Information Processing and Management 56 (2019) 102074

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Fast top-*k* similarity search in large dynamic attributed networks

Zaiqiao Meng[a,b], Hong Shen[*,a,c]

[a] School of Data and Computer Science, Sun Yat-sen University, China
[b] School of Computing Science, University of Glasgow, Glasgow, UK
[c] School of Computer Science, The University of Adelaide, Adelaide, Australia

ARTICLE INFO

Keywords:
Similarity search
Top-*k* search
Attributed network
Path sampling
Random walk

ABSTRACT

In this paper, we study the problem of retrieving top-*k* nodes that are similar to a given query node in large dynamic attributed networks. To tackle this problem, we propose a fast **A**ttribute augmented **S**ingle-source **P**ath similarity algorithm (**ASP**). Our **ASP** constructs an *attribute augmented network* that integrates both node structure and attribute similarities through similarity scores computed by an efficient *single-source path sampling* scheme. It also contains simple and effective updating schemes to maintain similarity scores for dynamic edge insertions and deletions. We provide an upper bound of the sampling size of **ASP** for obtaining an ϵ-approximation estimation of similarity scores with probability at least $1 - \delta$. We theoretically prove that the sampling size and computational complexity of ASP are significantly lower than that of deploying the currently known most effective method Panther. Implementation results from extensive experiments in both synthetic networks and real-world social networks show that our **ASP** achieves better convergence performance, runs faster than the baselines, and effectively captures both structure and attribute similarities of the nodes. Specifically, we show that our algorithm can return top-k similar nodes for any query node in the real-world network at least 44x faster than the state-of-the-art methods, and update similarity scores in about 1 second for a batch of network modifications in a network of size 1,000,000.

1. Introduction

Retrieving top-*k* similar nodes for a given query node in large dynamic attributed networks is critical to the success of further design of a number of applications such as link prediction (Martínez, Berzal, & Cubero, 2017), graph clustering (Zhou, Cheng, & Yu, 2009), friendship analysis (Celik & Dokuz, 2018) and community detection (Li, Bai, Wenjun, & Xihao, 2019). A number of algorithms have been proposed to address this problem, which can be simply classified into two categories: those based on node's local information and those based on global random walk information. Quite a few similarity metrics have been developed for those based on node's local information, classical examples including Jaccard similarity and Cosine similarity (Zhang et al., 2015). In these metrics, various local features of nodes, e.g. neighbors and attributes, can be used to formulate the similarity score. However, such local information-based metrics cannot evaluate similarities between nodes that share no local features. In contrast, well-known algorithms based on global random walk information include Personalized PageRank (PPR) (Haveliwala, 2002) and SimRank (Jeh & Widom, 2002). These global random walk-based methods provide a high-quality structure similarity measurement for nodes because of their ability to consider the global topological structure of the whole network. However, these methods suffer from high time and space complexities due to the recursive nature of the global random walk (Yu & McCann, 2015b) or the underlying matrix-vector

* Corresponding author at: School of Computer Science, The University of Adelaide, Adelaide, Australia.
E-mail addresses: zqmeng@aliyun.com (Z. Meng), hong.shen@adelaide.edu.au (H. Shen).

https://doi.org/10.1016/j.ipm.2019.102074
Received 26 November 2018; Received in revised form 4 December 2018; Accepted 2 July 2019
Available online 11 July 2019
0306-4573/

multiplications (Fujiwara, Nakatsuji, Onizuka, & Kitsuregawa, 2012). For instance, SimRank (Jeh & Widom, 2002) computes similarity between two nodes as the first-meeting probability of two random surfers, by using matrix-vector iterative multiplications, which is particularly time-consuming in large scale networks.

Moreover, many real-world networks today such as online social networks and scientific collaboration networks are not only large in size, but also contain rich node attributes and allow dynamic updates: edges and attributes are added and removed over time, aka **dynamic attributed network**. Therefore, in this paper, we study the problem **single-source similarity search** that *aims at retrieving the top-k nodes similar to a given query node q in large dynamic attributed networks*. Our goal is to approximately estimate the similarity scores of nodes for a given single query node in a faster way.

Although research on feature similarity and node similarity has gained to significant achievements (Yu & McCann, 2015b), measuring similarity in attributed networks still faces a number of challenges: (1) It is intractable to measure both structure similarity and attribute similarity between nodes in a unified metric, since attributes also provides complementary information to correlate two nodes in a network, and these two types of similarity are often measured in different metrics and scales. (2) Due to the expansion of the network scale, it is challenging for many previous algorithms to quickly and accurately measure similarities of the nodes in networks with billion size. (3) Edges and attributes in many real-world social networks evolve over time, which brings forward new demands for incremental and decremental computations.

To perform the top-*k* similarity search in large scale networks, recently a random path sampling-based method, Panther (Yu & McCann, 2015b; Zhang, Tang, et al., 2017), was proposed for handling this situation to approximately estimate similarity scores for all-pair nodes. This algorithm randomly generates paths with a specified length, and the similarity score is computed by the probability of co-occurrence of nodes in the same path. Given an error-bound and a confidence level, with the virtue of global random walks and the analysis of Vapnik-Chernovenkis (VC) dimension theory, Panther is able to accurately and quickly estimate similarities of all pairs of nodes in large networks, according to a sampling path set of bounded size. However, if we only need to retrieve similarity nodes for a single given node, the path sampling method of Panther still has to sample paths globally for all nodes beforehand, which would generate lots of irrelevant paths, resulting in huge unnecessary processing time. In addition, Panther assumes that networks are static, which is not realistic as real-world networks are evolving over time, e.g., new nodes would be added and attributes would be removed from time to time.

There are also many existing studies that attempt to cope with aforementioned challenges in attributes networks. For example, some work (Huang, Cheng, & Yu, 2015; Zhou et al., 2009) combines node similarity with attributes by using an attribute augmented graph. However, their random walk calculation involves matrix multiplication, which yields high computational complexity.

In this paper we tackle all of the three aforementioned challenges by the proposed **ASP**, a *path similarity* based method that estimates similarity scores for a single query in large dynamic attributed networks. The contributions of this paper are fourfold:

(1) We combine the structure similarity and the attribute similarity into one single metric by constructing an attribute augmented network, in which attributes are regarded as augmented nodes.
(2) We present the **ASP** algorithm with a single-source path sampling procedure for estimating the single source similarity efficiently. By utilizing the results of VC dimension learning theory (Vapnik & Chervonenkis, 2015), we theoretically prove that both the sampling size and computational complexity of **ASP** are significantly lower than that of deploying Panther.
(3) Based on that, we further propose two dynamic updating schemes for maintaining similarity scores in response to batch of updates in dynamic networks.
(4) We conduct extensive experiments in both synthetic datasets and four real-world attributed networks, and the results show that **ASP** achieves better convergence performance, runs faster than Panther, and effectively captures both structure and attribute similarities of the nodes.

The rest of the paper is organized as follows. Section 2 briefly surveys the related work. Section 3 provides a brief introduction to Panther and formally defines our task. In Section 4 we detail the main ideas of **ASP** for the task, and discuss the dynamic updating schemes. Section 5 reviews the results of our experiments. Finally, we conclude the paper in Section 6.

2. Related work

Early similarity metrics, such as Jaccard similarity and Cosine similarity, are based on the assumption that two nodes are similar if they share many local features, e.g. the same neighbors. However, these metrics cannot measure similarity between nodes without local features. In contrast, global random walk based metrics have been shown to be more effective than these local metrics in many applications as they also capture the global structure of the network. Personalized PageRank (PPR) (Haveliwala, 2002) is an effective random walk based similarity metric that iteratively computes the relevances of all nodes in a network until convergence for a given preference vector. Random Walk with Restart (RWR) (Tong, Faloutsos, & Pan, 2006) method is very similar to PPR, but it considers a random particle that starts from node rather than the request vector. The basic intuition of SimRank (Jeh & Widom, 2002) is that "two objects are similar if they are referenced by similar objects". Existing algorithms like P-Rank (Zhao, Han, & Sun, 2009), TopSim (Lee, Lakshmanan, & Yu, 2012) and SR# (Yu & McCann, 2015b) are the variations of SimRank. These method of calculate SimRank can be classified into iterative- and random walk-based method, and experimental result shows that algorithms based on iterative method often have higher accuracy while algorithms based on random walk can be more scalable (Zhang, Shao, Cui, & Zhang, 2017). Some other variations include penalized hitting probability (Zhang, Shou, Chen, Chen, & Bei, 2012) and nearest neighbor (Wu, Jin, & Zhang, 2014), and they also work with the global random walk principle. The basic approach for the global

random walk based methods is to use the global iteration method (Yu & McCann, 2015b) or matrix-vector iterative multiplications (Fujiwara et al., 2012), which are time-consuming and infeasible to handle networks with billion size and constantly changing. For example, the algorithm of computing SimRank scores in Lizorkin, Velikhov, Grinev, and Turdakov (2008) costs $O(|\mathcal{V}|^3)$ time and $O(|\mathcal{V}|^2)$ space, where $\mathcal{V}$ and $|\mathcal{V}|$ are the set of nodes in the network and the size of the set, respectively.

To further accelerate the computation of global random walk based similarity metrics, several approximate approaches have been proposed. A scheduled approximation strategy is proposed by Zhu, Fang, Chang, and Ying (2013) to compute RWR similarity. The Monte-Carlo is also an effective approach for approximately computing similarity, and typical applications using Monte-Carlo include PPR (Bahmani, Chowdhury, & Goel, 2010) and single pair SimRank (Kusumoto, Maehara, & Kawarabayashi, 2014). Lu, Gong, and Lin (2017) speed up the computation of SimRank by converting SimRank to the problem of solving a linear system in matrix form for computing all-pair SimRank scores. Wang, Chen, Che, and Luo (2018) and Wang, Lian, and Chen (2018) further accelerate the pairwise and all-pairs SimRank estimation for dynamic networks by the Monte Carlo method and local push based algorithm respectively, however, ignoring the node attributes. Moreover, most of these approaches still use iterative matrix computation, which has high computational complexity. Li et al. (2010) compute the SimRank similarities in non-iterative style by reformulating the SimRank equation using the Kronecker product and vectorization operators, but its time complexity is still $O(|\mathcal{V}|^2\gamma^4)$, where γ is the rank of graph adjacency matrix. To tackle the single-source similarity search problem, two best known algorithms, TopSim (Lee et al., 2012) and PrunPar-SR (Yu & McCann, 2015a), with the time complexities of them being $O(\Delta^{2l})$ and $O(\min\{k|\mathcal{E}|, \Delta^{2l}\})$, respectively, where Δ is the maximum degree and l is the path length. To accelerate the computation of SimRank metric, Jiang, Fu, and Wong (2017) propose a new definition of random walks, called SimRank-Aware random walk with the indexing scheme to compute SimRank similarity scores approximately. Song et al. (2018) applies the bidirectional sampling paths following the Monte Carlo to approximately compute the SimRank scores, which can handle large scale networks, but their method still needs huge time for constructing the index. Recently, Panther (Zhang et al., 2015; Zhang, Tang, et al., 2017) is proposed by Zhang et al. to evaluate the top-k similar nodes using a sampling-based method. We will briefly review this algorithm in the next section, as our proposed algorithm follows the same similarity metric, i.e. *path similarity*. Our work in this paper further accelerates the computation of this metric for the single-source similarity search problem, and extends it to attributed networks with dynamic updates.

3. Preliminaries and problem formulation

In this section, we first briefly review Panther algorithm for the problem of top-k similarity search in pure networks, and then formally define the task of top-k similarity search in attributed networks. In the problem setting of Panther, a pure networks is defined as $G = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where $\mathcal{V}$ and $\mathcal{E}$ are the sets of nodes and edges respectively, and $\mathbf{W}$ is the edge weight with ω_{uv} being edge (u, v)'s weight.

3.1. Panther

Panther (Zhang et al., 2015; Zhang, Tang, et al., 2017) is a path sampling-based method for approximately estimating similarity of all pairs of nodes in a pure network. The similarity metric used in Panther, i.e. the *path similarity*, is based on the principle of Katz Index (Zhang, Tang, et al., 2017): two nodes are similar if they frequently appear on the same paths.

In Panther algorithm, the minimum number of random paths needed to sample before it reaches an approximate estimation of the similarity between two nodes can be theoretically obtained according to the VC dimension theory (Vapnik & Chervonenkis, 2015). The main framework of Panther (Zhang et al., 2015) is outlined in Algorithm 1.

In Algorithm 1, Panther takes a pure network $G = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ without nodes' attributes as input, and returns top-k nodes that have the highest similarities to a given query node q by sampling r random paths of length l (i.e. the l-paths). The number of random paths r is calculated as:

$$r = \frac{c}{\epsilon^2}\left(\log_2\binom{l+1}{2} + 1 + \ln\frac{1}{\delta}\right), \tag{1}$$

where c is a universal positive constant, ϵ is the error-bound, δ is the confidence level, and l is the path length, i.e., the number of edges on the path, respectively.

The algorithm first initializes transition probabilities of all pairs of nodes, where the probability of walking from node u to node v is computed as $\frac{w_{uv}}{\sum_{v' \in \mathcal{N}(u)} w_{uv'}}$ with $\mathcal{N}(u) = \{v|(u, v) \in \mathcal{E}\}$ being the set of u's neighbors. Then the *GenerateRandomPath*(G, r) procedure randomly samples r paths (Step 3). Specifically, the algorithm randomly selects a node v of G as starting point, and conducts random walks of l steps from v to its neighbors according their transition probabilities. After that, the algorithm accumulates similarity scores for each node pairs by iterating all the sampled paths (Step 4–6). Finally, the algorithm sorts nodes by the similarity scores based on a heap structure to return a ranked list of top-k similar nodes.

Given a query q, Panther can approximately estimate similarity scores for *all pairs* of nodes in an error-bound ϵ with probability $1 - \delta$, within minimum r random paths. To the best of our knowledge, Panther is currently the fastest algorithm for computing similarity for all-pair nodes in pure networks. However, the limitation of Panther is that, if we only need to query top-k similar nodes for a single query q, the algorithm still needs to sample r random paths to ensure the approximate ratio. Since Panther generates paths by random walk starting from a random node, in which the sampled paths may not contain the query node and contribute nothing to compute the similarity scores between other nodes to the query node q. This makes the path sampling method of Panther inefficient

Input : A pure network $G = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, path length l, parameters c, ϵ, δ, k and a query q.
Output: Top-k similar nodes to q.

Initialize transition probabilities of all pairs of nodes in G
Compute $r = \frac{c}{\epsilon^2}\left(\log_2 \binom{l+1}{2} + 1 + \ln \frac{1}{\delta}\right)$
$\mathcal{P} \leftarrow$ GenerateRandomPath(G, r)
foreach $p \in \mathcal{P}$ **do**
    foreach *Unique* $u \in p$ **do**
        $s_{qu} += \frac{1}{r}$
return *Top-k similar nodes according to* s_{qu}

Algorithm 1. Framework of the Panther algorithm.

in the problem of top-k similarity search for a query node.

3.2. Problem formulation

Traditional methods of computing similarities between two nodes such as Panther mainly work with pure networks, and thus may not work well in the context of attributed networks where nodes are associated with their own attributes. In contrast, our paper focuses on attributed network rather than pure network, where we define an attributed network as an undirected weighted attributed network denoted as $G(\mathcal{V}, \mathcal{E}, \mathbf{W}, \mathbf{A})$. Here $\mathbf{A}$ is a $|\mathcal{V}| \times m$ binary matrix describing m binary attributes associated with each node in $\mathcal{V}$.

We use a binary representation for each categorical attribute. Though in principle there are other types of attributive variables, e.g., age, messages and other real-valued variables, etc., they can be clustered into categorical variables via vector quantification, or discretized to categorical variables.

Accordingly, the task we aim at addressing in this paper is to retrieve top-k nodes that are similar to a given query node in an attributed network. A formal definition of our problem is presented as follows.

Problem. Given an undirected weighted attributed network $G(\mathcal{V}, \mathcal{E}, \mathbf{W}, \mathbf{A})$, a positive integer k, and a query node $q \in \mathcal{V}$, quickly and approximately retrieve a set of top-k most similar nodes, i.e. $\mathcal{K}$, to query q (both in terms of structure similarity and attribute similarity), and ensure that:

$$\sup_{u \in \mathcal{K}} |s^*_{qu} - s_{qu}| \leq \sup_{u \in \mathcal{V}} |s^*_{qu} - s_{qu}| \leq \epsilon, \tag{2}$$

in a probability of at least $1 - \delta$, where s^*_{qu} and s_{qu} are the golden similarity of two nodes q and u according to their real distributions and the empirical estimated similarity between them respectively.

4. The proposed method

To tackle the problem, we propose **ASP**, an **A**ttribute augmented **S**ingle-source **P**ath similarity algorithm. Our proposed **ASP** is also based on *path similarity* (Zhang et al., 2015), but it differs from Panther in at least two aspects: (1) It is a more efficient global random walk based metric that is able to globally measure the similarities to a single query node. (2) Unlike Panther that only considers network topology to compute similarities, **ASP** measures similarity of each node pair by considering both the nodes' structure and attributes. Although there has been a variety of other similarity metrics (e.g. SimRank (Jeh & Widom, 2002) and PPR (Haveliwala, 2002)), they either have high computational complexity or have difficulties to capture nodes' augmented information, such as node attributes. There are also many methods for solving attribute similarity, such as Jaccard similarity and Cosine similarity, but they ignore the connections between nodes.

The overview of our **ASP** algorithm is outlined in Algorithm 2. To incorporate the attribute similarity into path similarity, **ASP** constructs an *attribute augmented network* that regards attributes of nodes as augmented nodes (Step 1). Then, **ASP** conducts the *single-source path sampling* procedure based on the attribute augmented network to sample random paths w.r.t the query (Step 4). Finally, the algorithm accumulates path similarity scores according to the sampled paths and returns the top-k nodes similar to the query node. Additionally, to consider the dynamic updates, we further propose two updating schemes for maintaining the similarity scores in dynamic networks. We will detail these main components of **ASP** in the following subsections.

4.1. Combining attribute similarity

4.1.1. Attribute augmented network

To combine the structure similarity with the attribute similarity, we construct an *attribute augmented network* from the original attributed network to explicitly represent node attributes as attribute augmented nodes. Similar idea can be found in Zhou et al. (2009) and Huang et al. (2015). A formal definition is as follows.

Definition 1. An **attribute augmented network** G is denoted as

$\mathcal{G} = (\mathcal{V} \cup \mathcal{V}_A, \mathcal{E} \cup \mathcal{E}_A, \mathbf{W})$ where $\mathcal{V}$ is a set of nodes and $\mathcal{V}_A$ is a ground set of attributes in $\mathcal{G}$. $a \in \mathcal{V}_A$ is an attribute node in $\mathcal{V}_A$ of $\mathcal{G}$. $\mathcal{E} = \{(u, v) | u, v \in \mathcal{V}\} \subseteq \mathcal{V} \times \mathcal{V}$ is the structure edge set. $\mathcal{E}_A = \{(u, a) | u \in \mathcal{V}, a \in \mathcal{V}_A\} \subseteq \mathcal{V} \times \mathcal{V}_A$ is the attribute edge set. An edge $(v, a) \in \mathcal{E}_A$ iff $v \in \mathcal{V}$ has an attribute $a \in \mathcal{V}_A$. $\mathbf{W}$ is the matrix of the weights of structure edges in G.

Fig. 1 shows an example describing how to build an attribute augmented network from a toy attributed network, which consists of five nodes and five attributes. The five attributes are represented as nodes (the square nodes) and twelve attribute associations (dotted lines) are linked in the attribute augmented network (Fig. 1(b)). With this newly inserted attribute edges, nodes sharing common attributes become more densely connected, thus the attribute similarity can be measured by the sampled paths from this attribute augmented network. The path sampling method therefore can be used in this attribute augmented network to measure both structure similarity and attribute similarity based on the path similarity metric. Then the path similarity can be measured by the principle: two nodes are similar if they frequently appear on the same paths and share common attributes.

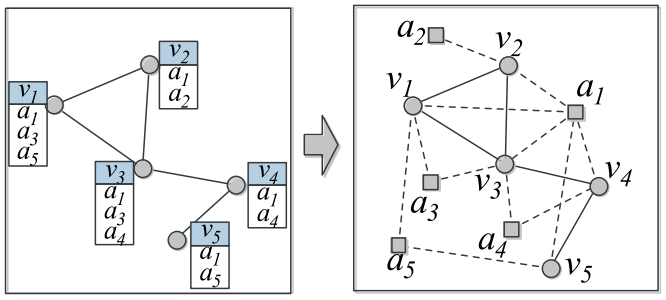
4.1.2. Transition probability

Panther (Zhang, Tang, et al., 2017) conducts random walks of l steps by using the proportional edge weight as their transition

Input : An attributed network $G = (\mathcal{V}, \mathcal{E}, \mathbf{W}, \mathbf{A})$, path length l, parameters c, ϵ, δ, k and a query q.
Output: Top-k similar nodes to q.

$\mathcal{G} \leftarrow$Construct an attribute augmented network for G
$\mathbf{P}_{\mathcal{G}} \leftarrow$Initialize transition probabilities of all pairs of nodes in $\mathcal{G}$
Compute $r = \frac{c}{\epsilon^2}\left(\log_2 l + 1 + \ln \frac{1}{\delta}\right)$
$\mathcal{P} \leftarrow$ Single-Source Path Sampling($\mathbf{P}_{\mathcal{G}}, r, l, q$)
foreach $p \in \mathcal{P}$ **do**
    foreach *Unique* $u \in p$ **do**
        $s_{qu} += \frac{1}{r}$
return *Top-k similar nodes according to* s_{qu}

Algorithm 2. Overview of the **ASP** algorithm.



(a) Attributed network (b) Attribute augmented network

Fig. 1. Building an attribute augmented network from a toy attributed network.

probability to distinguish different importance of their neighbors. However, in attribute augmented network $\mathcal{G}$, all attribute edges are weighted equally as all the attributes are binary-valued. Attributes may also have different importance in measuring the attribute similarity. For example, in Fig. 1(b), a random walk through the attribute node a_1 may be useless, because all the structure nodes associate attribute a_1. In contrast, attribute node a_3 seems to be more useful than a_1, because it is only shared between v_1 and v_3, which provides the additional information to show that node v_1 is more similar to v_3 than v_2.

In this paper, we use $p(a) = \frac{|\mathcal{E}_a|}{|\mathcal{E}_A|}$, the probability of attribute $a \in \mathcal{V}_A$ appearing in all the nodes, to distinguish different weight contributing to the similarity, where $|\mathcal{E}_a|$ is the number of edges connected to a.

The transition matrix $\mathbf{P}_{\mathcal{G}} = \begin{bmatrix} \mathbf{P}_{\mathcal{G}}^{\mathcal{V}\times\mathcal{V}} & \mathbf{P}_{\mathcal{G}}^{\mathcal{V}\times\mathcal{V}_A} \\ \mathbf{P}_{\mathcal{G}}^{\mathcal{V}_A\times\mathcal{V}} & \mathbf{P}_{\mathcal{G}}^{\mathcal{V}_A\times\mathcal{V}_A} \end{bmatrix}$ is a $|\mathcal{V} \oplus \mathcal{V}_A|$ by $|\mathcal{V} \oplus \mathcal{V}_A|$ matrix, with $\mathbf{P}_{\mathcal{G}}^{\mathcal{V}\times\mathcal{V}}$, $\mathbf{P}_{\mathcal{G}}^{\mathcal{V}\times\mathcal{V}_A}$, $\mathbf{P}_{\mathcal{G}}^{\mathcal{V}_A\times\mathcal{V}}$, and $\mathbf{P}_{\mathcal{G}}^{\mathcal{V}_A\times\mathcal{V}_A}$ being blocks in it, i.e., the transition probabilities from a structure node to a structure node, from a structure node to an attribute node, from an attribute node to a structure node, and from an attribute node to an attribute node, respectively. Here $\oplus$ is the concatenation operator for two sets.

For a structure node u, it can either walk to a structure node or an attribute node with an equal probability. Assuming that the weight of a structure edge (u, v) is ω_{uv}, the transition probability from structure node u to structure node v is:

$$\mathbf{P}_{\mathcal{G}}^{\mathcal{V}\times\mathcal{V}}(u, v) = \begin{cases} \frac{\omega_{uv}}{\sum_{w\in\mathcal{N}(u)}\omega_{uw}}, & if\ (u, v) \in \mathcal{E} \\ 0, & otherwise, \end{cases} \tag{3}$$

where $\mathcal{N}(u) = \{v|(u, v) \in \mathcal{E}\}$ represents the neighbor set of structure node u. The transition probability from a structure node u to an attribute node a is defined as:

$$\mathbf{P}_{\mathcal{G}}^{\mathcal{V}\times\mathcal{V}_A}(u, a) = \begin{cases} \frac{1-p(a)}{\sum_{b\in\mathcal{V}_A}(1-p(b))}, & if\ (u, a) \in \mathcal{E}_A \\ 0, & otherwise. \end{cases} \tag{4}$$

For an attribute node a, it can randomly walk with an equal probability to any structure nodes that associate attribute a, and the probability is defined as:

$$\mathbf{P}_{\mathcal{G}}^{\mathcal{V}_A\times\mathcal{V}}(a, u) = \begin{cases} \frac{1}{|\mathcal{V}(a)|}, & if\ (a, u) \in \mathcal{E}_A \\ 0, & otherwise, \end{cases} \tag{5}$$

where $\mathcal{V}(a) = \{u|(a, u) \in \mathcal{E}_A\}$ represents the set of structure nodes connected to attribute node a. The transition probability between any two attribute nodes is 0, i.e.,

$$\mathbf{P}_{\mathcal{G}}^{\mathcal{V}_A\times\mathcal{V}_A}(a, a) = 0, \forall\ a, b \in \mathcal{V}_A. \tag{6}$$

4.2. Single-source path sampling

In order to avoid sampling irrelevant paths, we propose the *single-source path sampling* method, and summarize the subroutine in Algorithm 3.

In this subroutine, the algorithm takes a transition matrix $\mathbf{P}_{\mathcal{G}}$ as input, and iteratively samples r paths of length l according to the transition probability. Our random walk procedure starts from the query q, which is different from Panther. Specifically, the algorithm first generates a random number to specify the index of the query node in a path array (step 3), then conducts random walks from the query node to fill the whole path array according to the transition matrix $\mathbf{P}_{\mathcal{G}}$ (step 5-8). As a result, all the sampled paths contain $l + 1$ nodes, one of which is the query q. We note that our *single-source path sampling* can be directly applied to the pure networks, which we referred to as **SP** (**S**ingle-source **P**ath similarity).

The sampling method in attribute augmented networks differs slightly from that in pure networks. In an attribute augmented

Input : A transition matrix $\mathbf{P}_{\mathcal{G}}$, number of paths r, path length l and a query q.
Output: Path set $\mathcal{P}$.

$\mathcal{P} \leftarrow \emptyset$
foreach *1 to r* **do**
  $rand \leftarrow$ GenerateRandomNumber$(0, l)$
  $p[rand] = q$
  `/* Sample the first part of the path */`
  for $i = rand - 1, \ldots, 0$ **do**
    $p[i] \leftarrow$ SampleNextNode$(p[i + 1])$
  `/* Sample the second part of the path */`
  for $j = rand + 1, \ldots, l$ **do**
    $p[j] \leftarrow$ SampleNextNode$(p[j - 1])$
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{p\}$
return $\mathcal{P}$

Algorithm 3. Single-source path sampling.

network, an attribute node in a random walk only plays a role to change the transition probability between two structure nodes, and the *SampleNextNode()* procedure do not collect attribute nodes into the paths. With this treatment, our **ASP** can effectively combine structure similarity and attribute similarity into a unified similarity metric, and the sample size as well as the computation complexity can be greatly lowered comparing to Panther. To obtain relative guarantees on the approximation, the least number of paths needed to sample in both **ASP** and **SP** is calculated by

$$r = \frac{c}{\epsilon^2}\left(\log_2 l + 1 + \ln\frac{1}{\delta}\right). \tag{7}$$

We give a theoretical analysis for the number of random path in the next subsection.

4.3. Theoretical analysis

We use the VC dimension theory (Vapnik & Chervonenkis, 2015) to demonstrate how large a sample size can obtain a desired ϵ-approximate similarity scores with probability at least $1-\delta$. The VC dimension of a class of subsets defined on a set of points is a measure of the complexity or expressiveness of such class (Vapnik & Chervonenkis, 2015). In statistical learning theory, the VC dimension can predict a probabilistic upper bound on the error of its empirical estimation.

Let $\mathcal{D}$ be a domain and $\mathcal{R}$ be a collection of subsets from $\mathcal{D}$. We call $\mathcal{R}$ a range set on $\mathcal{D}$. Given $\mathcal{B} \subseteq \mathcal{D}$, the projection of $\mathcal{R}$ on $\mathcal{B}$ is the set $P_{\mathcal{R}}(\mathcal{B}) = \{\mathcal{B} \cap \mathcal{A}: \mathcal{A} \in \mathcal{R}\}$. We say that the set $\mathcal{B}$ is shattered by $\mathcal{R}$, if $P_{\mathcal{R}}(\mathcal{B}) = 2^{\mathcal{B}}$, where $2^{\mathcal{B}}$ denotes the power set of set $\mathcal{B}$ (all subsets of $\mathcal{B}$). The VC dimension of $\mathcal{R}$, denoted as $VC(\mathcal{R})$, is the cardinality of the largest subset of $\mathcal{D}$ that is shattered by $\mathcal{R}$. For a set $\mathcal{A} \subseteq \mathcal{D}$, let $\varphi(\mathcal{A})$ be the probability that a sample from φ belongs to the set $\mathcal{A}$, and let $\varphi_{\mathcal{S}}(\mathcal{A})$ be the empirical average of $\varphi(\mathcal{A})$ on $\mathcal{S}$.

Definition 2. Vapnik and Chervonenkis (2015) Let φ be a probability distribution on $\mathcal{D}$. For $\epsilon \in (0, 1)$, an ϵ-approximation to $(\mathcal{R}, \varphi)$ is a set $\mathcal{S}$ of elements in $\mathcal{D}$ such that

$$\sup_{\mathcal{A}\in\mathcal{R}} |\varphi(\mathcal{A}) - \varphi_{\mathcal{S}}(\mathcal{A})| \le \epsilon. \tag{8}$$

Theorem 1. Har-Peled and Sharir (2011) Let $\mathcal{R}$ be a range set on a domain $\mathcal{D}$ with $VC(\mathcal{R}) \le d$. Given $\epsilon, \delta \in (0, 1)$, let $\mathcal{S}$ be a collection of $|\mathcal{S}|$ points from $\mathcal{D}$ sampled according to φ, with

$$|\mathcal{S}| = \frac{c}{\epsilon^2}(d + \ln\frac{1}{\delta}). \tag{9}$$

Then $\mathcal{S}$ is a relative ε-approximation to $(\mathcal{R}, \varphi)$ with probability at least $1-\delta$.

From this theorem, if we can bound the VC-dimension of $\mathcal{R}$, it is possible to build an ϵ-approximation probability as an unbiased estimator of φ by randomly sampling points of size $|\mathcal{S}|$ from the domain.

Based on these definitions and theorems, we can define the *golden similarity* of two nodes. Let domain Π denote all the possible l-paths in a pure network G or an attribute augmented network $\mathcal{G}$. With the principle of path similarity (Zhang, Tang, et al., 2017), the similarity between node u and node v can be defined as:

$$s^*_{uv} = \frac{|\Pi_{uv}|}{|\Pi|}, \tag{10}$$

where Π_{uv} is the domain of all the possible l-paths that contains u and v.

Accordingly, the similarity of u to a query node q for single-source similarity search is defined as:

$$s^*_{qu} = \frac{|\Pi_{qu}|}{|\Pi_q|}, \tag{11}$$

where Π_q is the domain of all the l-paths containing query q.

Clearly, we cannot directly obtain s^*_{uv} and s^*_{qu} as enumerating Π and Π_q are computationally infeasible. However, we can obtain an approximate estimation of s^*_{uv} and s^*_{qu} according to the sampled random paths from Π and Π_q, which techniques has been successfully applied in Zhang, Tang, et al. (2017) and Riondato and Kornaropoulos (2016). Hence, in our single-source similarity search problem, the *empirical estimated similarity* of u to the query q is defined as:

$$s_{qu} = \frac{|\mathcal{P}_{qu}|}{|\mathcal{P}|}, \tag{12}$$

where $|\mathcal{P}_{qu}|$ is the number of sampled paths that contain node u and query node q, and $|\mathcal{P}|$ is the number of all sampled paths sampled by **SP** or **ASP**. For **SP**, we now define the range set of path similarity sores to a single query in a pure network G. Accordingly, we define the range set $\mathcal{R}_q$ on domain Π_q to be $\mathcal{R}_q = \{\mathcal{P}_{qu}: u \in \mathcal{V}/q\}$. For the single-source similarity search problem in an attribute augmented network $\mathcal{G}$, **ASP** regards an attribute node in sampled path as a role to change the transition probability between two structure nodes, and does not include it into the path. Hence, the range set of path similarity sores in attributed network is also defined as $\mathcal{R}_q = \{\mathcal{P}_{qu}: u \in \mathcal{V}/q\}$. Lemma 1 shows an upper bound of the VC dimension of $\mathcal{R}_q$ in both pure network and attributed network.

Lemma 1. $VC(\mathcal{R}_q) \le \log_2 l + 1$.

Proof. We prove it by contradiction. Assume that $VC(\mathcal{R}_q) = d > \log_2 l + 1$, then $l < 2^{d-1}$.

W.l.o.g. assume that set $\mathcal{B}$ be the largest subset of Π_q that is shattered by $\mathcal{R}_q$, such that $P_{\mathcal{R}_q}(\mathcal{B}) = \{\mathcal{B} \cap \mathcal{A}: \mathcal{A} \in \mathcal{R}_q\}$ and $P_{\mathcal{R}_q}(\mathcal{B}) = 2^{\mathcal{B}}$. According to the definition of $VC(\mathcal{R}_q)$, the cardinality of $\mathcal{B}$ is d. Therefore, there are 2^d distinct ranges in $\mathcal{B}$, and for each path $p \in \mathcal{B}$, there are also 2^{d-1} distinct ranges in $\mathcal{B}$ that contain the path p. By definition of $\mathcal{B}$, for each nonempty subset $\mathcal{B}_i \subseteq \mathcal{B}$, we have $\mathcal{B}_i \cap \mathcal{R}_q \subseteq \mathcal{R}_q$, hence there are also 2^{d-1} distinct ranges in $\mathcal{R}_q$ that contain the path p.

Note that all the paths sampled by **ASP** and **SP** contain a query q and l other nodes. Hence, in addition to the query q and the duplicate nodes, the number of ranges in $\mathcal{R}_q$ that p belongs to is at most l. That means $2^{d-1} \leq l$, which is contradict to our preliminary assumption that $l < 2^{d-1}$.

Hence, $VC(\mathcal{R}_q) = d \leq \log_2 l + 1$.

□

A similar idea of proving can be found in Riondato and Kornaropoulos (2016) and Zhang, Tang, et al. (2017). Based on the VC dimension of $\mathcal{R}_q$, we now can provide theoretical guarantee for the number of sampled paths to achieve an error-bound ϵ with probability $1 - \delta$.

Theorem 2. Given an error-bound ϵ, and a confidence level $1 - \delta$, **SP** (**ASP**) can obtain an ϵ-approximation estimation for single-source similarity search in a pure network (an attributed network) with probability of at least $1 - \delta$ with at least r random paths, where $r = \frac{c}{\epsilon^2}\left(\log_2 l + 1 + \ln\frac{1}{\delta}\right)$.

Proof. From Lemma 1 we have $VC(\mathcal{R}_v) \leq d = \log_2 l + 1$. Plugging it into Theorem 1, we obtain: $r = \frac{c}{\epsilon^2}\left(\log_2 l + 1 + \ln\frac{1}{\delta}\right)$. According to definition of range set $\mathcal{R}_q$ and Theorem 1, we directly conclude the theorem.

□

4.4. Complexity analysis

The time complexity of constructing an attribute augmented network is $O(|\mathcal{V}||\overline{\mathcal{A}(u)}|)$, where $|\overline{\mathcal{A}(u)}|$ is the average number of attributes of each node. In Algorithm 3, generating r random paths of length l for a pure network needs $O(rl\log_2\bar{d})$, and $O(rl\log_2\bar{d}_a)$ for an attribute augmented network, where $\bar{d}$ and $\bar{d}_a$ are the average degree of the pure network and the attribute augmented network respectively. By using a heap structure, obtaining top-k similar nodes for a single query according to the path similarity scores needs $O(|\mathcal{V}|\log_2 k)$ time. Therefore, the time complexity of **SP** and **ASP** is $O(|\mathcal{V}|\log_2 k + rl\log_2\bar{d})$ and $O(|\mathcal{V}|(|\overline{\mathcal{A}(u)}| + \log_2 k) + rl\log_2\bar{d}_a)$ respectively.

Once we have already established the attribute augmented network, the main time cost of top-k similarity search is linear to l and r, where r is a function of constant c, path length l, error-bound ϵ, and confidence level δ. Our random path based methods can easily be parallelized, which makes our algorithms scale for massive networks with billions of edges.

4.5. Maintain similarity with batch updates

In this subsection, we discuss dynamic updates for the top-k similarity search problem. The problem setting is that: we have already sampled a set of paths, and initially computed the approximate similarity scores of nodes for a query q; upon seeing a batch of updates in nodes, edges or attributes, we expect to maintain the approximate similarity scores efficiently. This problem is of practical significance, as many real-world social networks today are highly dynamic and evolving rapidly, and these platforms need to periodically update the similarity scores of users for downstream applications.

There are several types of modifications in a real-world attributed network, e.g. a node insertion, a node deletion, a change of attribute value, a deletion of an edge and so on. We notice that all these modifications can be regarded as a sequence of edge insertions/deletions. A change of a node-attribute association may differs from that of a relational edge between nodes, but it is very convenient to cope with this in our algorithm because we treat attributes as nodes in the attribute augmented network. As a result, whenever a node, an attribute or an edge of an attributed network G is modified, we only need to focus on the modifications in its attribute augmented network $\mathcal{G}$. Let $\mathcal{G}^*$ be the new attribute augmented network that has accepted a batch of modifications. We collect all the unit modifications into two sets: edge deletion set C_d and edge insertion set C_i, and handle them separately.

4.5.1. Edge deletion

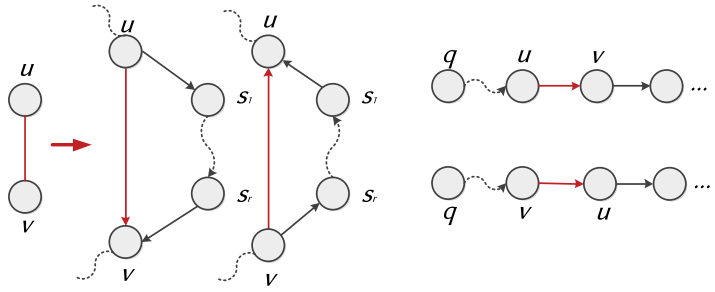
An edge deletion invalidates the paths that contain that edge, making the transition probability changed and excluding the contribution of similarity scores of other nodes on that path. Thus, given a deletion of edge (u, v) in $\mathcal{G}$, we first retrieve all the affected paths containing (u, v), then update the transition probability of the new network $\mathcal{G}^*$ and subtract similarity scores of nodes on these paths which are tailed after edge (u, v). Finally, we resample a new path p^*_{uv} to replace p_{uv} and recompute the approximate similarity scores according to p^*_{uv}. Algorithm 4 shows the pseudo-code for updating similarity scores after a batch of edge deletions. In particular, given an affected path p_{uv}, we use $\hat{p}_{uv}$ to denote a path that keeps the head part before u and truncates other nodes after u. So the *ResamplePath()* subroutine samples a new path from the last node u of $\hat{p}_{uv}$ according to the new the transition probability. To quickly retrieve affected paths, we build an *edge-to-path index* that indexes paths to edges, with which we can retrieve the affected paths for an arbitrary edge in $O(1)$ time.

Building the edge-to-path index can be done in the initial sampling phase of our algorithm, and needs $O(rl)$ time and space.

Input : A new attribute augmented network $\mathcal{G}^*$, a batch of edge deletions C_d, path length l and a query q.
Output: Similarity scores s_q.

UpdateTransitionProbability($\mathcal{G}^*$)
foreach $(u, v) \in C_d$ **do**
    $\mathcal{P}_{uv} \leftarrow$RetrieveRelatedPaths (u, v)
    foreach $p_{uv} \in \mathcal{P}_{uv}$ **do**
        foreach *Unique* $s \in p_{uv}$ *and tail after* v **do**
            $s_{qs} -= \frac{1}{R}$
        $p^*_{uv} \leftarrow$ ResamplePath $(\mathcal{G}^*, \hat{p}_{uv}, T)$
        foreach *New sampled node* $s \in p^*_{uv}$ **do**
            $s_{qs} += \frac{1}{R}$
return s_q

Algorithm 4. Update similarities after a batch of edge deletions.



(a) An edge (u, v) insertion would shortens the paths and changes the probability of generating these old paths

(b) Sample new paths from u and v

Fig. 2. Edge insertion and resampling.

4.5.2. Edge insertion

The insertion of an edge (u, v) on $\mathcal{G}$ will also cause changes to its transition probability. As described in Fig. 2(a), when an undirected edge (u, v) gets inserted, the probability of generating paths that contain u and v would be changed, because u (or v) have an additional chance to directly move to v (or u) in the random walk process. Therefore, a new set of paths need to be sampled in order to make the affected paths still be valid samples of the new graph $\mathcal{G}^*$. Recall that $s_{qu} = \frac{|\mathcal{P}_{qu}|}{|\mathcal{P}|}$ represents the estimated probability of node u appearing among all the paths. Whatever nodes walk to node u, it will choose to walk to v or to its other neighbors according to the transition probability of $\mathcal{G}^*$. The paths to u's other neighbors that have already sampled still in the range set of path similarity of the $\mathcal{G}^*$. Therefore, we need to sample a new set of paths having the new edge (u, v), and the number of new paths must be equal to $r_u^* = r \times s_{qu} \times p_{\mathcal{G}}^*(u, v)$, where r is the number of affected paths, $p_{\mathcal{G}}^*(u, v)$ is the new transition probability that node u walks to v. However, it is difficult to get such a set of paths that contain both q, u and v, because the nodes in paths are all stochastic. To cope with this issue, we directly construct a pseudo path $[q, u, v]$ and use it as input of the *ResamplePath()* subroutine to generate a new path of size l. Fig. 2(b) shows an example to illustrate the basic idea. For directed networks, we need to sample new paths for both directions. Algorithm 5 shows the pseudo-code for updating similarity scores after a batch of edge insertions.

In both Algorithms 4 and 5, the new sampled paths p_{qv}^* of $\mathcal{G}^*$ is randomly sampled according to the distribution of s_q on domain Π_q, and the number of paths $r_v^* + r$ of new sampled paths is clearly larger than initial sampled paths r, thus the new updated similarity scores still ensure an ϵ-approximation with probability as least $1 - \delta$, according to Theorem 2. We notice that our batch updating schemes can also be applied in Panther and **SP**.

5. Experiments

5.1. Research questions

The research questions guiding the remainder of the paper are:

(RQ1) How is the convergence performance of **SP** and **ASP**?
(RQ2) What is the time efficiency of our algorithms?
(RQ3) How is the time efficiency when they get a relative stable performance?
(RQ4) What is the accuracy performance of our algorithms in evaluating both the structure similarity and attribute similarity?
(RQ5) How is the performance of our methods in processing dynamic updates for large dynamic attributed networks?

5.2. Datasets

We consider four real-world network datasets with their brief statistical information listed in Table 1. The Facebook dataset is downloaded from SNAP.[1] This dataset is built from profile and relation data of 10 users' ego-networks in Facebook, and the attributes are constructed by their user profiles. The DBLP dataset is from the DBLP public bibliography data.[2] We build a coauthor network by extracting from the papers in top 172 computer science conferences (Tiers A and B from China Computer Federation[3]). We treat each author as a node, each collaboration relationship of paper as an edge. The 172 conferences are viewed as the binary attributes of each node. The AMiner coauthor dataset is downloaded from AMiner, which is an academic social network.[4] This network dataset is also built by collaboration relationships among the authors, but the difference between it and the DBLP dataset is that, the attributes in

[1] http://snap.stanford.edu/data/ .
[2] http://dblp.uni-trier.de/xml/ .
[3] http://www.ccf.org.cn/ .
[4] https://aminer.org/AMinerNetwork .

Input : A new attribute augmented network $\mathcal{G}^*$, a batch of edge insertions C_i, path length l and a query q.
Output: Similarity scores s_q.

UpdateTransitionProbability($\mathcal{G}^*$)
foreach $(u, v) \in C_i$ **do**
    $r_u^* = r \times s_{qu} \times \mathbf{P}_{\mathcal{G}}^*(u, v)$
    for *1 to* r_u^* **do**
        $p_{uv}^* \leftarrow$ ResamplePath $(\mathcal{G}^*, [q, u, v], l)$
        foreach *New sampled node* $s \in p_{uv}^*$ **do**
            $s_{qs} += \frac{1}{R}$
    $r_v^* = r \times s_{qv} \times \mathbf{P}_{\mathcal{G}}^*(v, u)$
    for *1 to* r_v^* **do**
        $p_{vu}^* \leftarrow$ ResamplePath $(\mathcal{G}^*, [q, v, u], l)$
        foreach *New sampled node* $s \in p_{vu}^*$ **do**
            $s_{qs} += \frac{1}{R}$
return s_q

Algorithm 5. Update similarities after a batch of edge insertions.

Table 1
Statistics of real-world datasets.

Datasets	Nodes $\|\mathcal{V}\|$	Edges $\|\mathcal{E}\|$	Attributes m
Facebook	4039	88,234	1406
DBLP	73,242	373,797	172
AMiner	1,560,640	4,258,946	2,868,034
Google+	28,942,911	462,994,069	4,443,631

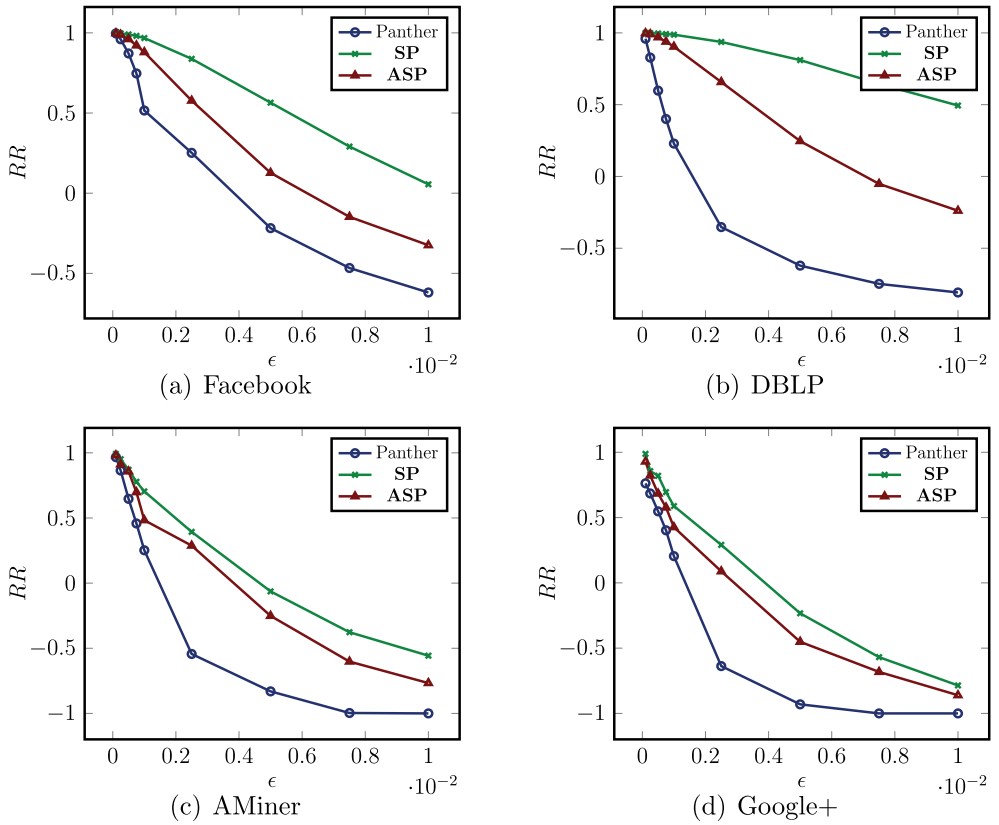


Fig. 3. Comparison on convergence.

AMiner dataset are constructed by the extracted key terms of their papers. The Google+ network dataset was crawled from Google+ social network by Gong et al. (2014). In our experiments, we only use one of the four snapshots, including both social structure and node attributes.

We also generate a collection of synthetic network datasets by using the Erdös-Rényi (ER) random network model (Meng et al., 2018), and randomly generate a fix size of attributes for each nodes in these synthetic networks.

5.3. Parameters and environment

We empirically set $l = 5$, $\delta = 0.1$ in all the experiments. All experiments are conducted on a Ubuntu 14.04 server with two Intel Xeon E5-2683 v3 (2.0GHz) CPU and 512G RAM. All algorithms are implemented by C++. In our paper, we only evaluate our algorithms (**ASP** and **SP**) comparing with Panther, since it uses the same similarity metric as ours. We refer the reader to Zhang et al. (2015), Zhang, Tang, et al. (2017) for additional comparison with other similarity metrics.

5.4. Evaluation metrics

We use *Ranking Robustness score* (Cummins, 2014) to quantitatively evaluate the convergence performance of our algorithms. The *Ranking Robustness score* is defined as $RR = \frac{2}{|\mathcal{L}|(|\mathcal{L}|-1)} \sum_{l_1,l_2 \in \mathcal{L}} \rho(l_1, l_2)$, where $\mathcal{L}$ is a set of ranked result lists returned by the search algorithms, and ρ is the *Spearman's rank correlation coefficient*. *RR* is range from − 1 to + 1, where a positive *RR* implies a positive correlation between two ranked lists and fully opposed for a correlation of − 1.

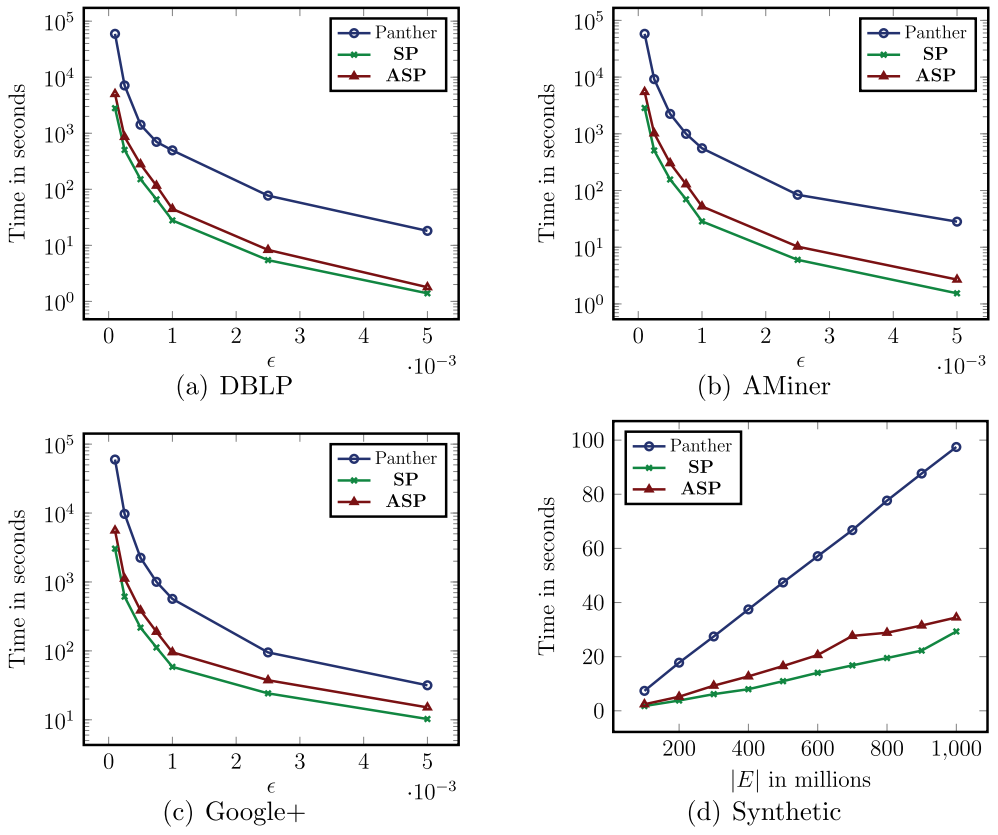


Fig. 4. Comparison on running time.

Table 2
Running time with same convergence performance. The best performance is underlined.

Dataset	Facebook		DBLP		AMiner		Google+	
Method	ϵ	time	ϵ	time	ϵ	time	ϵ	time
Panther	0.0005	0.39 h	0.00025	2.56 h	0.00025	2.69 h	0.000025	47.5 h
SP	0.0025	4.66 s	0.005	1.53 s	0.0005	217.32 s	0.0005	603.32 s
ASP	0.0025	5.45 s	0.005	2.69 s	0.0005	385.55 s	0.00025	0.34 h
speedup	–	301x	–	6023x	–	44x	–	283x

We quantify the accuracy of our proposed algorithms in terms of two metrics that measure the structure similarity and the attribute similarity. The first metric is the *Density* of the induced subgraph of the top-k result. The *Density* of a graph is the number of edges existing in the graph divided by the maximum possible number of edges in the graph (Meng & Shen, 2018). The greater *Density* implies the better structure similarity performance. The second metric is *Common Attribute score* (*CA*) of the result set, which is defined as $CA = \sum_{u,v \in S} \frac{|\mathcal{A}(u) \cap \mathcal{A}(v)|}{|\mathcal{A}(u) \cup \mathcal{A}(v)|}$, where $\mathcal{A}(u)$ represents the attribute set of node u. *CA* evaluates the attribute similarity of all pair nodes in result set.

5.5. Evaluation results

5.5.1. Convergence

To answer research question **RQ1**, we discuss how the error bound affects the convergence performance of our algorithms. Fig. 3 shows performance comparisons on *RR score* by different error bound ϵ. A higher *RR score* indicates better convergence performance. Here we set $|\mathcal{L}| = 100$ and $|l_i| = 100$ for all the algorithms. We can clearly see from Fig. 3 that both **SP** and **ASP** can consistently achieve a better convergence performance than Panther, and **SP** performs the best. Since the number of the sampling paths r is determined by the error bound ϵ when we keep other parameters fixed, which means that with the same robust search performance **SP** and **ASP** need fewer path samples.

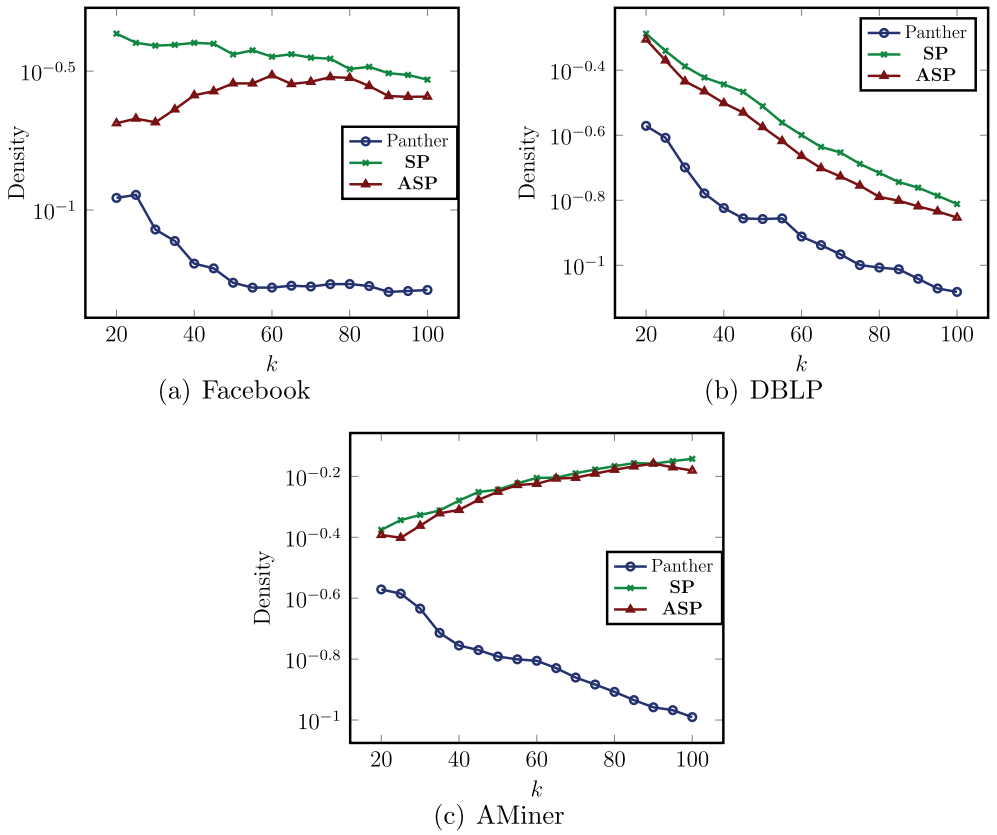


Fig. 5. Comparison of *density* score.

5.5.2. Scalability

Next, we turn to answer research question **RQ2**. Fig. 4(a–c) shows the running time of the three comparison methods in DBLP, AMiner, and Google+ networks by different error bound ϵ, and Fig. 4(d) shows their efficiency performance (CPU time) by different edge sizes in the synthetic networks. Clearly, our methods (both **SP** and **ASP**) are much faster than Panther in these experiments and **SP** achieves the best scalability performance.

Yet we have showed in Fig. 3 that the error bound ϵ of the three algorithms are different within the same robustness performance. To answer research question **RQ3**, we tune the error bound ϵ and record the running time when their *RR* scores approximately equal to 0.82, in which value the difference of top-100 result is quite modest. Table 2 lists the running time of comparison methods and the best performance is underlined. We can observe that our methods significantly lower the running time when their result are convergent, comparing with Panther. In DBLP network, **SP** achieves a 6000x speedup, and **ASP** achieves a 3400x speedup, compared with Panther. Even in the worst case of AMiner network, **ASP** also achieves a 25x speedup compared with Panther. The speed-up is moderate when the edge of the network is sparse, because all the three algorithms are more difficult to convergence in a sparse network.

5.5.3. Accuracy

Then, we turn to research questions **RQ4**. Fig. 5 shows the density result over the result size k within the same error bound ϵ. We can observe that both **SP** and **ASP** significantly outperform Panther on density metric, and **SP** scores the best in most case. It means that the single-source path sampling method can improve accuracy in estimating structure similarity.

Fig. 6 shows the *CA* score over the result number k. We can see that **ASP** significantly outperforms Panther and **SP**, and Panther has the worst *CA* performance. It means that **ASP** can improve accuracy in capturing the attribute similarity, and combining attributes into the sampled path certainly enhances similarity search performance. Combining with the experimental results on *density* and *CA* score, we can conclude that our **ASP** can effectively estimate similarity of nodes both on structure similarity and attribute similarity with a higher accuracy.

5.5.4. Performance on dynamic networks

Finally, we turn to answer **RQ5**. In order to show the performance of our methods in processing dynamic updates in large dynamic attributed networks, we conduct the following experiments in Aminer dataset. In addition to the initial similarity calculation (INI), we set four types of modifications in our dataset: adding a batch of structure edges (ASE), pruning a batch of structure edges (PSE),

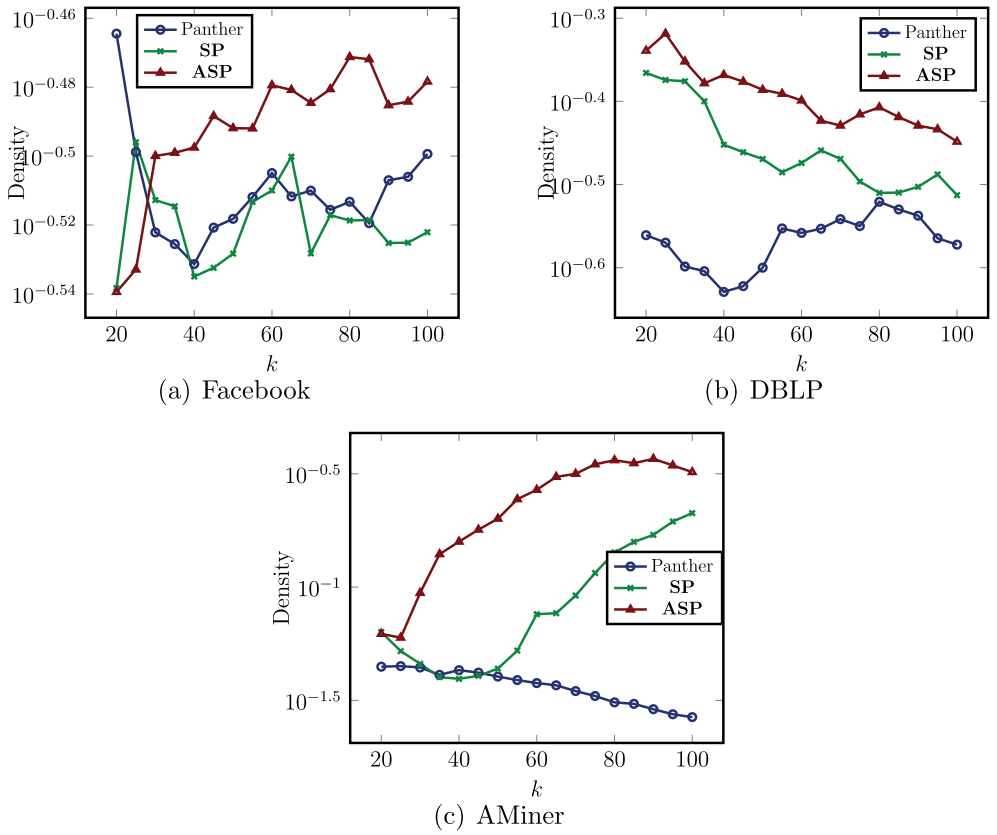


Fig. 6. Comparison on *CA* score.

Table 3
Time efficiency of incremental calculation.

Method	Panther	SP	ASP
INI	656.91 s	28.58 s	52.41 s
ASE	9.48 s	1.60 s	1.72 s
PSE	2.15 s	0.91 s	1.19 s
AAE	–	–	1.32 s
PAE	–	–	1.10 s

adding a batch of attribute edges (AAE), and pruning a batch of attribute edges (PAE). In each kind of modification, we run our algorithms 100 times by randomly selecting a batch of 200 edges, and obtain the average running time. Here we set $\epsilon = 0.001$. Table 3 shows the running time of the three methods. We can clearly see that the three algorithms all obtain an enormous speedup comparing with the initial similarity calculation, and a batch of edge modifications or attribute perturbations can be updated in about 1 second by using **ASP**. Therefore, our proposed **ASP** can efficiently maintain similarities in large dynamic attributed networks.

6. Conclusions

In this paper, we developed an efficient algorithm for approximately estimating top-k similar nodes for a single given query node in large dynamic attributed networks. Our algorithm combines structure similarity and attribute similarity into a unified similarity metric by constructing the attribute augmented network, and then samples random paths among nodes and attributes to approximately estimate the path similarity scores. It also contains two dynamic updating schemes for maintaining similarity scores in response to dynamic edge insertions and deletions. Our empirical evaluations show that the single-source path sampling method used in our algorithms has less running time, better convergence performance and accuracy than the path sampling method in Panther, and our dynamic updating schemes can efficiently maintain similarity scores for large dynamic attributed networks. The experimental results also show that the extra attribute information from a unified similarity metric significantly improves the effectiveness of similarity search.

For the future work, we plan to incorporate other sources of information into the similarity metric, such as edge attributes, node influence or information diffusion.

Acknowledgements

This work is supported by National Key R & D Program of China Project #2017YFB0203201 and Australian Research Council Discovery Project DP150104871. The corresponding author is Hong Shen.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.ipm.2019.102074 .

References

Bahmani, B., Chowdhury, A., & Goel, A. (2010). Fast incremental and personalized pagerank. *Proceedings of the VLDB Endowment, 4*(3), 173–184.
Celik, M., & Dokuz, A. S. (2018). Discovering socially similar users in social media datasets based on their socially important locations. *Information Processing & Management, 54*(6), 1154–1168.
Cummins, R. (2014). Document score distribution models for query performance inference and prediction. *ACM Transactions on Information Systems (TOIS), 32*(1), 2.
Fujiwara, Y., Nakatsuji, M., Onizuka, M., & Kitsuregawa, M. (2012). Fast and exact top-k search for random walk with restart. *Proceedings of the VLDB Endowment, 5*(5), 442–453.
Gong, N. Z., Talwalkar, A., Mackey, L., Huang, L., Shin, E. C. R., Stefanov, E., ... Song, D. (2014). Joint link prediction and attribute inference using a social-attribute network. *ACM Transactions on Intelligent Systems and Technology (TIST), 5*(2), 27.
Har-Peled, S., & Sharir, M. (2011). Relative (p, ε)-approximations in geometry. *Discrete & Computational Geometry, 45*(3), 462–496.
Haveliwala, T. H. (2002). *Topic-sensitive pagerank. Proceedings of the 11th international conference on world wide web*. ACM517–526.
Huang, X., Cheng, H., & Yu, J. X. (2015). Dense community detection in multi-valued attributed networks. *Information Sciences, 314*, 77–99.
Jeh, G., & Widom, J. (2002). *Simrank: a measure of structural-context similarity. Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining*. ACM538–543.
Jiang, M., Fu, A. W. C., & Wong, R. C. W. (2017). Reads: a random walk approach for efficient and accurate dynamic simrank. *Proceedings of the VLDB Endowment, 10*(9), 937–948.
Kusumoto, M., Maehara, T., & Kawarabayashi, K.i. (2014). *Scalable similarity search for simrank. Proceedings of the 2014 ACM SIGMOD international conference on management of data*. ACM325–336.
Lee, P., Lakshmanan, L. V., & Yu, J. X. (2012). *On top-k structural similarity search. 2012 IEEE 28th international conference on data engineering*. IEEE774–785.
Li, C., Bai, J., Wenjun, Z., & Xihao, Y. (2019). Community detection using hierarchical clustering based on edge-weighted similarity in cloud environment. *Information Processing & Management, 56*(1), 91–109.
Li, C., Han, J., He, G., Jin, X., Sun, Y., Yu, Y., & Wu, T. (2010). *Fast computation of simrank for static and dynamic information networks. Proceedings of the 13th international conference on extending database technology*. ACM465–476.
Lizorkin, D., Velikhov, P., Grinev, M., & Turdakov, D. (2008). Accuracy estimate and optimization techniques for simrank computation. *Proceedings of the VLDB Endowment, 1*(1), 422–433.
Lu, J., Gong, Z., & Lin, X. (2017). A novel and fast simrank algorithm. *IEEE Transactions on Knowledge and Data Engineering, 29*(3), 572–585.
Martínez, V., Berzal, F., & Cubero, J. C. (2017). A survey of link prediction in complex networks. *ACM Computing Surveys (CSUR), 49*(4), 69.
Meng, Z., & Shen, H. (2018). Dissimilarity-constrained node attribute coverage diversification for novelty-enhanced top-k search in large attributed networks. *Knowledge-Based Systems, 150*, 85–94.
Meng, Z., Shen, H., Huang, H., Liu, W., Wang, J., & Sangaiah, A. K. (2018). Search result diversification on attributed networks via nonnegative matrix factorization. *Information Processing & Management, 54*(6), 1277–1291.
Riondato, M., & Kornaropoulos, E. M. (2016). Fast approximation of betweenness centrality through sampling. *Data Mining and Knowledge Discovery, 30*(2), 438–475.
Song, J., Luo, X., Gao, J., Zhou, C., Wei, H., & Yu, J. X. (2018). Uniwalk: Unidirectional random walk based scalable simrank computation over large graph. *IEEE Transactions on Knowledge and Data Engineering, 30*(5), 992–1006.
Tong, H., Faloutsos, C., & Pan, J. Y. (2006). *Fast random walk with restart and its applications. Sixth international conference on data mining (ICDM'06)*. IEEE613–622.
Vapnik, V. N., & Chervonenkis, A. Y. (2015). *On the uniform convergence of relative frequencies of events to their probabilities. Measures of complexity*. Springer11–30.
Wang, Y., Chen, L., Che, Y., & Luo, Q. (2018). Accelerating pairwise simrank estimation over static and dynamic graphs. *The VLDB Journal*, 1–24.
Wang, Y., Lian, X., & Chen, L. (2018). *Efficient simrank tracking in dynamic graphs. 2018 IEEE 34th international conference on data engineering (ICDE)*. IEEE545–556.
Wu, Y., Jin, R., & Zhang, X. (2014). *Fast and unified local search for random walk based k-nearest-neighbor query in large graphs. Proceedings of the 2014 ACM SIGMOD international conference on management of data*. ACM1139–1150.
Yu, W., & McCann, J. A. (McCann, 2015a). Efficient partial-pairs simrank search on large networks. *Proceedings of the VLDB Endowment, 8*(5), 569–580.
Yu, W., & McCann, J. A. (McCann, 2015b). *High quality graph-based similarity search. Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. ACM83–92.
Zhang, C., Shou, L., Chen, K., Chen, G., & Bei, Y. (2012). *Evaluating geo-social influence in location-based social networks. Proceedings of the 21st ACM international conference on information and knowledge management*. ACM1442–1451.
Zhang, J., Tang, J., Ma, C., Tong, H., Jing, Y., & Li, J. (2015). *Panther: Fast top-k similarity search on large networks. Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM1445–1454.
Zhang, J., Tang, J., Ma, C., Tong, H., Jing, Y., Li, J., ... Moens, M. F. (2017). Fast and flexible top-k similarity search on large networks. *ACM Transactions on Information Systems (TOIS), 36*(2), 13.
Zhang, Z., Shao, Y., Cui, B., & Zhang, C. (2017). An experimental evaluation of simrank-based similarity search algorithms. *Proceedings of the VLDB Endowment, 10*(5), 601–612.
Zhao, P., Han, J., & Sun, Y. (2009). *P-rank: a comprehensive structural similarity measure over information networks. Proceedings of the 18th ACM conference on information and knowledge management*. ACM553–562.
Zhou, Y., Cheng, H., & Yu, J. X. (2009). Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment, 2*(1), 718–729.
Zhu, F., Fang, Y., Chang, K. C. C., & Ying, J. (2013). Incremental and accuracy-aware personalized pagerank through scheduled approximation. *Proceedings of the VLDB Endowment, 6*(6), 481–492.