炜)

h-index: 134 | #Paper: 819 | #Citation: 81713

Professor

Department of Computer Science, University of Illinois at Urbana-Champaign, USA

Data Mining · Data Cube · Data Analysis · Information Retrieval · Indexation

Similar Experts:

34155 views

Follow | 108

Similar

4

Philip S. Yu

h-index: 134 | #Paper: 910 | #Citation: 79770

Professor and Wexler Chair in Information Technology, ACM/IEEE Fellow, ACM SIGKDD Innovation Award

Department of Computer Science, University of Illinouis Chicago

Data Mining · Indexation · Social Network · Internet · Data Analysis

Similar Experts:

1871 views

Follow | 56

Similar

4

Jian Pei

h-index: 71 | #Paper: 304 | #Citation: 62686

Professor

School of Computing Science, Simon Fraser University

Data Mining · Synthetic Data · Data Analysis · Empirical Study · Synthetic Data Set

Similar Experts:

Follow | 17

Similar

Fig. 2. Prototype system of similar expert recommendation in Aminer.org.

is used to estimate metapath-based path similarity in a heterogeneous information network, and Panther$_v$ is used to achieve the goal of structure similarity search.

We evaluate the efficiency of the proposed methods on a large microblogging "following" network from Tencent[1] and show the results later in Table 2. Clearly, our methods are much faster than the comparison methods. Panther$_v$ achieves a 300× speedup over the fastest comparison method on a Tencent subnetwork of 443,070 vertices and 5,000,000 edges. Our methods are also scalable. Panther is able to return top-k similar vertices for all vertices in a network with 51,640,620 vertices and 1,000,000,000 edges. On average, it only needs 0.0001 second to perform a top-k search for each vertex.

We build a prototype system to demonstrate the effectiveness of the Panther method. Specifically, in the system Aminer.org, for each searched expert we recommend similar authors. Figure 2 shows that authors similar to Professor Jiawei Han are Professor Xifeng Yan, Professor Philip Yu, and so on, all of whom share many direct or indirect coauthors with Professor Jiawei Han in the coauthor network. All codes and datasets used in this article are publicly available.[2]

[1]http://t.qq.com.
[2]https://cn.aminer.org/billboard/panther.

This article is an extension of prior work [59]. Compared to the prior work, we have the following new contributions: (1) definition of a new similarity metric in a heterogeneous information network, named *metapath similarity* (Section 3); (2) proposal of a sampling algorithm for calculating the new metric efficiently (Section 5); (3) empirical evaluation of effectiveness of the newly proposed algorithm for top-k similarity search in a large medical information network (Section 8.4.3); (4) development of a prototype system of similar expert recommendation in Aminer.org based on the proposed sampling method (Figure 2); and (5) adding new motivation of similarity search in heterogeneous information networks in Section 1, unified problem definition of three similarity metrics in Section 3, including path similarity, metapath similarity, and vector similarity, and the classified related work in Section 2.

Organization. This article is organized as follows. Section 2 reviews the related work. Section 3 formulates the problem and defines three similarity metrics. In Section 4, we detail the proposed method for top-k path similarity search and provide a theoretical analysis. Sections 5 and 6 introduce the extended methods for top-k metapath similarity search and vector similarity search, respectively. Section 7 compares our proposed methods to existing methods. Section 8 presents experimental results to validate the efficiency and effectiveness of our methods. Finally, Section 9 concludes the article.

2 RELATED WORK

In this section, we review the neighborhood similarity and structural similarity in homogeneous information networks and heterogeneous information networks, respectively.

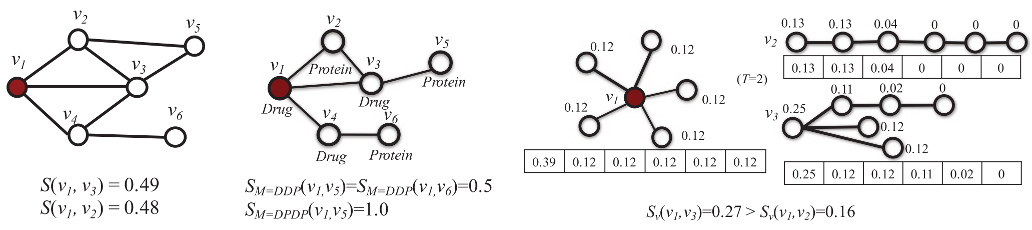
Neighborhood similarity in homogeneous information networks. Early neighborhood similarity measures, including bibliographical coupling [30] and cocitation [49], are based on the assumption that two vertices are similar if they have many common neighbors. However, they cannot estimate similarity between vertices without common neighbors. A direct consequence is that two nodes with no common neighbors will be treated as not similar at all. Thus, several measures have been proposed to address this problem. For example, Katz [29] counts two vertices as similar if there are more and shorter paths between them. RWR [39] measures the similarity between v_i and v_j as the steady-state probability that v_i will finally walk at v_j. Tsourakakis [53] learns a low-dimension vector for each vertex from the adjacent matrix and calculates similarities between the vectors. Jeh and Widom [24] propose SimRank, which follows a basic recursive intuition that two nodes are similar if they are referenced by similar nodes. Leicht et al. [34] develop an asymmetrical version of SimRank named vertex similarity. It is based on the assumption that two vertices are similar if any pair of their neighbors are similar. However, all SimRank-based methods share a common drawback: their computational complexities are too high. For example, SimRank requires $O(IN^2d^2)$ time and $O(N^2)$ space, where I is the number of iterations, N is the number of vertices, and d is the average degree over all vertices. Further studies have been done to reduce the computational complexity of SimRank [32, 33]. Fast random walk–based graph similarities, such as in other works [16, 46, 52], have also been studied. However, the efficiency can still be improved when the networks get larger.

Neighborhood similarity in heterogeneous information networks. Several researchers extend the neighborhood similarity to multityped neighborhood similarity in heterogeneous information networks. The state-of-the-art work is proposed by Sun et al. [50], who measure the similarities between vertices by enumerating all paths following a given metapath, where the metapath is used to represent the semantics of a path. Yun et al. [58] solve the problem of similarity join in a heterogeneous information network, whereas our work targets at solving the problem of top-k

similarity search. Shi et al. [48] propose HeteSim to conduct top-k similarity search in a heterogeneous information network. Their method also suffers from the efficiency issue, and thus the experimental dataset is not big enough. We extend our sampling-based method to solve metapath-based path similarity in a heterogeneous information network efficiently.

Structure similarity in homogeneous information networks. For structure similarity, Blondel et al. [5] provide a HITS-based recursive method to measure similarity between vertices across two different graphs. RoleSim [27] extends SimRank by allowing to calculate similarities between disconnected vertices. Similar to SimRank, the computational complexity of the two methods is very high. Feature-based methods can match vertices with similar structures. The basic idea is to define several features for each vertex and then calculate the Euclidean distance between feature vectors of two vertices as their structure similarity. For example, Burt [7] counts 36 kinds of triangles in one's ego network to represent a vertex's structural characteristics. In the same way, vertex centrality, closeness centrality, and betweenness centrality [14] of two different vertices can be compared to produce a structure similarity measure. However, any of the preceding metrics are too limited to explain the structural characteristic of a vertex. Aoyama et al. [3] present a fast method to estimate similarity search between objects instead of vertices in networks. ReFeX [19, 20] defines basic features such as degree, the number of within/out-egonet edges, and recursive features as the aggregated values of these features over neighbors. The computational complexity of ReFeX depends on the recursive times. Although they use a pruning technique to reduce the complexity, no theoretical proof is given to show how many recursive times is enough. More references about feature-based similarity search in networks can be found in the survey of Rossi and Ahmed [44]. Structure similarity in heterogeneous information networks cannot be directly or easily extended from the sampling-based method and thus will be studied in the future.

Graph sampling. Our problem of estimating top-k similar vertices is related to estimating the frequency of subgraph patterns in a network. Traditional research extensively studied how to enumerate the number of subgraph structures in a given graph, such as the number of triangles [25, 35, 41], the count of four-node subgraphs [26], and the macrofrequency and microfrequency of two-, three-, and four-node connected and disconnected subgraphs [45]. A few studies can be generalized to any type of subgraphs. For example, Kashtan et al. [28] and Rahman et al. [42] propose random edge enumeration algorithms to sample different kinds of subgraphs. Wernicke et al. [56] propose a random node enumeration algorithm to uniformly sample different subgraphs. Ahmed et al. [2] propose a general edge sampling framework to estimate the number of triangles, connected paths of length 2, clustering coefficient, and so on. The methods that are used to estimate frequency of any type of subgraphs can be used to solve our problem. However, the difference lies in that the weight of a path is also considered when counting paths in our problem. Duffield et al. [12] propose a weighted reservoir sampling method; however, they solve the problem of aggregation from data streams, and the weight is defined for the probability that discards the keys of aggregated data. The problem is totally different from the problem in this article—similarity search in a graph—and thus it is not clear how to directly use the method proposed in the work of Duffield et al. to solve our problem. Other work has been conducted on graph stream sampling. For example, Sarma et al. [47] uniformly sample nodes from graph streams to estimate PageRank scores. Buriol et al. [6] and Pavan et al. [40] estimate the number of triangles in graph streams. Cormode and Muthukrishnan [10] use a min-wise hash function to sample edges nearly uniformly to maintain the cascaded summaries of the graph stream. Aggarwal et al. [1] propose a structural reservoir sampling method for structural summarization. Graph stream sampling methods assume the edges arrive as a stream, and the target is to improve space and time complexity for fundamental



(a) Path similarity (b) Meta-path similarity (c) Vector similarity

Fig. 3. Illustration of similarity metrics. The vertices in red are source vertices. Others are in white. (a) Path similarity: the top two similar vertices of v_1 are v_3 and v_2. (b) Metapath similarity: following metapath DDP, the similarity between v_1 and v_5 is 0.5 and is 1.0 following metapath DPDP, where D represents drug and P represents protein. (c) Vector similarity: v_3 is more similar to v_1 than v_2.

problems. The graph is assumed to be static in this article. We will study how to calculate the proposed path similarity in graph streams in the future.

3 PROBLEM FORMULATION

In this section, we first define the problem of top-*k* similarity search and then define three similarity metrics that will be used in solving the problem: path similarity and its two extended similarity metrics, metapath similarity and vector similarity. Path similarity and metapath similarity are both neighborhood similarity metrics, and vector similarity belongs to the category of structure similarity.

Definition 3.1 (Information Network). Let $G = (V, E, W)$ denote a directed network, where V is a set of $|V|$ vertices and $E \subset V \times V$ is a set of $|E|$ edges between vertices. We use $v_i \in V$ to represent a vertex and $e_{ij} \in E$ to represent an edge from vertex v_i to vertex v_j. Let W be a weight matrix, with each element $w_{ij} \in W$ representing the weight associated with edge e_{ij}.

If there exists an edge from vertex v_i to v_j, we also build an inverse edge from v_j to v_i. We use $\mathcal{N}(v_i)$ to indicate the set of neighboring vertices of vertex v_i. Our purpose is to find the top-*k* similar vertices for any vertex in the network. Precisely, the problem can be defined as follows: given a network $G = (V, E, W)$ and a query vertex $v \in V$, how to find a set $X_{v,k}$ of k vertices that have the highest similarities to vertex v, where k is a positive integer and the similarity metric is defined as any of the following three ones.

Path similarity. We define a new similarity metric, referred to as *path similarity*. The basic idea of path similarity is that two vertices are similar if they frequently appear on the same paths. The principle is similar to that in Katz [29]. To begin with, we first define T-path as a sequence of vertices $p = (v_1, \ldots, v_{T+1})$, which consists of $T + 1$ vertices and T edges.[3] Let Π denote all T-paths in G, and let $w(p)$ be the weight of a path p. The weight can be defined in different ways. Given this, the path similarity between v_i and v_j is defined as follows:

$$S(v_i, v_j) = \frac{\sum_{p \in P(v_i, v_j)} w(p)}{\sum_{p \in \Pi} w(p)}, \tag{1}$$

where $P(v_i, v_j)$ is a subset of Π that contain both v_i and v_j. Figure 3(a) shows an example of path similarity. Considering vertex v_1, the ranking of the similarities with other vertices is

[3]Vertices in the same path are not necessary to be distinct.

$S(v_1, v_3) > S(v_1, v_2) > S(v_1, v_4) > S(v_1, v_5) > S(v_1, v_6)$. If calculating by SimRank [24], the ranking is $S(v_1, v_5) > S(v_1, v_6) > S(v_1, v_3) > S(v_1, v_2) > S(v_1, v_4)$. We can see among the first-order neighbors (i.e., v_2, v_3 and v_4) that v_1 is most similar to v_3 by both path similarity and SimRank, because according to path similarity, v_3 shares most paths with v_1, and according to SimRank, v_3 shares most neighbors with v_1. Among the second-order neighbors (i.e., v_5 and v_6), v_1 is more similar to v_5 by both the similarity metrics, and the reason is the same as that of the first-order neighbors. The difference is that SimRank ranks the second-order neighbors before the first-order neighbors, because according to SimRank, two directly connected vertices always treat each other as two different neighbors, and thus the similarity is weakened. However, path similarity counts paths of all lengths and ranks the first-order neighbors before the second-order neighbors, because first-order neighbors share more paths than the second-order neighbors.

Metapath similarity. The proposed path similarity is limited in homogeneous information networks, containing a single type of vertices and edges. Thus, path similarity cannot distinguish the semantics among paths that connect two vertices. However, in the real world, many networks are inherently heterogeneous, involving multiple types of vertices and edges, such as medical information networks, bibliographic networks, and social networks [31]. In those networks, the paths connecting two vertices present different semantics. In many cases, it is interesting to know the semantics of the paths, which may help us understand the reasons two vertices are closely related to each other. To distinguish the semantics among paths, we extend path similarity in homogeneous information networks to metapath-based path similarity in heterogeneous information networks. We first introduce the definitions of heterogeneous information network and metapath.

Definition 3.2 (Heterogeneous Information Network). A heterogeneous information network can be defined as a multityped directed network $G = (V, E, W; \phi, \mathcal{A}, \mathcal{R})$, where V, E, and W are the same notations as those in the information network. There is a vertex type mapping function $\phi : V \rightarrow \mathcal{A}$ with $\mathcal{A}$ as the set of vertex types—that is, each vertex $v \in V$ belongs to a particular vertex type in $\mathcal{A}$. Similarly, there is also an edge type mapping function $\psi : E \rightarrow \mathcal{R}$ with $\mathcal{R}$ as the set of edge types—that is, each edge $e \in E$ belongs to a particular edge type in $\mathcal{R}$.

Note that when there is only one vertex type and one edge type (i.e., $|\mathcal{A}| = 1$ and $|\mathcal{R}| = 1$), the network reduces to a homogeneous information network. If there exists an edge type R from vertex type $\mathcal{A}_i$ to $\mathcal{A}_j$, we also build an inverse relation of R from $\mathcal{A}_j$ to $\mathcal{A}_i$.

A typical example is a medical information network, with multiple types of vertices such as compounds/drugs, disease, proteins, side effects, and pathways. Accordingly, multiple types of directed edges can be defined between different types of vertices. For example, an express relationship from a drug to a protein can be represented by edge $(v_1, v_2) \in E$, where $v_1, v_2 \in \mathcal{V}$, $\phi(v_1) = Drug$, $\phi(v_2) = Protein$, and $\psi(v_1, v_2) = Drug \xrightarrow{express} Protein$; symmetrically, the edge (v_2, v_1) represents that the protein v_2 is expressed by the drug v_1.

Because of the multiple types for vertices and edges, the paths from one vertex to another can also be associated with multiple types. We use the concept of metapath [50] to represent the type/semantics of a path.

Definition 3.3 (Metapath). In a heterogeneous network G, a T-length metapath is an ordered sequence of T edge types connecting two vertices with type $\mathcal{A}_1$ and type $\mathcal{A}_{T+1}$, denoted by $\mathcal{M} = (\mathcal{A}_1 \xrightarrow{R_1} \mathcal{A}_2 \xrightarrow{R_2} \cdots \xrightarrow{R_T} \mathcal{A}_{T+1})$, where $\mathcal{A}_i \in \mathcal{A}$ and $\mathcal{R}_i \in \mathcal{R}$. An instantiation of $\mathcal{M}$ is a path in G, denoted by $p = (v_1 v_2 \ldots v_{T+1})$, satisfying $\phi(v_i) = \mathcal{A}_i, \forall i = 1, 2, \ldots, T+1$ and $\psi(v_i, v_{i+1}) = \mathcal{R}_i, \forall i = 1, 2, \ldots, T$. In addition, we represent a set of path instances following a metapath as $P_{\mathcal{M}}(v_i, v_j)$.

For example, in a medical information network, $\mathcal{M} = (Drug \xrightarrow{bind} Protein \xrightarrow{bind} Drug \xrightarrow{bind} Protein)$ represents a metapath, where $Drug, Protein \in \mathcal{A}$ and $Drug \xrightarrow{bind} Protein, Protein \xrightarrow{bind} Drug \in \mathcal{R}$. An instantiation of this metapath connects a drug and a protein by their directly connected protein and drug.

Given a metapath $\mathcal{M} = (\mathcal{A}_1 \xrightarrow{R_1} \mathcal{A}_2 \xrightarrow{R_2} \cdots \xrightarrow{R_T} \mathcal{A}_{T+1})$, the path space is changed to the T-paths instantiated by metapath $\mathcal{M}$ and can be denoted as $\Pi_{\mathcal{M}}$. Then metapath-based path similarity, abbreviated as metapath similarity, from v_i to v_j is defined as

$$S_{\mathcal{M}}(v_i, v_j) = \frac{\sum_{p \in P_{\mathcal{M}}(v_i, v_j)} w(p)}{\sum_{p \in \Pi_{\mathcal{M}}} w(p)}, \tag{2}$$

where $P_{\mathcal{M}}(v_i, v_j)$ is a subset of paths instantiated by metapath $\mathcal{M}$, between v_i and v_j.[4] Figure 3(b) shows an example of metapath similarity. When considering $Drug \xrightarrow{bind} Drug \xrightarrow{bind} Protein$ (DDP), the top similar vertices of v_1 are v_5 and v_6, whereas only v_5 is similar to v_1 when following the metapath $Drug \xrightarrow{bind} Protein \xrightarrow{bind} Drug \xrightarrow{bind} Protein$ (DPDP). In the figure, the paths with metapath DDP may be explained as drug v_1 can bind to the proteins v_5 or v_6 of v_1's similar drugs v_3 or v_4, and the paths with metapath DPDP may be explained as drug v_1 can bind to protein v_2, which shares another drug v_3 with protein v_5. Note that actually the preceding problem is to find the top-k related vertices instead of similar vertices, because the type of the source vertex and the target vertex may not be the same in a heterogeneous information network.

Vector similarity. One limitation of path similarity is that the estimated top-k similar vertices have a bias to close neighbors, although in principle it considers the structural information. We therefore present an extension of path similarity, referred to as *vector similarity*. The idea is to augment each vertex with a feature vector, which is expected to represent the structural characteristic of the vertex (e.g., star network vs. chain network) as opposed to the identity of neighbors of the vertex (e.g., connect to Bill Gates) [20]. To construct the feature vector, we follow the intuition that the topology structures of two vertices are similar to each other if the probabilities of the two vertices linking to all other vertices are similar to each other [21]. Given a vertex, the probability distribution can be represented by the top-D ranked similarities between it and other vertices, where D is an integer number:

$$\theta(v_i) = (S(v_i, v_{(1)}), S(v_i, v_{(2)}), \ldots, S(v_i, v_{(D)})), \tag{3}$$

where $S(v_i, v_{(d)})$ denotes the d-th largest similarity between v_i and another vertex $v_{(d)}$ and the similarity metric can be chosen as any kind of neighborhood similarities.

Naturally, the vector similarity between v_i and v_j can be calculated as the reciprocal Euclidean distance between their feature vectors:

$$S_v(v_i, v_j) = \frac{1}{\|\theta(v_i) - \theta(v_j)\|}. \tag{4}$$

Figure 3(c) shows an example of vector similarity. Based on the definition of vector similarity, we can see that v_3 is more similar to v_1 than v_2 because the structural characteristics of v_3 are more similar to those of v_1, even though the nodes are in two disconnected networks.

[4]Vertices in the same path are distinct to better present the semantics of a metapath.

4 PANTHER

We propose Panther to quickly conduct top-k similarity search based on the defined path similarity. A straightforward method to address the problem is to first calculate the similarity $S(v_i, v_j)$ between vertex v_i and any other vertex v_j in the network, and then select a set $X_{v,k}$ of k vertices that have the highest similarities to vertex v_i. However, it is generally difficult to scale up to large networks. One important idea is to obtain an approximate set $X^*_{v,k}$ for each vertex. From the accuracy perspective, we aim to guarantee that the difference between the two sets $X^*_{v,k}$ and $X_{v,k}$ is less than a threshold $\varepsilon \in (0,1)$—that is,

$$\text{Diff}(X^*_{v,k}, X_{v,k}) \leq \varepsilon,$$

with a probability of at least $1 - \delta$.

The difference between $X^*_{v,k}$ and $X_{v,k}$ can be also viewed as the error bound of the approximation. We propose a sampling-based method to approximate the top-k vertex similarity. In statistics, sampling is a widely used technique to estimate a target distribution [54]. Unlike traditional sampling methods, we propose a random path sampling method, named *Panther*, to estimate the predefined path similarity. We will explain in detail how the method can guarantee the error bound and how it is able to efficiently achieve the goal.

Random sampling. To calculate Equation (1), we need to enumerate all T-paths in G. However, the time complexity is exponentially proportional to the path length T and thus is inefficient when T increases. Therefore, we propose a sampling-based method to estimate the path similarity. Since path similarity between two vertices can be cast as estimating the probability that two vertices appear on a same path, our goal is to estimate the probability based on the sampled paths from the whole path space to guarantee a small error bound with a high probability. Specifically, we randomly sample R paths from the network and recalculate Equation (1) as

$$S(v_i, v_j) = \frac{\sum_{p \in P(v_i, v_j)} w(p)}{\sum_{p \in P} w(p)}, \tag{5}$$

where P is the set of sampled paths.

To generate a path, we randomly select a vertex v_i in G as the starting point and then conduct random walks of T steps from v_i using t_{ij} as the transition probability from vertex v_i to v_j:

$$t_{ij} = \frac{w_{ij}}{\sum_{v_k \in \mathcal{N}(v_i)} w_{ik}}, \tag{6}$$

where w_{ij} is the weight between v_i and v_j. In a unweighted network, the transition probability can be simplified as $1/|\mathcal{N}(v_i)|$.

We define $w(p)$ based on the random walk theory [13]:

$$w(p) = \prod_{i=1, j=i+1}^{T} t_{ij}.$$

The path weight also represents the probability that a path p is sampled from Π; thus, $w(p)$ in Equation (5) is absorbed in the random walk process. Therefore, we can rewrite the equation as follows:

$$S(v_i, v_j) = \frac{|P(v_i, v_j)|}{R}, \tag{7}$$

where R is the number of the sampled paths.

Algorithm 1 presents the details of searching top-k vertices for a given vertex. Algorithm 1 invokes algorithm 2. Algorithm 2 summarizes the process for generating R random paths. To calculate Equation (7), the time complexity is $O(RT)$ because it has to enumerate all R paths. To

```
ALGORITHM 1: Panther
Input: A network $G = (V, E, W)$, path length $T$, parameters $\varepsilon$, $c$, $\delta$, $k$ and a vertex $v$.
Output: top-$k$ similar vertices with regard to $v$.
Calculate sample size $R = \frac{c}{\varepsilon^2}(\log_2 \binom{T+1}{2} + 1 + \ln \frac{1}{\delta})$;
GenerateRandomPath($G$, $R$);
foreach $p_n \in P(v)$ do
    foreach Unique $v_j \in p_n$ do
        $S(v, v_j) += \frac{1}{R}$;
    end
end
Retrieve top-$k$ similar vertices according to $S(v, v_j)$.
```

```
ALGORITHM 2: GenerateRandomPath
Input: A network $G = (V, E, W)$ and sample size $R$.
Output: Paths $\{p_r\}_{r=1}^{R}$ and vertex-to-path index $\{P(v_i)\}_{i=1}^{N}$.
Calculate transition probabilities between every pair of vertices according to Equation (6);
Initialize $r = 0$;
repeat
    Sample a vertex $v = v_i$ uniformly at random;
    Add $v$ into $p_r$;
    Add $p_r$ into the path set of $v$, i.e., $P(v)$;
    repeat
        Randomly sample a neighbor $v_j$ according to transition probabilities from $v$ to its neighbors;
        Set current vertex $v = v_j$;
        Add $v$ into $p_r$ and add $p_r$ into $P(v)$;
    until $|p_r| = T + 1$;
    $r += 1$;
until $r = R$;
```

improve efficiency, we build an inverted index of vertex to path [4]. Using the index, we can retrieve all paths that contain a specific vertex v with a complexity of $O(1)$. Then Equation (7) can be calculated with a complexity of $O(\bar{R}T)$, where $\bar{R}$ is the average number of paths that contain a vertex and is proportional to the average degree $\bar{d}$. Details of the algorithm are presented in Algorithm 1, where lines 1 through 5 represent the preprocessing steps and line 6 refers to the top-k similarity searching for a vertex.

Theoretical analysis. We give a theoretical analysis for the random path sampling algorithm. In general, the path similarity can be viewed as a probability measure defined over all paths Π. Thus, we can adopt the results from Vapnik-Chernovenkis (VC) learning theory [54] to analyze the proposed sampling-based algorithm. To begin with, we will introduce some basic definitions and fundamental results from VC theory and then demonstrate how to utilize these concepts and results to analyze our method.

Let $(\mathcal{D}, \mathcal{R})$ be a range space, where $\mathcal{D}$ denotes a domain and $\mathcal{R}$ is a range set on $\mathcal{D}$. For any set $B \subseteq \mathcal{D}$, $P_{\mathcal{R}}(B) = \{B \cap A : A \in \mathcal{R}\}$ is the projection of $\mathcal{R}$ on B. If $P_{\mathcal{R}}(B) = 2^B$, where 2^B is the powerset of B, we say that the set B is shattered by $\mathcal{R}$. The following definitions and theorem are derived from the work of Riondato and Kornaropoulos [43].

Definition 4.1. The Vapnik-Chervonenkis dimension of $\mathcal{R}$, denoted as $VC(\mathcal{R})$, is the maximum cardinality of a subset of $\mathcal{D}$ that can be shattered by $\mathcal{R}$.

We give an example to explain the concept of VC dimension. For example, if a range set $\mathcal{R}$ is collections of intervals on a line, where each interval classifies the points inside the interval as 1 and those outside the interval as 0, $\mathcal{R}$ can shatter two points in a line but not three, because no interval can classify the three points on a line as "1 0 1." Thus, $VC(\mathcal{R}) = 2$.

Let $S = \{x_1, \ldots, x_n\}$ be a set of independent and identically distributed random variables sampled according to a distribution ϕ over domain $\mathcal{D}$. For a set $A \subseteq \mathcal{D}$, let $\phi(A)$ be the probability that an element sampled from ϕ belongs to A, and let the empirical estimation of $\phi(A)$ on S be

$$\phi_S(A) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_A(x_i),$$

where $\mathbf{1}_A$ is the indicator function with the value of $\mathbf{1}_A(x)$ that equals 1 if $x \in A$, and 0 otherwise.

The question of interest is that how well we can estimate $\phi(A)$ using its unbiased estimator, the empirical estimation $\phi_S(A)$. We first give the goodness of approximation in the following definition.

Definition 4.2. Let $\mathcal{R}$ be a range set on $\mathcal{D}$ and ϕ be a probability distribution on $\mathcal{D}$. For $\varepsilon \in (0, 1)$, an ε-approximation to $(\mathcal{R}, \phi)$ is a set S in $\mathcal{D}$ such that

$$sup_{A\in\mathcal{R}} |\phi(A) - \phi_S(A)| \leq \varepsilon.$$

One important result of VC theory is that if we can bound the VC dimension of $\mathcal{R}$, it is possible to build an ε-approximation by randomly sampling points from the domain according to the distribution ϕ. This is summarized in the following theorem.

THEOREM 4.3. *Let $\mathcal{R}$ be a range set on a domain $\mathcal{D}$, with $VC(\mathcal{R}) \leq d$, and let ϕ be a distribution on $\mathcal{D}$. Given $\varepsilon, \delta \in (0, 1)$, let S be a set of $|S|$ points sampled from $\mathcal{D}$ according to ϕ, with*

$$|S| = \frac{c}{\varepsilon^2}\left(d + \ln \frac{1}{\delta}\right),$$

where c is a universal positive constant. Then S is a ε-approximation to $(\mathcal{R}, \phi)$ with probability of at least $1 - \delta$.

According to the preceding theory, we set the domain in our problem to be Π—the set of all paths with length T in the graph G. Accordingly, we define the range set $\mathcal{R}_G$ on Π to be

$$\mathcal{R}_G = \{P(v_i, v_j) : v_i, v_j \in V\}.$$

This is a valid range set, as it is the collection of subsets P_{v_i,v_j} of domain Π. We first show an upper bound of the VC dimension of $\mathcal{R}_G$ in Lemma 4.4. The proof is inspired by Riondato and Kornaropoulos [43].

LEMMA 4.4. $VC(\mathcal{R}_G) \leq \log_2 \binom{T+1}{2} + 1$

PROOF. We prove the lemma by contradiction. Assume that $VC(\mathcal{R}_G) = l$ and $l > \log_2 \binom{T+1}{2} + 1$. By the definition of VC dimension, there is a set $Q \subseteq \Pi$ of size l that can be shattered by $\mathcal{R}_G$. In other words, we have the following statement:

$$\forall S_i \subseteq Q\ , \exists P_i \in \mathcal{R}_G,\ \text{s.t. } P_i \cap Q = S_i,$$

where P_i is the i-th range. Since each subset $S_i \subseteq Q$ is different from the other subsets, the corresponding range P_i making $P_i \cap Q = S_i$ is also different from the other ranges. Moreover, the set Q is shattered by $\mathcal{R}_G$ if and only if $\{P_i \cap Q : P_i \in \mathcal{R}\} = 2^Q$. Thus, $\forall p \in Q$, and there are 2^{l-1}

nonempty distinct subsets $S_1, \ldots, S_{2^{l-1}}$ of Q containing the path p. Therefore, there are also 2^{l-1} distinct ranges in $\mathcal{R}_G$ that contain the path p—that is,

$$|\{P_i | p \in P_i \text{ and } P_i \in \mathcal{R}_G\}| = 2^{l-1}.$$

In addition, according to the definition of range set, $\mathcal{R}_G = \{P(v_i, v_j) : v_i, v_j \in V\}$, a path belongs to the ranges corresponding to any pair of vertices in path p—that is, to the pairwise combinations of the vertices in p. According to the definition, a T-path contains $T + 1$ vertices, and some paths paths may contain a same vertex more than once. Thus, the number of ranges in $\mathcal{R}_G$ that p belongs to is no more than the combinatorial number $\binom{T+1}{2}$—that is,

$$|\{P_i | p \in P_i \text{ and } P_i \in \mathcal{R}_G\}| \leq \binom{T+1}{2}.$$

However, from our preliminary assumption, we have $l > \log_2 \binom{T+1}{2} + 1$, which is equal to $\binom{T+1}{2} < 2^{l-1}$. Thus,

$$|\{P_i | p \in P_i \text{ and } P_i \in \mathcal{R}_G\}| \leq \binom{T+1}{2} < 2^{l-1}.$$

Hence, we reach a contradiction: it is impossible to have 2^{l-1} distinct ranges $P_i \in \mathcal{R}_G$ containing p. Since there is a one-to-one correspondence between S_i and P_i, we get that it is also impossible to have 2^{l-1} distinct subsets $S_i \subseteq Q$ containing p. Therefore, we prove that Q cannot be shattered by $\mathcal{R}_G$ and $VC(\mathcal{R}_G) \leq \log_2 \binom{T+1}{2} + 1$.

We now provide a theoretical guarantee for the number of sampled paths. How many random paths do we need to achieve an error-bound ε with probability $1 - \delta$? We define a probability distribution on the domain Π, where Π denote all T-paths in G. $\forall p \in \Pi$, we define

$$\phi(p) = \text{prob}(p) = \frac{w(p)}{\sum_{p \in \Pi} w(p)}.$$

We can see that the definition of $S(v_i, v_j)$ in Equation (1) is equivalent to $\phi(P(v_i, v_j))$. This observation enables us to use a sampling-based method (empirical average) to estimate the original path similarity (true probability measure).

Plugging Lemma (4.4) into Theorem (4.3), we obtain

$$R = \frac{c}{\varepsilon^2}(\log_2 \binom{T+1}{2} + 1 + \ln \frac{1}{\delta}).$$

In other words, with at least R random paths, we can estimate the path similarity between any two vertices with the desired error bound and confidence level. The preceding equation also implies that R only depends on the path length T, given an error-bound ε and a confidence level $1 - \delta$.

5 PANTHER$_M$

Top-k metapath similarity search can be explained as follows: given a heterogeneous information network $G = (V, E, W; \phi, \mathcal{A}, \mathcal{R})$, a metapath $\mathcal{M} = (\mathcal{A}_1 \xrightarrow{R_1} \mathcal{A}_2 \xrightarrow{R_2} \cdots \xrightarrow{R_T} \mathcal{A}_{T+1})$, a positive integer k, any vertex $v \in V$ with type $\phi(v) = \mathcal{A}_1$, how to retrieve the top-k related vertices of v based on metapath similarity, (i.e., the type of the retrieved vertices is $\mathcal{A}_{T+1}$) and the path instances from v to the retrieved vertices are instantiated by $\mathcal{M}$.

ALGORITHM 3: Panther$_m$

```
Input: A network $G = (V, E, W; \phi, \mathcal{A}, \mathcal{R})$, a vertex $v$, parameters $\varepsilon, c, \delta, k$, meta-path $\mathcal{M} = (\mathcal{A}_1 \xrightarrow{R_1} \mathcal{A}_2 \xrightarrow{R_2} \cdots \xrightarrow{R_T} \mathcal{A}_{T+1})$.
Output: top-$k$ related vertices with regard to $v$.
Calculate transition probabilities between every pair of vertices according to Equation (9);
Initialize $r = 0$;
repeat
    Sample a vertex $v_c = v_i$ uniformly at random from the nodes with source type $\mathcal{A}_1$;
    for $t = 1; t \leq T; t++$ do
        Randomly sample a neighbor $v_t$ according to transition probabilities from $v_c$ to its neighbors with type $\mathcal{A}_{t+1}$,
          satisfying $\psi(v_c, v_t) = \mathcal{R}_t$;
        Set current vertex $v_c = v_t$;
    end
    if $t = T$ then
        $S_{\mathcal{M}}(v_i, v_c) += \frac{1}{R}$;
    end
until $r = R$;
Retrieve top-$k$ similar vertices according to $S_{\mathcal{M}}(v, v_j)$;
```

Similarly, in large-scale networks, we need to obtain an approximate set of the related vertices that can guarantee an error bound with a confidence interval. We will introduce how the sampling-based method can efficiently estimate metapath similarity and guarantee the error bound.

Random sampling for metapath similarity. To calculate Equation (2), we need to enumerate all paths instantiated by metapath $\mathcal{M}$ from v_i to v_j, which is also inefficient. Thus, we estimate metapath similarity approximately by sampling metapath-based paths:

$$S_{\mathcal{M}}(v_i, v_j) = \frac{\sum_{p \in P_{\mathcal{M}}(v_i, v_j)} w(p)}{\sum_{p \in P_{\mathcal{M}}} w(p)}, \tag{8}$$

where $P_{\mathcal{M}}$ is the sampled paths instantiated by $\mathcal{M}$.

To generate a path instantiated by $\mathcal{M}$, we randomly select a vertex v_i in G with the type $\mathcal{A}_1$ as the starting point and then conduct random walks of T steps from v_i using t_{ij} as the transition probability from node v_i to v_j:

$$t_{i,i+1}^{(\mathcal{R}_i)} = \frac{w_{i,i+1}}{\sum_{v_k \in \mathcal{N}(v_i) \cap \psi(v_i, v_k) = \mathcal{R}_i} w_{ik}}, \tag{9}$$

where we restrict the neighbors of v_i into those associated with edge type $\mathcal{R}_i$ in metapath $\mathcal{M}$. Accordingly, the weight of a path $w(p_{\mathcal{M}}) = \prod_{i=1}^{T} t_{i,i+1}^{\mathcal{R}_i}$. Because $w(p_{\mathcal{M}})$ is absorbed in the random walk process, Equation (8) can be rewritten as follows:

$$S_{\mathcal{M}}(v_i, v_j) = \frac{|P_{\mathcal{M}}(v_i, v_j)|}{R}. \tag{10}$$

Panther$_m$ only considers the paths between two vertices initialized by a given metapath $\mathcal{M}$. Thus, we only count the source and target vertices in a path rather than each distinct vertex in a path. Therefore, we do not need to record the paths and the vertex-to-path index. Instead, when generating random paths, we directly update the similarities between source and target vertices in a same random path. Thus, the space complexity is reduced to $O(|V|\bar{d})$. The updated algorithm is presented in Algorithm 3.

Theoretical analysis. Theoretical proof is similar to that of Panther. For Panther$_m$, the domain $\mathcal{D}$ is changed to the paths instantiated by a metapath $\mathcal{M}$ (i.e., $\Pi_{\mathcal{M}}$). Range set $\mathcal{R}$ is changed to $\mathcal{R}_G = \{P_{\mathcal{M}}(v_i, v_j) : v_i, v_j \in V\}$.

ALGORITHM 4: Panther$_v$

```
Input: A network $G = (V, E, W)$, path length $T$, parameters $\varepsilon$, $c$, $\delta$, $D$, $k$ and a vertex $v$.
Output: top-$k$ similar vertices with regard to $v$.
Calculate sample size $R = \frac{c}{\varepsilon^2}(\log_2 \binom{T}{2} + 1 + \ln \frac{1}{\delta})$;
GenerateRandomPath($G$, $R$);
foreach $v_i \in V$ do
    foreach $p_n \in P(v_i)$ do
        foreach Unique $v_j \in p_n$ do
            $S(v_i, v_j) += \frac{1}{R}$;
        end
    end
    Construct a vector $\theta(v_i)$ by taking the largest $D$ values from $\{S(v_i, v_j) : v_j \in p_n$ and $p_n \in P(v_i)\}$;
end
Build a kd-tree index based on the Euclidean distance between any vectors $\theta(v_i)$ and $\theta(v_j)$;
Query the top-$k$ similar vertices from the index for $v$.
```

Accordingly, we need to prove the lemma $VC(\mathcal{R}_G) \leq 1$. We also prove the lemma by contradiction. Assume that $VC(\mathcal{R}_G) = l$ and $l > 1$. Similar to the proof of Panther, we also know that there are 2^{l-1} distinct ranges containing a path p. In addition, in Panther$_m$, each path only belongs to the ranges corresponding to the source and target vertices in the path—that is, the number of ranges in $\mathcal{R}_G$ that p belongs to is equal to 1. However, from the preliminary $l > 1$, we know that $2^{l-1} \neq 1$. Hence, we reach a contradiction and prove the given lemma.

The definition of $S_{\mathcal{M}}(v_i, v_j)$ in Equation (2) is exactly a probability distribution on the domain $\Pi_{\mathcal{M}}$. Thus, we can plug $VC(\mathcal{R}_G) \leq 1$ into Theorem 4.3 and obtain $R = \frac{c}{\varepsilon^2}(1 + \ln \frac{1}{\delta})$.

6 PANTHER$_V$

According to the definition of vector similarity, for each vertex we construct a vector by top-D ranked similarities between it and other vertices according to Equation (3). The similarity metric can be chosen as any kind of neighborhood similarity. We determine to use our proposed path similarity, which can be quickly estimated by Panther.

Specifically, for vertex v_i in the network, we first calculate the similarity between v_i and all other vertices using Panther. Then we construct a feature vector for v_i by taking the largest D similarity scores as feature values.

Finally, the similarity between v_i and v_j is calculated as the reciprocal Euclidean distance between their feature vectors according to Equation (3).

Index of feature vectors. Again, we use the indexing techniques to improve the algorithm's efficiency. We build a memory based kd-tree [55] index for feature vectors of all vertices. Then given a vertex, we can retrieve top-k vertices in the kd-tree with the least Euclidean distance to the query vertex efficiently. At a high level, a kd-tree is a generalization of a binary search tree that stores points in D-dimensional space. In level h of a kd-tree, given a node v, the $h\%D$-th element in the vector of each node in its left subtree is less than the $h\%D$-th element in the vector of v, whereas the $h\%D$-th element of every node in the right subtree is no less than the $h\%D$-th element of v. Figure 4 shows the data structure of the index built in Panther$_v$. Based on the index, we can query whether a given point is stored in the index very fast. Specifically, given a vertex v, if the root node is v, return the root node. If the first element of v is strictly less than the first element of the root node, look for v in the left subtree, then compare it to the second element of v. Otherwise, check the right subtree. It is worth noting that we can easily replace the kd-tree with any other

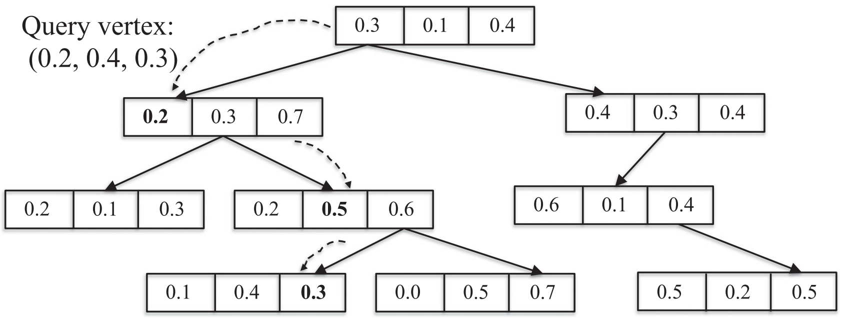


Fig. 4. Data structure of the index built in Panther$_v$.

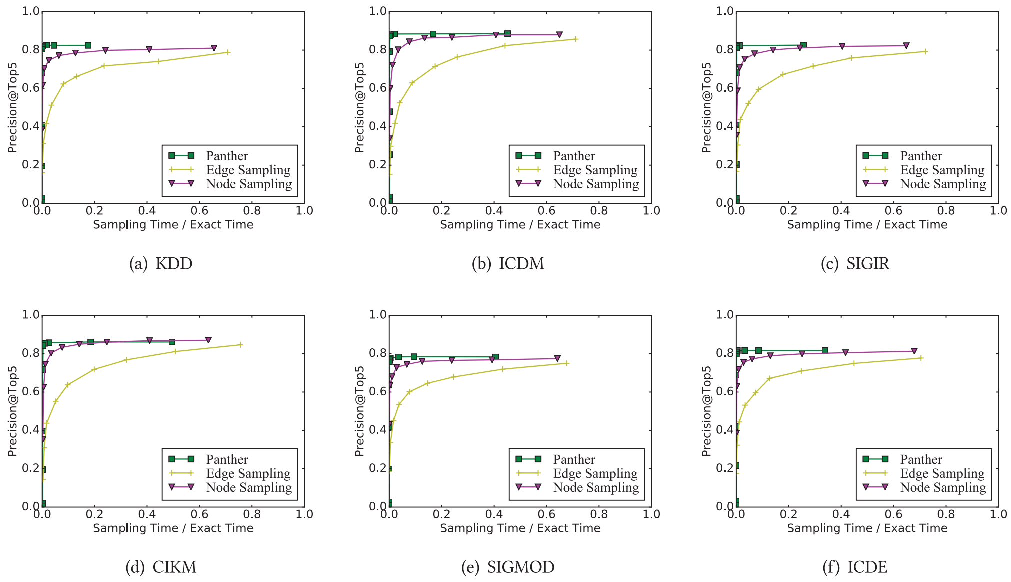


Fig. 5. Trade-off between accuracy (i.e., Precision@Top-5 at the y-axis) and efficiency (i.e., Sampling-time/Exact-time at the x-axis) performance by varying error-bound ε of Panther, edge sampling probability p of the edge sampling method, and node sampling probability q of the node sampling method on the coauthor networks.

index methods, such as an r-tree. The algorithms for calculating feature vectors of all vertices and the similarity between vertices are shown in Algorithm 4, where lines 1 through 8 represent the preprocessing steps and line 9 refers to the top-k similarity searching for a vertex.

7 COMPARISON WITH EXISTING METHODS

In general, most of the existing methods result in high complexities. For example, the time complexities of SimRank [24], RWR [39], and RoleSim [27] are all proportional to $|V|^2$. TopSim, the top-k version of SimRank, is more efficient but is still exponentially proportional to the number of random walk steps. Like our method Panther$_v$, ReFeX [19, 20] also constructs a feature vector for each vertex and then calculates the Euclidean distance between vectors as their similarities. However, the complexity of constructing features is determined by the iteration times and is also

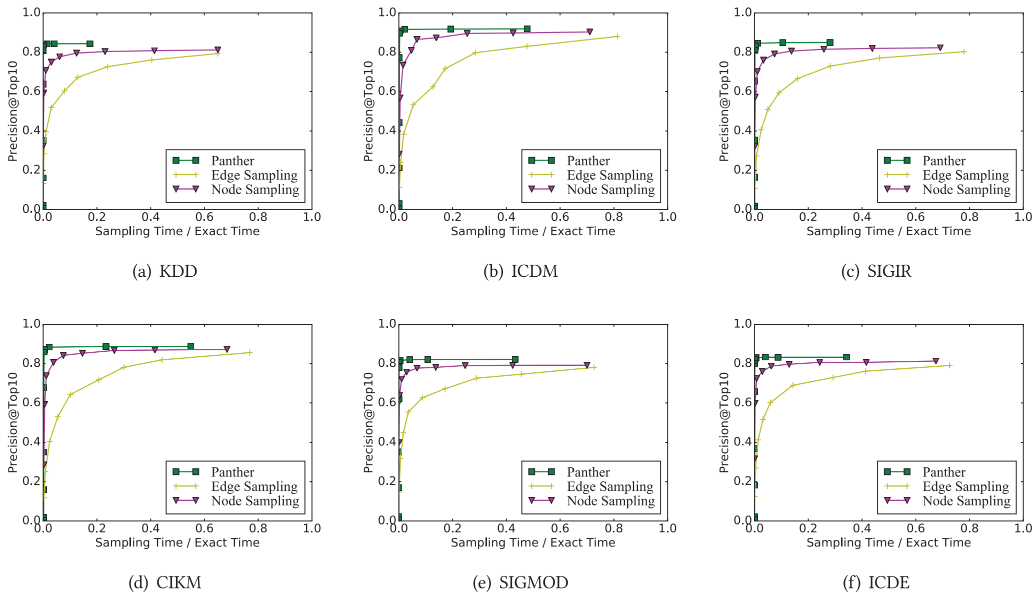


Fig. 6. Trade-off between accuracy (i.e., Precision@Top-10 at the y-axis) and efficiency (i.e., Sampling-time/Exact-time at the x-axis) performance by varying error-bound ε of Panther, edge sampling probability p of the edge sampling method, and node sampling probability q of the node sampling method on the coauthor networks.

Table 1. Time and Space Complexity for Calculating Top-k Similar Vertices for All Vertices in a Network

Method	**Equation**	**Time Complexity**	**Space Complexity**
SimRank [24]	$S(v_i, v_j) = \frac{\beta}{\lvert\mathcal{N}_i\rvert\lvert\mathcal{N}_j\rvert} \sum_{v_k \in \mathcal{N}_i} \sum_{v_h \in \mathcal{N}_j} S(v_k, v_h)$	$O(I\lvert V\rvert^2\bar{d}^2)$	$O(\lvert V\rvert^2)$
TopSim [33]	$S(v_i, v_j) = \sum_{l=1}^{T} \sum_{p \in SP_l(v_i, v_j)} \beta^l w(p)$	$O(\lvert V\rvert T\bar{d}^T)$	$O(\lvert V\rvert + \lvert E\rvert)$
RWR [39]	$\vec{u}_i = (1-\beta)A\vec{u}_i + \beta\vec{I}_i$	$O(I\lvert V\rvert^2\bar{d})$	$O(\lvert V\rvert^2)$
RoleSim [27]	$S(v_i, v_j) = (1-\beta)\max_{M_{i,j}} \frac{\sum_{(v_k, v_h) \in M_{i,j}} S(v_k, v_h)}{\mathcal{N}_i + \mathcal{N}_j - \lvert M_{i,j}\rvert} + \beta$	$O(I\lvert V\rvert^2\bar{d}^2)$	$O(\lvert V\rvert^2)$
ReFeX [20]	$S_v(v_i, v_j) = \frac{1}{\lVert\theta(v_i) - \theta(v_j)\rVert}$	$O(\lvert V\rvert + I(f\lvert E\rvert + \lvert V\rvert f^2))$	$O(\lvert V\rvert + \lvert E\rvert f)$
Panther	$S(v_i, v_j) = \frac{\sum_{p \in P(v_i, v_j)} w(p)}{\sum_{p \in P} w(p)}$	$O(RTc + \lvert V\rvert\bar{d}T)$	$O(RT + \lvert V\rvert\bar{d})$
Panther$_m$	$S_M(v_i, v_j) = \frac{\sum_{p \in P_M(v_i, v_j)} w(p)}{\sum_{p \in P_M} w(p)}$	$O(RTc + \lvert V\rvert\bar{d}T)$	$O(\lvert V\rvert\bar{d})$
Panther$_v$	$S_v(v_i, v_j) = \frac{1}{\lVert\theta(v_i) - \theta(v_j)\rVert}$	$O(RTc + \lvert V\rvert\bar{d}T + \lvert V\rvert c)$	$O(RT + \lvert V\rvert\bar{d} + \lvert V\rvert D)$

I, number of iterations; $\bar{d}$, average degree; f, feature number; D, vector dimension; T, maximal path length; l, path length; $SP_l(u, v)$, set of random walk paths on $G \times G$ of length l ending at the target vertex uv; β, decay factor; $\vec{u}_q$, steady state probability vector with v_i as the start vertex; $\vec{I}_i$, restart vector; A, adjacency matrix of the network; $M_{i,j}$, matching between $\mathcal{N}_i$ and $\mathcal{N}_j$.

exponentially proportional to the iteration times. Table 1 shows the time and space complexity of different methods and our methods.

For our method Panther, its time complexity is determined by two main steps. The first step is a random path sampling process. The time complexity of generating random paths is $O(RT \log \bar{d})$, where $\log \bar{d}$ is for randomly sampling a neighbor and can be simplified as a small constant c. Hence, the time complexity is $O(RTc)$. The second step is the top-k similarity search process. The time complexity of calculating top-k similar vertices for all vertices is $O(|V|\bar{R}T + |V|\bar{M})$. The first part $O(|V|\bar{R}T)$ is the time complexity of calculating Equation (7) for all pairs of vertices, where $\bar{R}$ is the average number of paths that contain a vertex and is proportional to the average degree $\bar{d}$. The second part $O(|V|\bar{M})$ is the time complexity of searching top-k similar vertices based on a heap structure, where $\bar{M}$ represents the average number of co-occurred vertices with a vertex and is proportional to $\bar{d}$. Hence, the time complexity is $O(|V|\bar{d}T)$. The space complexity for storing paths and vertex-to-path index is $O(RT)$ and $O(|V|\bar{d})$, respectively. The time and space complexity of Panther$_m$ is the same as that of Panther.

Panther$_v$ requires additional computation to build the kd-tree. The time complexity of building a kd-tree is $O(|V|\log|V|)$ and querying top-k similar vertices for any vertex is $O(|V|\log|V|)$, where $\log|V|$ is small and can be viewed as a small constant c. Additional space (with a complexity of $O(|V|D)$) is required to store $|V|$ D-dimension vectors.

8 EXPERIMENTS

8.1 Experimental Setup

In this section, we conduct various experiments to evaluate the proposed methods for top-k similarity search.

Datasets. We evaluate the proposed method on four different networks: Tencent, Twitter, Mobile, and Medicine:

- *Coauthor* [51]: The dataset is from AMiner.org[5] and contains 2,092,356 papers. We extracted a weighted coauthor graph from each of the following conferences from 2005 to 2013: KDD, ICDM, SIGIR, CIKM, SIGMOD, and ICDE.[6] The weight associated with each edge is the number of works collaborated by the two connected authors. We use the dataset to evaluate the sampling performance (i.e., the trade-off between the accuracy and efficiency performance) of Panther compared to alternative sampling algorithms.
- *Tencent* [57]: The dataset is from Tencent Weibo,[1] a popular Twitter-like microblogging service in China, and consists of more than 355,591,065 users and 5,958,853,072 “following” relationships. The weight associated with each edge is set as 1.0 uniformly. This is the largest network in our experiments. We use it to evaluate the efficiency performance of Panther and Panther$_v$ compared to alternative similarity metrics.
- *Twitter* [18, 22]: We collect the dataset by first selecting the most popular user on Twitter (i.e., Lady Gaga) and randomly selecting 10,000 of her followers, then collecting all followers of these users. In the end, we have 113,044 users and 468,238 following relationships in total. The weight associated with each edge is also set as 1.0 uniformly. We use this dataset to evaluate the accuracy of Panther and Panther$_v$ compared to alternative similarity metrics.
- *Mobile* [11]: We build the mobile network using call records from a mobile communication company within 2 weeks by treating each user as a vertex and the communication between users as an edge. The resultant network consists of 194,526 vertices and 206,934 edges, with the weight associated with each edge as the number of calls. We also use this dataset to evaluate the accuracy of Panther and Panther$_v$ compared to alternative similarity metrics.
- *Medicine* [9]: The dataset is a heterogeneous medical information network that consists of 295,897 nodes and 727,931 edges. There are 9 types of vertices in the networks, including 258,030 drugs and 22,056 proteins. Then 12 types of edges are built between these vertex

[5]https://cn.aminer.org/citation.
[6]The numbers of vertices/edges of different conferences are as follows: KDD, 2,867/7,637; ICDM, 2,607/4,774; SIGIR, 2,851/6,354; CIKM, 3,548/7,076; SIGMOD, 2,616/8,304; and ICDE, 2,559/6,668.

types. Please refer to Chen et al. [9] for details on the dataset. We use the dataset to evaluate the efficiency and accuracy of Panther$_m$.

Evaluation aspects. To quantitatively evaluate the proposed methods, we consider the following performance measurements:

— *Sampling performance*: We apply our method Panther to the coauthor networks to evaluate the sampling performance—that is, the trade-off between accuracy and efficiency performance.
— *Efficiency performance*: We apply our methods to the Tencent network to evaluate the computational time.
— *Accuracy performance*: We use the number of common neighbors as ground truth to evaluate Panther on Twitter and Mobile networks. We apply Panther$_v$ to find top-k structural hole spanners in Twitter and Mobile networks. We evaluate Panther$_m$ using the task of link prediction on a medical information network.
— *Parameter sensitivity analysis*: We analyze the sensitivity of different parameters in our methods: path length T, vector dimension D, and error-bound ε.

All codes are implemented in C++ and compiled using GCC 4.8.2 with −O3 flag. The experiments were conducted on a Ubuntu server with four Intel Xeon CPU E5-4650s (2.70GHz) and 1T RAM.

Comparison methods. We compare to the following methods:

— *Edge sampling* [2]: This samples a subnetwork by selecting each edge in the original network by probability p and then enumerates all the T-paths in the sampled subnetwork. Based on the sampled paths, we calculate path similarity according to Equation (7).
— *Node sampling* [56]: This samples a path by iteratively selecting a neighbor of each vertex by probability q until the length of a path is T. Based on the sampled paths, we calculate path similarity according to Equation (7).
— *RWR* [39]: RWR starts from v_i, iteratively walking to its neighbors with the probability proportional to their edge weights. At each step, it also has some probability to walk back to v_i (set as 0.1). The similarity between v_i and v_j is defined as the steady-state probability that v_i will finally reach v_j. We calculate RWR scores between all pairs and then search the top-k similar vertices for each vertex.
— *TopSim* [33]: TopSim extends SimRank [24] on one graph G to find top-k authoritative vertices on the product graph $G \times G$ efficiently.
— *RoleSim* [27]: RoleSim refines SimRank [24] by changing the average similarity of all neighbor pairs to all matched neighbor pairs. We estimate RoleSim scores between all pairs and select the top-k similar vertices for each vertex.
— *ReFeX* [20]: ReFeX defines local, egonet, and recursive features to capture structural characteristics. The local feature is the vertex degree. Egonet features include the number of within-egonet edges and the number of out-egonet edges. For weighted networks, they contain weighted versions of each feature. Recursive features are defined as the mean and sum value of each local or egonet feature among all neighbors of a vertex. In our experiments, we only extract recursive features once and construct a vector for each vertex by a total of 18 features. For fair comparison, to search top-k similar vertices, we also build the same kd-tree as that used in our method.
— *SLAP* [9]: SLAP is proposed to predict links in a medical information network. Given a pair of source and target vertices in a network, it employs a heap-based Dijkstra algorithm to find all paths between the two vertices and sums up the scores of all paths together.

The score for each path is a combination of the path weight (i.e., the multiplication of the transition probabilities in the path) and the significance score of the metapath to which the path belongs (i.e., the expected mean and standard deviation of the weights of the paths that belong to the metapath).

The codes of TopSim, RoleSim, ReFeX, and SLAP are provided by their authors. We used the fast versions of TopSim and RoleSim mentioned in their work.

Implementation notes. In our experiments, we empirically set the parameters as follows: $c = 0.5$, $\delta = 0.1$, $T = 4$, $D = 50$, and $\varepsilon = \sqrt{1/|E|}$. The optimal values of T, D, and ε are discussed in Section 8.5. We build the kd-tree using the toolkit ANN.[7]

8.2 Sampling Performance

We use Precision@Top-k, where we vary k as 5 and 10, as the metric to measure how far the approximate estimation by Panther is from the exact estimation, Equation (1). Specifically, for each vertex v_i, given an approximate estimation of the top-k similar vertices, $\widetilde{\mathrm{TS}}_i$, and the exact estimation of the top-k similar vertices, TS_i, Precision@Top-k is defined as follows:

$$\text{Precision@Top-}k = \frac{\sum_{i=1}^{|V|} |\widetilde{\mathrm{TS}}_i \cap \mathrm{TS}_i|}{k|V|}. \tag{11}$$

To quantitatively evaluate the sampling performance of Panther, we compare Panther to two alternative sampling methods: edge sampling and node sampling. The comparison methods use the same method as Panther to calculate path similarity except different methods sample paths in different ways. We show how Precision@Top-k and speed vary by varying the parameters in different methods. The experimental setting is as follows:

— We tune the parameters. For Panther, we try the error-bound ε as all possible values in the range $\{0.0006, 0.001, 0.003, 0.006, 0.01, 0.03, 0.06, 0.1, 0.3, 0.6\}$; for the edge sampling method, we try the edge sampling probability p as all possible values in the range $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$; and for the node sampling method, we try the node sampling probability q with the same values of p.
— For each configuration of the parameters ε, p, and q, we calculate Precision@Top-k over 10 independent sampling results.
— For each configuration of the parameters ε, p, and q, we record the average CPU time of 10 independent sampling processes and divide them by the CPU time of the exact estimation (i.e., Sampling-time/Exact-time).

Figures 5 and 6 plot all (Precision@Top-k, Sampling-time/Exact-time) pairs when varying the parameters ε, p, and q on six different coauthor networks. From the results, we can see that for all three sampling methods, Precision@Top-k increases quickly at the beginning and then becomes almost stable. Precision@Top-k of Panther increases dramatically at the beginning and takes much shorter time to obtain a higher accuracy than the other two methods and thus performs best. This is because Panther actually absorbs the path weight (Equation (6)) in the random sampling process, which exactly reserves the effect of path weight defined in path similarity (Equation (1)), whereas edge sampling and node sampling methods do not consider path weight in their sampling processes. The experimental results demonstrate the superiority of the proposed sampling method.

[7] http://www.cs.umd.edu/~mount/ANN/.

8.3 Efficiency and Scalability Performance

In this section, we first fix $k = 5$, and evaluate the efficiency and scalability performance of different comparison methods using the Tencent dataset. We randomly extract different (large and small) versions of the Tencent networks. For TopSim and RoleSim, we only show the computational time for similarity search. For ReFex, edge sampling, node sampling, Panther, and Panther$_v$, we also show the computational time used for preprocessing. Since Tencent is a homogeneous network, we show the time efficiency of Panther$_m$ on a heterogeneous medical information network in Section 8.4. Sampling probability p for edge sampling is set as 0.6, and q for node sampling is set as 0.3 according to the experimental results on coauthor, Twitter, and Mobile datasets.

Table 2 lists statistics of the different Tencent subnetworks and the efficiency performance of the comparison methods. Clearly, our methods (both Panther and Panther$_v$) are much faster than the comparison methods. For example, on the Tencent6 subnetwork, which consists of 443,070 vertices and 5,000,000 edges, Panther$_v$ achieves a 390× speedup compared to the fastest (ReFeX) of all comparative methods.

We can also see that RWR, TopSim, and RoleSim cannot complete top-k similarity search for all vertices within a reasonable time when the number of edges increases to 500,000. ReFeX can deal with larger networks but also fails when the edge number increases to 10,000,000. According to the results shown in Figures 5 through 7, when setting sampling probability p for edge sampling as 0.6 and q for node sampling as 0.3, the accuracy performance can approach that of ; however, according to the results in Table 2, Panther is much more efficient than the two baseline sampling methods. Our methods can scale up to handle very large networks with more than 10,000,000 edges. On average, Panther only needs 0.0001 second to perform top-k similarity search for each vertex in a large network.

8.4 Accuracy Performance

8.4.1 Performance of Panther.. We evaluate how Panther can approximate the similarity based on common neighbors. The evaluation procedure is described as follows:

(1) For each vertex u in the seed set S, generate top k vertices $\text{Top}_{A,k}(u)$ that are the most similar to u by algorithm A.
(2) For each vertex $v \in \text{Top}_{A,k}(u)$, calculate $g(u,v)$, where g is a coarse similarity measure defined as the ground truth. Define $f_{A,k} = \sum_u \sum_v g(u,v)$.
(3) Let $f_{R,k}$ denotes the result of a random algorithm.
(4) Finally, we define the score for algorithm A as $\text{score}(A,k) = \frac{f_{A,k} - f_{R,k}}{|S| \times k}$, which represents the improvement of algorithm A over a random method.

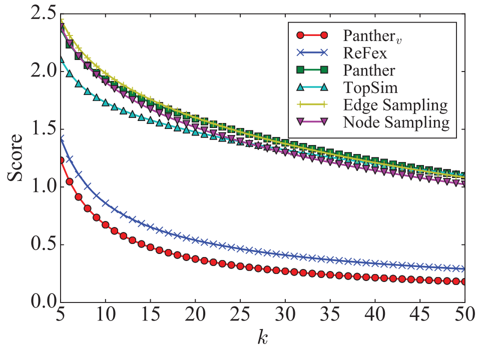
Specifically, we define $g(u,v)$ as the number of common neighbors between u and v on each dataset.

Figure 7 shows the performance of Panther evaluated on the ground truth of common neighbors on Twitter and Mobile networks. Some baselines, such as RWR and RoleSim, are ignored on the datasets because they cannot complete top-k similarity search for all vertices within a reasonable time. It can be seen that Panther performs better than any other methods on both datasets. Panther and ReFeX perform worst, as they are not devised to address the similarity between near vertices. Our method Panther performs as good as TopSim, the top-k version of SimRank, and the two baseline sampling methods, edge sampling ($q = 0.6$) and node sampling ($p = 0.3$), because they are all based on the principle that two vertices are considered structurally equivalent if they have many common neighbors in a network. However, according to our previous analysis, TopSim, edge sampling, and node sampling perform much slower than Panther.

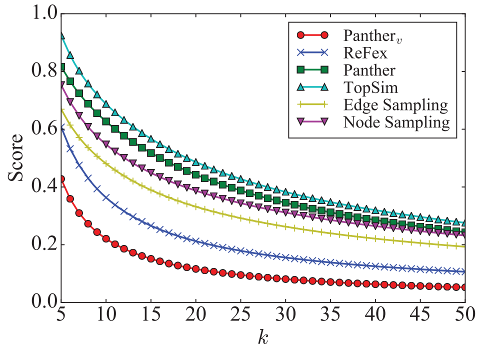
Table 2. Efficiency Performance (CPU Time) of Comparison Methods on Different Sizes of the Tencent Subnetworks

Subnetwork	$\lvert V \rvert$	$\lvert E \rvert$	RWR	TopSim	RoleSim	ReFeX	Edge Sampling	Node Sampling	Panther	Panther$_v$
Tencent1	6,523	10,000	+7.79hr	+28.58m	+37.26s	3.85s+0.07s	14.16m+1.68hr	58.82s+1.12m	0.07s+0.26s	0.99s+0.21s
Tencent2	25,844	50,000	+>150hr	+11.20hr	+12.98m	26.09s+0.40s	—	14.22m+26.14m	0.28s+1.53s	2.45s+4.21s
Tencent3	48,837	100,000	—	+30.94hr	+1.06hr	2.02m+0.57s	—	1.50hr+1.67hr	0.58s+ 3.48s	5.30s+5.96s
Tencent4	169,209	500,000	—	+>120hr	+>72 hr	17.18m+2.51s	—	—	8.19s+16.08s	27.94s+24.17s
Tencent5	230,103	1,000,000	—	—	—	31.50m+3.29s	—	—	15.31s+30.63s	49.83s+22.86s
Tencent6	443,070	5,000,000	—	—	—	24.15hr+8.55s	—	—	50.91s+2.82m	4.01m+1.29m
Tencent7	702,049	10,000,000	—	—	—	>48hr	—	—	2.21m+6.24m	8.60m+6.58m
Tencent8	2,767,344	50,000,000	—	—	—	—	—	—	15.78m+1.36hr	1.60hr+2.17hr
Tencent9	5,355,507	100,000,000	—	—	—	—	—	—	44.09m +4.50hr	5.61hr +6.47hr
Tencent10	26,033,969	500,000,000	—	—	—	—	—	—	4.82hr +25.01hr	32.90hr +47.34hr
Tencent11	51,640,620	1,000,000,000	—	—	—	—	—	—	13.32hr +80.38hr	98.15hr +120.01hr

Note: The time before the plus sign (+) denotes the time used for processing and the time after the plus sign denotes that used for top-k similarity search. The dash (—) indicates that the corresponding algorithm cannot finish the computation within a reasonable time.

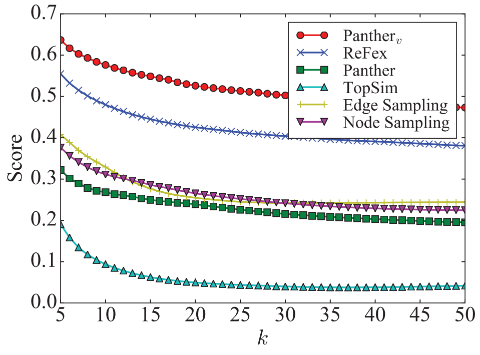


(a) Twitter

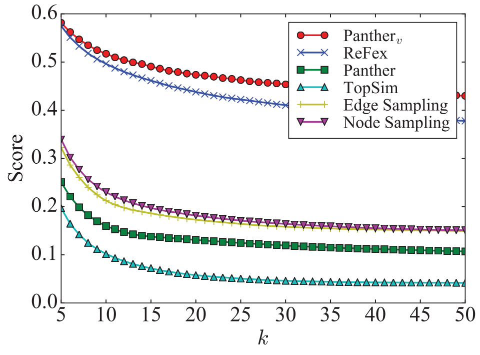


(b) Mobile

Fig. 7. Performance of approximating common neighbors on the Twitter and Mobile networks.



(a) Twitter



(b) Mobile

Fig. 8. Performance of mining structural hole spanners on the Twitter and Mobile networks.

8.4.2 Performance of Panther$_v$.. We consider an application of the top-k structural hole spanner finding to evaluate the performance of Panther$_v$. The theory of structural holes [8] suggests that in social networks, individuals would benefit from filling the “holes” between people or groups that are otherwise disconnected. The problem of finding top-k structural hole spanners was proposed in the work of Lou and Tang [36], which also shows that 1% of users who span structural holes control 25% of the information diffusion (retweeting) in Twitter.

Structural hole spanners are not necessarily connected, but they share the same structural patterns (e.g., local clustering coefficient and centrality). Thus, the idea here is to feed a few seed users to Panther$_v$ and use it to find other structural hole spanners. For evaluation, we use network constraint [8] to obtain the structural hole spanners in Twitter and Mobile, and use this as the ground truth. Then we apply different methods—Panther$_v$, ReFeX, Panther, TopSim, edge sampling ($p = 0.6$), and node sampling ($q = 0.3$)—to retrieve top-k similar users for each seed. If an algorithm can find another structural hole spanner in the top-k returned results, it makes a correct search. We define $g(u, v) = 1$ if both u and v are structural hole spanners, and $g(u, v) = 0$ otherwise.

Figure 8 shows the performance of comparison methods for finding structural hole spanners in different networks. Panther$_v$ achieves consistently better performance than the comparison methods by varying the value of k. TopSim, Panther, and the two baseline sampling methods seem inapplicable to this task. This is reasonable, as the underlying principle of them is to find vertices with more connections to the query vertex.

8.4.3 Performance of Panther$_m$.. We use the same task and dataset in the work of Chen et al. [9] to evaluate the efficiency and effectiveness performance of Panther$_m$. The task is to predict links between drugs and proteins. In a medical information network, due to the investigation difficulty and latency in clinical field experiments, few reliable links are constructed, and thus the existing links in the network are usually quite sparse. For example, the OMIM dataset used in the work of Natarajan and Dhillon [37] contains 3,209 diseases and 8,755 genes, but only 3,954 gene-disease associations. Therefore, automatically mining and discovering the potential associations between medical entities can effectively enrich the links between medical entities and meanwhile provide a useful hypothesis for new clinical experiments.

Predicting links in a medical information network has been extensively studied. Some research builds links between two entities two-hops away from each other based on the number of their common neighbors [17]. Other research work further extends the two-hop away associations to more distant associations. For example, Chen et al. [9] consider the paths with different lengths between two entities and restrict the maximal length of paths to 3. However, their method is inefficient, taking several days to finish on such a dataset.

Setting. We collect the ground truth from the public database PubChem BioAssay,[8] which contains bioactivity screens of chemical substances described in PubChem Substance. For example, there are active or inactive bioactivities between some drugs and proteins in the database. We first choose the pairs of drugs and proteins with activity records as candidate links, then treat the active interactive pairs as positive links and the inactive ones as negative links. Finally, we remove the links existing in the medical information network. The resulting ground truth contains a total of 36,254 positive links and 343,043 negative links.

For each drug in the ground truth, we use Panther$_m$ to estimate top-k similar vertices and treat the links between the drug and its top-k similar proteins as the predicted positive links and others as negative links. Specifically, we first enumerate all kinds of metapaths with length no longer than 4. The paths longer than 4 are considered too long to predict significant links. Then for each drug, we estimate top-k similar vertices following each selected metapath and aggregate the similarities of different metapaths together. Please refer to the work of Chen et al. [9] for details of the metapaths. We compare to the baseline SLAP [9] and our method Panther.

Results. Figure 9 shows the ROC curves and AUC of comparison methods for predicting links in a medical information network. In the result, k is set as 500 and error-bound ϵ is set as 1×10^{-4}. We see that SLAP performs better than Panther$_m$. The results are reasonable, because SLAP is an exact solution, which enumerate all possible paths between two vertices, whereas Panther$_m$ only samples a subset in the whole path space. However, Panther$_m$ takes only 1 hour, whereas SLAP takes about 58 hours. Panther$_m$ achieves a more than 50x speedup compared to SLAP. We also conduct Panther in the network. The results show that Panther performs much worse than Panther$_m$, as Panther ignores the type of paths, which may introduce additional noises.

We then reduce the value of error-bound ϵ to investigate whether the performance of Panther$_m$ can be improved. The results are shown in Figure 10(a). We see that with the decreasing of the error bound, the performance of Panther$_m$ approaches that of SLAP. When the error bound is

[8]http://www.ncbi.nlm.nih.gov/pcassay/.

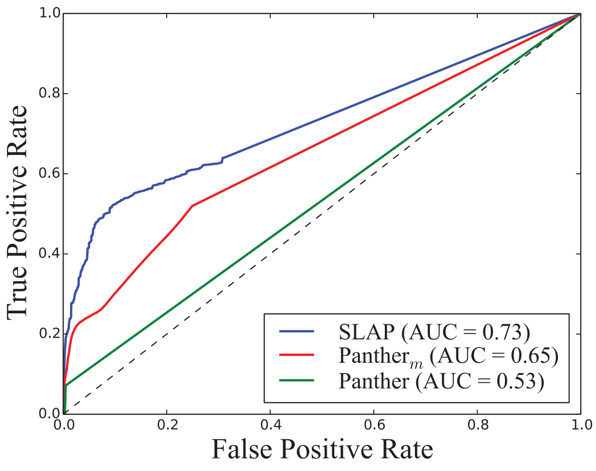
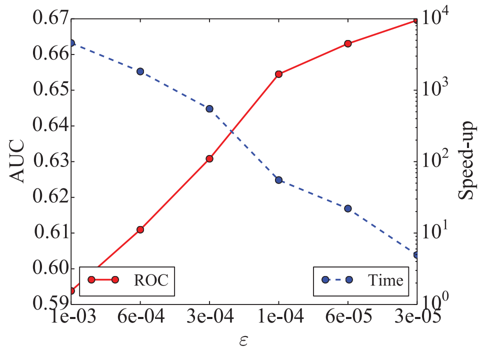
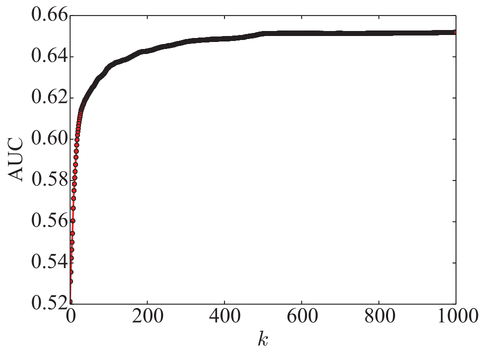


Fig. 9. ROC curve and AUC of Panther$_m$.



(a)



(b)

Fig. 10. AUC of Panther$_m$ by varying ε (a) and k (b).

reduced to 6×10^{-5}, AUC is improved to 0.66, with a time cost of 2.5 hours, which still achieves 20x speedup compared to SLAP.

Furthermore, we change the value of k to see whether the performance is sensitive to k. In this experiment, we fix ε as 1×10^{-4}. Figure 10(b) shows AUC when changing the value of k from 1 to 1,000. We can see that AUC increases and then becomes stable when k is 500. This indicates that some positive instances in the ground truth are not ranked quite high in the top similar list. Thus, more positive instances can be recalled when increasing k. We further study several vertices ranked before the positive instances in the ground truth. Figure 11 shows a case study of the paths between protein AURKB and drug 5312137, where AURKB is ranked 46th by our algorithm and is included in the ground truth. Figure 1(b) presents the paths between protein MET and drug 5312137, where MET is ranked 8th by our algorithm but is not in the ground truth. We plot the paths with length no longer than 3 from drug 5312137 to the predicted proteins. The number of paths from drug 5312137 to protein MET is 42, which is more than the 24 paths from drug 5312137 to protein AURKB. Moreover, from the figure, we can see that there are three metapaths, $Drug \xrightarrow{bind} Protein \xrightarrow{bind} Drug \xrightarrow{bind} Protein$, $Drug \xrightarrow{bind}$

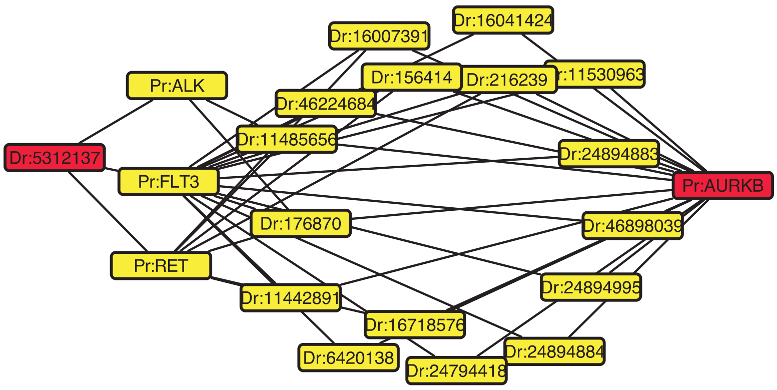


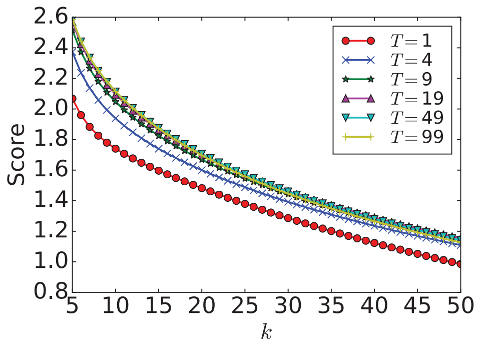
Fig. 11. Paths between drug 5312137 and protein AURKB. The vertices in red are source and target vertices. The prefix "Dr" in the vertex label denotes drug, "Pr" denotes protein, and "Pa" denotes pathway..

Table 3. AUC of Different Metapaths

Metapath	AUC
$Drug \xrightarrow{bind} Protein \xrightarrow{bind} Drug \xrightarrow{bind} Protein$	0.64
$Drug \xrightarrow{bind} Protein \xrightarrow{bind} GeneOntology \xrightarrow{bind} Protein$	0.57
$Drug \xrightarrow{bind} Protein \xrightarrow{bind} Protein \xrightarrow{bind} Protein$	0.56
$Drug \xrightarrow{bind} Protein \xrightarrow{bind} Pathway \xrightarrow{bind} Protein$	0.53
$Drug \xrightarrow{bind} Substructure \xrightarrow{bind} Drug \xrightarrow{bind} Protein$	0.51
$Drug \xrightarrow{bind} Substructure \xrightarrow{bind} Drug \xrightarrow{bind} Protein$	0.51
$Drug \xrightarrow{bind} Substructure \xrightarrow{bind} Drug \xrightarrow{bind} Protein$	0.51
$Drug \xrightarrow{bind} Protein \xrightarrow{bind} Tissue \xrightarrow{bind} Protein$	0.50
$Drug \xrightarrow{bind} Protein \xrightarrow{bind} Tissue \xrightarrow{bind} Protein$	0.50
$Drug \xrightarrow{bind} ChemicalOntology \xrightarrow{bind} Drug \xrightarrow{bind} Protein$	0.50

$Protein \xrightarrow{bind} Protein \xrightarrow{bind} Protein$, and $Drug \xrightarrow{bind} Protein \xrightarrow{bind} Pathway \xrightarrow{bind} Protein$ from drug 5312137 to protein MET, whereas there is only one metapath, $Drug \xrightarrow{bind} Protein \xrightarrow{bind} Drug \xrightarrow{bind} Protein$, from drug 5312137 to protein AURKB. The metapaths from drug 5312137 to protein MET are richer than those to protein AURKB. The results indicate that the proteins ranked before the positive instances in the ground truth are more closely related to the source drug, although many top-ranked proteins are not included in the ground truth.

Finally, we compare the performance of different metapaths. Specifically, we set ε as 1×10^{-4}. For each metapath, we evaluate its top-k ($k = 500$) similar vertices based on the ground truth. Table 3 shows 10 metapath examples, with the evaluated scores of AUC no less than 0.5. We can see that metapath $Drug \xrightarrow{bind} Protein \xrightarrow{bind} Drug \xrightarrow{bind} Protein$ is the most significant compared to all other metapaths. The metapath may be explained as a drug can bind to a protein that shares another compound/drug with another protein. Most of the metapaths are actually insignificant and can be ignored when predicting links.


(a) Twitter

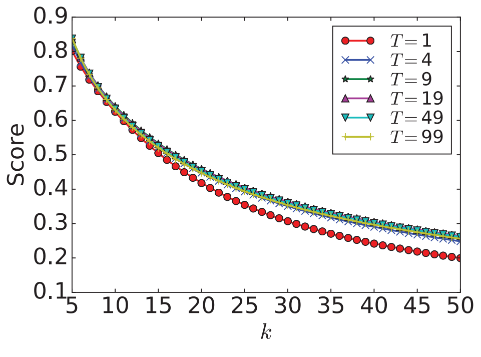
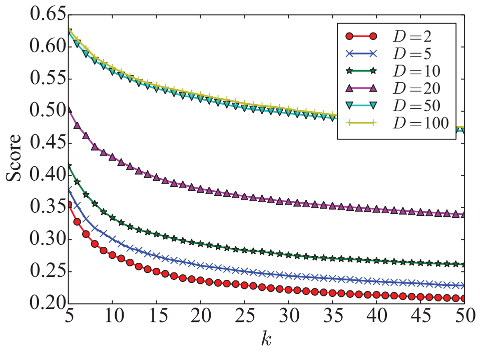

(b) Mobile

Fig. 12. Effect of T on the performance of Panther.


(a) Twitter

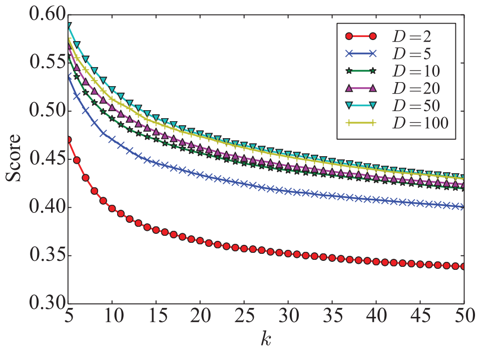

(b) Mobile

Fig. 13. Effect of D on the performance of Panther$_v$.

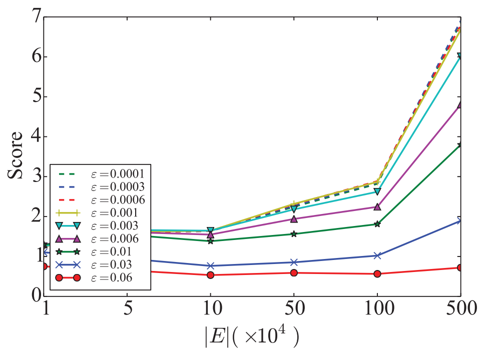
8.5 Parameter Sensitivity Analysis

We now discuss how different parameters influence the accuracy performance of our methods.

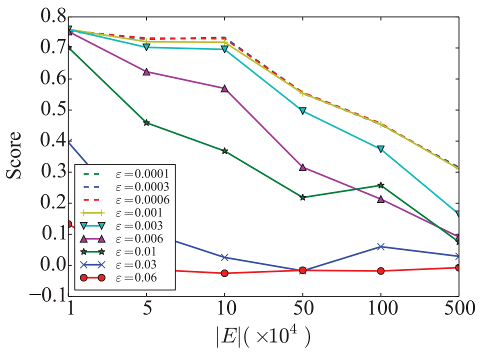
Effect of path length T. Figure 12 shows the accuracy performance of Panther evaluated by common neighbors by varying T as 1, 4, 9, 19 , 49, and 99 (i.e., varying the number vertices in a path as 2, 5, 10, 20, 50, and 100). A too small $T(<5)$ would result in inferior performance, and when increasing its value up to 5, it almost becomes stable.

Effect of vector dimension D. Figure 13 shows the accuracy performance of Panther$_v$ for mining structural hole spanners by varying the vector dimension D as 2, 5, 10, 20, 50, and 100. Generally speaking, the performance gets better when D increases, and it remains the same after D gets larger than 50. This is reasonable, as Panther$_v$ reflects the distribution of a vertex linking to the other vertices. Thus, the higher the vector dimension is, the better the representation will be. The performance will be stable when the dimension exceeds a threshold.

Effect of error-bound ε. Figure 14 shows the accuracy performance of Panther evaluated by common neighbors and that of Panther$_v$ for mining structural hole spanners on the Tencent networks with different scales by varying error-bound ε from 0.06 to 0.0001. We see that when the ratio $\frac{|E|}{(1/\varepsilon)^2}$ ranges from 5 to 20, scores of Panther are almost convergent on all datasets. And when



(a) Panther



(b) Panther$_v$

Fig. 14. Effect of ε on the performance of Panther and Panther$_v$ on different sizes of Tencent networks.

the ratio $\frac{|E|}{(1/\varepsilon)^2}$ ranges from 0.2 to 5, the scores of Panther$_v$ are almost convergent on all datasets. Thus, we can reach the conclusion that the value of $(1/\varepsilon)^2$ is almost linearly positively correlated with the number of edges in a network. Therefore, we can empirically set $\varepsilon = \sqrt{1/|E|}$.

9 CONCLUSION

In this article, we propose a sampling method to quickly conduct top-k similarity search on large networks. The algorithm is based on the idea of *random path* and solving neighborhood similarity in homogeneous networks. One extended method is to solve neighborhood similarity in heterogeneous networks, and another extended method is presented to enhance the structure similarity when two vertices are completely disconnected. We provide theoretical proofs for the error bound and confidence of the proposed algorithm. We perform an extensive empirical study and show that our algorithm can obtain top-k similar vertices for any vertex in a network approximately 300× faster than state-of-the-art methods. We also build a prototype system of recommending similar authors to demonstrate the effectiveness of our proposed method.

ACKNOWLEDGMENTS

The authors thank Pei Lee, Laks V. S. Lakshmanan, Jeffrey Xu Yu, Ruoming Jin, Victor E. Lee, Hui Xiong, Keith Henderson, Brian Gallagher, Lei Li, Leman Akoglu, Tina Eliassi-Rad, and Christos Faloutsos for sharing codes of the comparison methods. We thank Tina Eliassi-Rad for sharing the datasets. We thank Gang Fu, Bin Chen, Ying Ding, and Abhik Seal for sharing the datasets and baseline results. This work was supported by the National Basic Research Program of China (2014CB340506, 2014CB340402), the National Natural Science Foundation of China (61631013,61561130160, 61532021), the National Social Science Foundation of China (13&ZD190), the National Key Research and Develop Plan (2016YFB1000702), a research fund supported by MSRA, the Royal Society-Newton Advanced Fellowship Award, the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China (15XNLQ06, 17XNLF09).

REFERENCES

[1] Charu C. Aggarwal, Yuchen Zhao, and S. Yu Philip. 2011. Outlier detection in graph streams. In *Proceedings of the 2011 ICDE Conference (ICDE'11)*. 399–409.
[2] Nesreen K. Ahmed, Nick Duffield, Jennifer Neville, and Ramana Kompella. 2014. Graph sample and hold: A framework for big-graph analytics. In *Proceedings of the 2014 KDD Conference (IDD'14)*. 1446–1455.

[3] Kazuo Aoyama, Kazumi Saito, Hiroshi Sawada, and Naonori Ueda. 2011. Fast approximate similarity search based on degree-reduced neighborhood graphs. In *Proceedings of the 2011 KDD Conference (KDD'11)*. 1055–1063.

[4] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*. Vol. 463. ACM, New York, NY.

[5] Vincent D. Blondel, Anahí Gajardo, Maureen Heymans, Pierre Senellart, and Paul Van Dooren. 2004. A measure of similarity between graph vertices: Applications to synonym extraction and Web searching. *SIAM Review* 46, 4, 647–666.

[6] Luciana S. Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. 2006. Counting triangles in data streams. In *Proceedings of the 2006 PODS Conference (PODS'06)*. 253–262.

[7] Ronald S. Burt. 1990. Detecting role equivalence. *Social Networks* 12, 1, 83–97.

[8] Ronald S. Burt. 2009. *Structural Holes: The Social Structure of Competition*. Harvard University Press, Cambridge, MA.

[9] Bin Chen, Ying Ding, and David J. Wild. 2012. Assessing drug target association using semantic linked data. *PLoS Computational Biology* 8, 7, e1002574.

[10] G. Cormode and S. Muthukrishnan. 2005. Space efficient mining of multigraph streams. In *Proceedings of the 2005 PODS Conference (PODS'05)*. 271–282.

[11] Yuxiao Dong, Yang Yang, Jie Tang, Yang Yang, and Nitesh V. Chawla. 2014. Inferring user demographics and social strategies in mobile social networks. In *Proceedings of the 2014 KDD Conference (KDD'14)*. 15–24.

[12] Nick Duffield, Yunhong Xu, Liangzhen Xia, Nesreen Ahmed, and Minlan Yu. 2017. Stream aggregation through order samplingarXiv:1703.02693.

[13] William Feller. 2008. *An Introduction to Probability Theory and its Applications*. Vol. 2. John Wiley & Sons.

[14] Linton C. Freeman. 1977. A set of measures of centrality based on betweenness. *Sociometry* 40, 1, 35–41.

[15] Linton C. Freeman. 1979. Centrality in social networks conceptual clarification. *Social Networks* 1, 3, 215–239.

[16] Yasuhiro Fujiwara, Makoto Nakatsuji, Hiroaki Shiokawa, Takeshi Mishima, and Makoto Onizuka. 2013. Efficient ad-hoc search for personalized PageRank. In *Proceedings of the 2013 SIGMOD Conference (SIGMOD'13)*. 445–456.

[17] Kwang-Il Goh, Michael E. Cusick, David Valle, Barton Childs, Marc Vidal, and Albert-László Barabási. 2007. The human disease network. *Proceedings of the National Academy of Sciences of the United States of America* 104, 21, 8685–8690.

[18] Wentao Han, Xiaowei Zhu, Ziyan Zhu, Wenguang Chen, Weimin Zheng, and Jianguo Lu. 2016. A comparative analysis on Weibo and Twitter. *Tsinghua Science and Technology* 21, 1, 1–16.

[19] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. 2012. RolX: Structural role extraction and mining in large graphs. In *Proceedings of the 2012 KDD Conference (KDD'12)*. 1231–1239.

[20] Keith Henderson, Brian Gallagher, Lei Li, Leman Akoglu, Tina Eliassi-Rad, Hanghang Tong, and Christos Faloutsos. 2011. It's who you know: Graph mining using recursive structural features. In *Proceedings of the 2011 KDD Conference (KDD'11)*. 663–671.

[21] Paul W. Holland and Samuel Leinhardt. 1981. An exponential family of probability distributions for directed graphs. *Journal of the American Statistical Association* 76, 373, 33–50.

[22] John Hopcroft, Tiancheng Lou, and Jie Tang. 2011. Who will follow you back? Reciprocal relationship prediction. In *Proceedings of the 2011 CIKM Conference (CIKM'11)*. 1137–1146.

[23] Paul Jaccard. 1901. Étude comparative de le distribution florale dans une portion de alpes et du jura. *Bulletin de la Société Vaudoise des Sciences Naturelles* 37, 547–579.

[24] Glen Jeh and Jennifer Widom. 2002. SimRank: A measure of structural-context similarity. In *Proceedings of the 2002 KDD Conference (KDD'02)*. 538–543.

[25] M. Jha, C. Seshadhri, and A. Pinar. 2013. A space efficient streaming algorithm for triangle counting using the birthday paradox. In *Proceedings of the 2013 KDD Conference (KDD'13)*. 589–597.

[26] M. Jha, C. Seshadhri, and A. Pinar. 2015. Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In *Proceedings of the 2015 WWW Conference (WWW'15)*. 495–505.

[27] Ruoming Jin, Victor E. Lee, and Hui Hong. 2011. Axiomatic ranking of network role similarity. In *Proceedings of the 2011 KDD Conference (KDD'11)*. 922–930.

[28] Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. 2004. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics* 20, 11, 1746–1758.

[29] Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18, 1, 39–43.

[30] M. M. Kessler. 1963. Bibliographic coupling between scientific papers. *American Documentation* 14, 1, 10–25.

[31] Angelos Kremyzas, Norman Jaklin, and Roland Geraerts. 2016. Towards social behavior in virtual-agent navigation. *Science China Information Sciences* 59, 11, 112102.

[32] Mitsuru Kusumoto, Takanori Maehara, and Ken-Ichi Kawarabayashi. 2014. Scalable similarity search for SimRank. In *Proceedings of the 2014 SIGMOD Conference (SIGMOD'14)*. 325–336.

[33] Pei Lee, Laks V. S. Lakshmanan, and Jeffrey Xu Yu. 2012. On top-k structural similarity search. In *Proceedings of the 2012 ICDE Conference (ICDE'12)*. 774–785.

[34] E. A. Leicht, P. Holme, and M. E. J. Newman. 2006. Vertex similarity in networks. *Physical Review E* 73, 2, 026120.

[35] Y. Lim and U. Kang. 2015. Mascot: Memory-efficient and accurate sampling for counting local triangles in graph streams. In *Proceedings of the 2015 SIGKDD Conference (SIGKDD'15)*. 685–694.

[36] Tiancheng Lou and Jie Tang. 2013. Mining structural hole spanners through information diffusion in social networks. In *Proceedings of the 2013 WWW Conference (WWW'13)*. 837–848.

[37] Nagarajan Natarajan and Inderjit S. Dhillon. 2014. Inductive matrix completion for predicting gene–disease associations. *Bioinformatics* 30, 12, i60–i68.

[38] Mark E. J. Newman. 2006. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E* 74, 3, 036104.

[39] Jia-Yu Pan, Hyung-Jeong Yang, Christos Faloutsos, and Pinar Duygulu. 2004. Automatic multimedia cross-modal correlation discovery. In *Proceedings of the 2004 KDD Conference (KDD'04)*. 653–658.

[40] Aduri Pavan, Kanat Tangwongsan, Srikanta Tirthapura, and Kun-Lung Wu. 2013. Counting and sampling triangles from a graph stream. *Proceedings of the VLDB Endowment* 6, 14, 1870–1881.

[41] Mahmudur Rahman and Mohammad Al Hasan. 2013. Approximate triangle counting algorithms on multi-cores. In *Proceedings of the 2013 IEEE International Conference on Big Data*. 127–133.

[42] Mahmudur Rahman, Mansurul Bhuiyan, and Mohammad Al Hasan. 2012. Graft: An approximate graphlet counting algorithm for large graph analysis. In *Proceedings of the 2012 CIKM Conference (CIKM'12)*. 1467–1471.

[43] Matteo Riondato and Evgenios M. Kornaropoulos. 2014. Fast approximation of betweenness centrality through sampling. In *Proceedings of the 2014 WSDM Conference (WSDM'14)*. 413–422.

[44] Ryan A. Rossi and Nesreen K. Ahmed. 2015. Role discovery in networks. *IEEE Transactions on Knowledge and Data Engineering* 27, 4, 1112–1131.

[45] Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. 2017. Estimation of graphlet statisticsarXiv:1701.01772.

[46] Purnamrita Sarkar and Andrew W. Moore. 2010. Fast nearest-neighbor search in disk-resident graphs. In *Proceedings of the 2010 KDD Conference (KDD'10)*. 513–522.

[47] Atish Das Sarma, Sreenivas Gollapudi, and Rina Panigrahy. 2011. Estimating PageRank on graph streams. *Journal of the ACM* 58, 3, 13.

[48] Chuan Shi, Xiangnan Kong, Yue Huang, and Philip S. Yu. 2014. HeteSim: A general framework for relevance measure in heterogeneous networks. *IEEE Transactions on Knowledge and Data Engineering* 26, 10, 2479–2492.

[49] Henry Small. 1973. Co-citation in the scientific literature: A new measure of the relationship between two documents. *Journal of the American Society for Information Science* 24, 4, 265–269.

[50] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta path-based top-k similarity search in heterogeneous information networks. In *Proceedings of the 2011 VLDB Conference (VLDB'11)*. 992–1003.

[51] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. ArnetMiner: Extraction and mining of academic social networks. In *Proceedings of the 2008 KDD Conference (KDD'08)*. 990–998.

[52] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. In *Proceedings of the 2006 ICDM Conference (ICDM'06)*. 613–622.

[53] Charalampos E. Tsourakakis. 2014. Toward quantifying vertex similarity in networks. *Internet Mathematics* 10, 3–4, 263–286.

[54] Vladimir N. Vapnik and A. Ya Chervonenkis. 1971. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications* 16, 2, 264–280.

[55] Ingo Wald and Vlastimil Havran. 2006. On building fast kd-trees for ray tracing, and on doing that in O(N log N). In *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing*. 61–69.

[56] Sebastian Wernicke. 2006. Efficient detection of network motifs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 3, 4, 347–359.

[57] Yang Yang, Jie Tang, Cane Wing-Ki Leung, Yizhou Sun, Qicong Chen, Juanzi Li, and Qiang Yang. 2014. RAIN: Social role-aware information diffusion. In *Proceedings of the 2014 AAAI Conference (AAAI'14)*. 367–373.

[58] Xiong Yun, Yangyong Zhu, and S. Yu Philip. 2015. Top-k similarity join in heterogeneous information networks. *IEEE Transactions on Knowledge and Data Engineering* 27, 6, 1710–1723.

[59] Jing Zhang, Jie Tang, Cong Ma, Hanghang Tong, Yu Jing, and Juanzi Li. 2015. Panther: Fast top-k similarity search on large networks. In *Proceedings of the 2015 KDD Conference (KDD'15)*. 1445–1454.

Received August 2016; revised March 2017; accepted March 2017