Expert Systems with Applications 42 (2015) 699–712

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Top-k similarity search in heterogeneous information networks with x-star network schema

Mingxi Zhang [a,b,*], Hao Hu [b], Zhenying He [b], Wei Wang [b]

[a] College of Communication and Art Design, University of Shanghai for Science and Technology, 516 Jungong Road, Shanghai 200093, China
[b] School of Computer Science, Fudan University, 825 Zhangheng Road, Shanghai 201203, China

ARTICLE INFO

Article history:
Available online 10 September 2014

Keywords:
Similarity search
Information network
x-star network schema

ABSTRACT

An x-star network is an information network which consists of centers with connections among themselves, and different type attributes linking to these centers. As x-star networks become ubiquitous, extracting knowledge from x-star networks has become an important task. Similarity search in x-star network aims to find the centers similar to a given query center, which has numerous applications including collaborative filtering, community mining and web search. Although existing methods yield promising similar results, such as SimRank and P-Rank, they are not applicable for massive x-star networks. In this paper, we propose a structural-based similarity measure, NetSim, towards efficiently computing similarity between centers in an x-star network. The similarity between attributes is computed in the pre-processing stage by the expected meeting probability over attribute network that is extracted from the whole structure of x-star network. The similarity between centers is computed online according to the attribute similarities based on the intuition that similar centers are linked with similar attributes. NetSim requires less time and space cost than existing methods since the scale of attribute network is significantly smaller than the whole x-star network. For supporting fast online query processing, we develop a pruning algorithm by building a pruning index, which prunes candidate centers that are not promising. Extensive experiments demonstrate the effectiveness and efficiency of our method through comparing with the state-of-the-art measures.

1. Introduction

Information networks are becoming ubiquitous and form the critical components of the information infrastructure. A network with x-star network schema (called x-star network) is an information network which consists of centers with connections among themselves, and different type attributes linking to these centers. X-star network is the extension of star network (Sun, Yu, & Han, 2009) with considering the connections among centers. Typical centers are products, papers, people and the like. There are large amount of examples of x-star networks, for example, bibliographic network which consists of papers with citations, and each paper *contains* several terms, *is published in* one venue, *is written by* several authors; product co-purchasing network among products with description that a product *belongs to* several categories, *contains* several terms; social network among users with personal information that consists of interest, department and profession. With x-star networks becoming diverse and complex, there is a need for designing algorithmic tools and developing applications to exploit the underlying structure in the data.

Similarity search focuses on finding the most similar objects to a given object. For the x-star network, we are particularly interested in providing a similarity search functions for searching the similar objects of center type. For example, in a bibliographic network, a user may be interested in the most similar paper for a given paper; in the product co-purchasing network, a user may be interested in searching for the most similar products for a given product.

As one of the most important aspects in information network analysis, similarity search in x-star network has numerous real applications. For example, recommender system over product co-purchasing network, which recommends the most similar products to the product chosen by a user; purchasing relationship prediction, which predicts each purchasing relationship as the link between a user and a product similar to the chosen product. These applications usually require an effective and trustworthy similarity

* Corresponding author at: College of Communication and Art Design, University of Shanghai for Science and Technology, 516 Jungong Road, Shanghai 200093, China.

E-mail addresses: mingxizhang10@fudan.edu.cn (M. Zhang), huhao@fudan.edu.cn (H. Hu), zhenying@fudan.edu.cn (Z. He), weiwang1@fudan.edu.cn (W. Wang).

http://dx.doi.org/10.1016/j.eswa.2014.08.039
0957-4174/© 2014 Elsevier Ltd. All rights reserved.

search function to answer the question "Which centers are most similar to this one?".

For satisfying the above requirement, a great number of similarity measures are devoted (Ganesan, Garcia-Molina, & Widom, 2003; Jeh & Widom, 2002; Lin, King, & Lyu, 2006; Xi et al., 2005; Zhao, Han, & Sun, 2009; Zhou, Cheng, & Yu, 2009), which can be divided into two broad categories: (1) content-based similarity measures treat each object as a bag of items or as a vector of word weights (Ganesan et al., 2003); and (2) structural-based similarity measures, consider object-to-object relationships expressed in terms of links (Jeh & Widom, 2002; Lin et al., 2006; Xi et al., 2005; Zhao et al., 2009; Zhou et al., 2009). Comparative studies on different similarity measures demonstrate that structural-based similarity measures produce systematically better correlation with human judgements compared to the content-based ones (Maguitman, Menczer, Erdinc, Roinestad, & Vespignani, 2006). From this perspective, it is reasonable to assume that structural-based similarity measure is worth thoroughly exploring for tackling similarity search problem in x-star networks.

Many structural-based similarity measures can be found in previous work, such as SimRank (Jeh & Widom, 2002), P-Rank (Zhao et al., 2009) and SimFusion (Xi et al., 2005). These methods compute similarities by utilizing the global information of information networks. The similarity between two objects is usually defined recursively with respect to a "random surfer" model and is under the iterative computation framework, based on this, the indirect links between objects are explored for discovering the underlying relationships. These methods are applicable to any domain with object-to-object relationships and can provide a good way for effectively measuring object similarities in information networks.

Unfortunately, although existing methods yield promising similar results, they are not applicable for large networks due to their high time and space complexity. When applying these methods to x-star networks, large similarity matrix need to be maintained for storing similarities among different type objects, the matrix would become full after just a few iterations, and then the storage and efficiency problems would be run into with x-star networks becoming massive. For optimizing similarity computation, some optimization techniques have been proposed (Li et al., 2010a; Lizorkin, Velikhov, Grinev, & Turdakov, 2008, 2010; Yu, Lin, Zhang, Chang, & Pei, 2013; Zhao, Xiao, Lin, Liu, & Zhang, 2013). However, these optimization techniques are particularly inefficient in practice since the whole similarity matrix among different type objects need to be maintained as well.

The above illustrates the significance of similarity search and discusses the drawback of existing methods, which motivates our research. In this paper, we study the top-*k* similarity search problem in x-star networks, which aims to reduce the time and space cost of similarity search. For tackling this problem, two main challenges need to be overcome. First, although it is critical to reduce the cost in pre-processing stage, the high time and space complexity of similarity computation would become an obstacle for evaluating center similarities. Second, the process of on-line query processing without pre-computing the whole similarity matrix involves expensive operations to calculate the similarity between the query and each candidate (Lee, Lakshmanan, & Yu, 2012; Li, Liu, Yu, He, & Du, 2010b), which would increase the response time significantly.

Our contributions are based on these challenges. For improving the efficiency of similarity computation, we propose a structural-based similarity measure NetSim based on the intuition that "similar centers are usually linked with similar attributes". We compute the similarity between centers according to attribute similarities which are computed off-line over each attribute network. The attribute network is built according to global structure information of x-star network, and each attribute network contains only the attributes of one attribute type, which is in fact a small portion of the whole x-star network, hence the total similarity computation over different type attribute networks would be evidently decreased. The global structure is utilized by NetSim as well since the global structure information is integrated into attribute networks.

For supporting fast online query processing, we also develop a pruning algorithm NetSim-pruning by building a pruning index. The candidates that are not promising are pruned by setting the user-controlled thresholds. Based on the pruning index, the execution time of query processing is significantly decreased without pre-computing the whole similarity matrix among all the objects. Extensive experiments on real datasets demonstrate the effectiveness and efficiency of our approach through comparing with the state-of-the-art methods.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 gives the notations and problem definition. In Section 4, relations and links are weighed. Section 5 builds the attribute network and computes attribute similarity. Section 6 defines NetSim similarity, proposes NetSim-baseline and NetSim-pruning. Experimental studies are reported in Section 7. Section 8 concludes this paper and discusses the future work.

2. Related work

Similarity measures are common standards for measuring the closeness of different entities, which are crucial and required by considerable amount of applications include clustering, recommender system, proximity query processing, community detection and other research fields. There are considerable amount of literature on structure-based similarity measures in the previous work.

Co-citation (Small, 1973) and bibliographic coupling (Kessler, 1963) are two early noteworthy methods from bibliometrics fields. Co-citation scheme measures similarity between two papers *p* and *q* based on the number of papers which cite both *p* and *q*, while Bibliographic coupling measures similarity based on the number of papers cited by both *p* and *q*. Amsler (1972) proposed another approach which fuses both Co-citation and bibliographic coupling for similarity computation by considering both in- and out-links, and the relative weight of in-link and out-link directions by tuning a factor $\alpha \in [0, 1]$. These approaches are efficient for similarity computation due to their simple computation procedure, by which the sparsity of real networks can be utilized for reducing computation cost. However, these approaches use only the local structure information for computing similarities, the valuable underlying relationships can not be considered, which would neglect much latent similar objects due to the sparseness of real networks.

Computing similarity recursively based on structure has also been explored in several existing methods, where the global structure information of information networks is utilized for computing similarity. SimRank is one of the most renowned structural-based similarity measures, invented by Jeh and Widom (2002), which has been widely used in various fields (Fogaras & Rácz, 2005; Pan, Yang, Faloutsos, & Duygulu, 2004; Yin, Han, & Yu, 2006). The similarity between objects is defined as the expected distance for two random surfers when they walk along the network backwards, based on the intuition that "objects are similar if they are referenced by similar objects". As mentioned by Jeh and Widom (2002), the weakness of SimRank is the limited information problem since it uses only in-link directions only for similarity computation, and hence the similarity conveyed from out-links directions would be neglected. To overcome the limited information problem of SimRank, Zhao et al. (2009) proposed P-Rank which enriches SimRank by jointly encoding both in- and out-link relationships into structural similarity computation. The intuition behind P-Rank

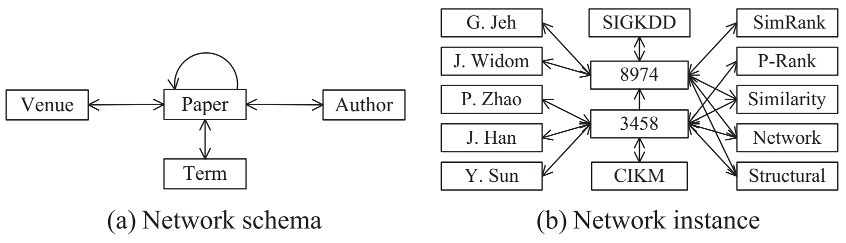


Fig. 1. Bibliographic information network.

is that "two objects are similar if (1) they are referenced by similar objects; and (2) they reference similar objects". P-Rank solves the limited information problem effectively, and achieves much better results. In fact, Co-citation, Bibliographic coupling, Amsler and SimRank can be considered as special cases of P-Rank as mentioned by Zhao et al. (2009). PageSim (Lin et al., 2006) is a link-based similarity based on PageRank score propagation through link paths, which is capable of measuring similarity between web pages. Contrast to SimRank and P-Rank, PageSim can measure similarity between any two web pages at the case of meeting when walking along the paths of different length, whereas SimRank cannot, since SimRank considers only the meetings of equal path length at each iterations.

SimFusion (Xi et al., 2005) uses Unified Relationship Matrix (URM) to represent the heterogeneous objects and the interrelationships among these web objects, where the relation importance is defined manually. The similarity matrix is computed iteratively over URM, which helps overcome the data sparseness problem and detect the latent relationships among heterogeneous data objects. In fact, SimFusion can be transformed into SimRank easily with minor modification, which is also a special case of P-Rank (Cai, Zhang, Ding, & Chakravarthy, 2010). Compared to SimRank and P-Rank, SimFusion addresses the heterogeneity of information networks by effectively combining relationships from multiple heterogeneous data sources. SA (Cheng, Zhou, & Yu, 2011; Zhou et al., 2009) is a random walk-based method for clustering over x-star networks by considering graph topological structure and the objects properties. Compared to SimFusion, the weights of different type links are defined by proposing iterative clustering process.

A problem of the above similarity measures is that, due to the high computational complexity, high space and time cost would be required. When computing similarities, the matrices among different type objects need to be maintained, and the efficiency problem would be run into with network becoming massive. As a result, the high space and time complexity for computing similarities among different type objects would become an obstacle for applying these measures to large information networks.

Different from the existing similarity measures, NetSim computes only the attribute similarity matrix for each attribute type in pre-processing stage, the whole matrix among different type objects is not maintained. NetSim pre-compute only the similarity matrix for each attribute type, and only t small matrices need to be maintained, where t is the number of attribute type in a given x-star network. Besides, the global structure information of the whole x-star networks can be utilized as well, since the global information of x-star networks is integrated while building the attribute network.

Some optimization techniques on SimRank have been developed recently. Lizorkin et al. (2008, 2010) optimized SimRank by essential node pairs, partial sums and threshold-sieved. Li et al. (2010a) introduced a no-iterative SimRank computation method in dynamic networks, which rewrite SimRank to into a non-iterative form based on the Kronecker product and vectorization operators. Zhao et al. (2013) proposed partition-based approach to tackle the efficiency problem of SimRank, by dividing the data graphs into variable-size non-overlapping partitions. Yu et al. (2013) proposed a revised version of SimRank which resolves the counter-intuitive zero-similarity issues while inheriting merits of the basic SimRank philosophy, and leveraged a novel clustering strategy for optimizing the SimRank computation.

Nevertheless, these optimization techniques still require huge matrix to maintain the whole similarities among different type objects, which are particularly inefficient in practice and can not be applied to large x-star networks. Compared to these optimization techniques, NetSim is in fact a general similarity search framework. NetSim is not conflict with the existing optimization techniques, since these optimization techniques can be easily integrated into NetSim for speeding up similarity computation over attribute networks (as shown in Section 7).

3. Notations and problem definition

3.1. Notations

We first introduce the definitions of information network and network schema, which are defined by Sun et al. (2009) and Sun, Han, Yan, Yu, and Wu (2011).

Definition 1 (*Information network*). An information network is defined as a directed graph $G = (V, E, W)$, with an object type mapping function $\Phi : V \to \Lambda$ and a link type mapping function $\Psi : E \to R$. An object $v \in V$ belongs to one particular object type, i.e., $\Phi(v) \in \Lambda$, and a link $e \in E$ belongs to particular relation, i.e., $\Psi(e) \in R$. The weight of link $e(u, v) \in E$ is denoted as $w(u, v) \in W$.

The object set of X_i type is denoted by V_{X_i}. The in- and out-neighbor sets of type X_i of object v are denoted by $I_i(v)$ and $O_i(v)$, respectively. Information networks are divided into two types: (1) **heterogeneous information network** if $|\Lambda| > 1$ or $|R| > 1$; and (2) **homogeneous information network** for otherwise. The relation from object type X_i to X_j is denoted as X_iX_j. The link set of X_iX_j type is denoted by $E_{X_iX_j}$.

Definition 2 (*Network schema*). The network schema is a template for information network $G = (V, E, W)$, with an object type mapping function $\Phi : V \to \Lambda$ and a link type mapping function $\Psi : E \to R$, which is a directed graph over object types Λ, denoted as $S_G = (\Lambda, R)$.

A network schema is a template for instancing a information network, which determines the characteristics of network structure. Next we give the definition of x-star network schema by extending star network schema (Sun et al., 2009).

Definition 3 (*X-star network schema*). The x-star network schema is a template for x-star network $G = (V, E, W)$ with $t + 1$ object types, which is defined as $S_G = (\Lambda, R)$, where $\Lambda = \cup_{i=0}^{t}\{X_i\}$, $R = \{X_0X_0\} \cup_{i=1}^{t} \{X_0X_i, X_iX_0\}$. X_0 and $X_i (i > 0)$ are center type and attribute type respectively.

Objects of X_0 type and X_i type are center type objects (also called center) and attribute type objects (also called attribute) respectively. For symbol X_i, we hold $i > 0$ if there is no special note. Relation X_0X_0 is a special relation that contains two semantics, for example, relation between papers may be *cite* or *cited by* relation, for simplicity, we consider only one type of them. Fig. 1(a) shows the schema of bibliographic network, which forms an x-star network schema, where “Paper” is center type which is denoted by X_P, “Author”, “Term” and “Venue” are attribute types, which are denoted by X_A, X_T and X_V respectively. The object type set is denoted by $\{X_P, X_A, X_T, X_V\}$, and the relation set is denoted by $\{X_PX_P, X_PX_A, X_PX_V, X_PX_T, X_AX_P, X_VX_P, X_TX_P\}$. The semantic of relation X_PX_P is *cite*, X_PX_A is *written by* while X_AX_P is *write*, X_PX_V is *published in* while X_VX_P is *publish*, and other relations can be explained similarly. One instance of this schema is shown in Fig. 1(b), where “8974” and “3458” represents two papers, which are center type objects, other objects are attributes of different types, such as “SimRank” and “Similarity” are X_T type objects, “J. Widom” and “P. Zhao” are X_A type objects, and “SIGKDD” and “CIKM” are X_V type objects. There are different type links among these different type objects, such as the link of X_PX_T type between “8974” and “SimRank”, the link of X_PX_A type between “3458” and “J. Widom”.

Definition 4 (*Attribute network*). An attribute network for $X_i(i > 0)$ type is defined as a directed graph $G_i = (V_{G_i}, E_{G_i}, W_{G_i})$ with network schema $S_{G_i} = (\Lambda, R)$, where $\Lambda = \{X_i\}$ and $R = \{X_iX_i\}$.

An attribute network consists of attributes of one type and the links among themselves. The algorithm for building attribute network is given in Section 5.

3.2. Problem definition

Under the definition of NetSim (defined in subSection 6.1), we formally define our top-k similarity search problem, which is described as follows.

Definition 5 (*Top-k similarity search under NetSim*). Given an x-star network $G = (V, E, W)$, the top-k similarity search for a given center u is to find k most similar centers ranking with similarities descending, such that $S^l(u, v) \geqslant S^l(u, v')$ for v in the returning list and v' not, where S^l is the similarity function under NetSim of path length l.

4. Defining weights of links

Roughly, the weight between two objects is defined as the link frequency between them, however, the relation importance may be very different since the relations are diverse and independent in heterogeneous network. Intuitively, in relation X_iX_j, if the size of set V_{X_j} is bigger, then the objects of X_j type are more informative for the objects of X_i type, and vice versa. For example, in relation $R_{published\ in}$: *paper* $\rightarrow$ *venue*, the venues are not so informative for papers since one venue usually publishes many papers from many fields, such as *EDBT* usually receives papers from more than 30 different research topics, and there is a fact that the number of the venues is usually small; and in relation $R_{written\ by}$: *paper* $\rightarrow$ *author*, the authors are more informative than venues since the research interest of one author is not too broad, such as we found that only 6 research topics are shown in *Jiawei Han*’s home page, and there is a fact that the number of the authors is bigger than venues. Formally, the importance of relation X_iX_j is defined as the probability for generating an object of X_i type over all the objects of different types, which is formalized as:

$$rw(X_iX_j) = \frac{|V_{X_j}|}{\sum_{X_k \in \Lambda} \sum_{X_iX_k \in R} |V_{X_k}|} \tag{1}$$

This formula can applied to any type network schema besides x-star schema for measuring relation importance. In this paper, we focus on only the x-star network schema.

For link $e(u, v)$ of X_iX_j type, if u has more out-neighbors of X_j type, then link $e(u, v)$ is weaker, and vice versa. For example, in relation $R_{contained\ in}$: *term* $\rightarrow$ *paper*, the link from a frequently used term (such as *data*) to a paper is weaker than from a rarely used term (such as *P-Rank*), since rarely used term is more informative than the frequently used. Formally, the weight of $e(u, v) \in E_{X_iX_j}$ is defined as:

$$w(u, v) = rw(X_iX_j) \frac{N(u, v)}{\sum_{x \in O_j(u)} N(u, x)} \tag{2}$$

where $N(u, v)$ is the frequency of u that links to v. The idea of penalizing the node with high degree in Eq. (2) is similar with that of setting a universal sink node (Sarkar & Moore, 2010; Satuluri & Parthasarathy, 2011), these methods are designed for homogenous networks, and our proposed method can be used for defining the weights in heterogenous networks.

5. Building attribute network and computing attribute similarity

5.1. Building attribute network.

Algorithm 1. Building attribute network

```
Input:
    X-star network $G(V, E, W)$;
Output:
    Attribute network $G_i(V_{G_i}, E_{G_i}, W_{G_i})$;
1: Initialize $E_{G_i}$, $W_{G_i}$ as $\emptyset$, and $V_{G_i}$ be $V_{X_i}$;
2: For $u \in V_{X_0}$ do
3:    For $v \in O_0(u)$ do
4:       For $(a, b) \in I_i(u) \times O_i(v)$ do
5:          if $e(a, b) \in E_{G_i}$ then
6:             $w(a, b) \leftarrow w(a, b) + w(a, u)w(u, v)w(v, b)$;
7:          else
8:             $E_{G_i} \leftarrow E_{G_i} \cup e(a, b)$;
9:             $w(a, b) \leftarrow w(a, u)w(u, v)w(v, b)$;
10:            $W_{G_i} \leftarrow W_{G_i} \cup w(a, b)$;
11:         end if
12:      end for
13:   end for
14:   For $(a, b) \in I_i(u) \times O_i(u)$ do
15:      if $e \in E_{G_i}$ then
16:         $w(a, b) \leftarrow w(a, b) + w(a, u)w(u, b)$;
17:      else
18:         $E_{G_i} \leftarrow E_{G_i} \cup e(a, b)$;
19:         $w(a, b) \leftarrow w(a, u)w(u, b)$;
20:         $W_{G_i} \leftarrow W_{G_i} \cup w(a, b)$;
21:      end if
22:   end for
23: end for
```

We build the attribute network by extracting the relationship among attributes from the whole x-star network, where different relationships among centers and attributes are considered. The weight of each link means the transition probability from one attribute to another through the paths over x-star network. Like the random surfer model, high transition probability implicates that a surfer starts from one attribute can easily arrive at another one. For modeling such transition probability, the links among centers and the co-links among attributes are considered. Formally, when building attribute network $G_i(V_{G_i}, E_{G_i}, W_{G_i})$ for X_i type, we rashly assume that the paths from one object to anther are always independent, and define the weight of $e(a,b) \in E_{G_i}$ as the transition probability that a surfer starts at $a \in V_{X_i}$ and ends at $b \in V_{X_i}$ through the instances of combined relation $R_{X_i \to X_0 \to X_i}$ or $R_{X_i \to X_0 \to X_0 \to X_i}$. The weight of $e(a,b)$ is defined as:

$$w(a,b) = \sum_{u \in O_0(a) \cap I_0(b)} w(a,u)w(u,b) + \sum_{(u,v) \in O_0(a) \times I_0(b)} w(a,u)w(u,v)w(v,b) \tag{3}$$

Algorithm 1 shows the pseudo-code for building attribute network. From line 3 to 13, the weight between two attributes is modified according to links of X_0X_0 relation between centers they are linking to; and from line 14 to 22, the weight is modified by considering the co-links of them. Specifically, given an x-star network G, the attribute network G_i of X_i type is built as: for each $u \in V_{X_0}$: (1) we get each $v \in O_0(u)$, and for each $(a,b) \in I_i(u) \times O_i(v)$, if $e \in E_{G_i}$ we update $w(a,b)$ by accumulating $w(a,u)w(u,v)w(v,b)$, otherwise we build a new edge $e(a,b)$ and assign $w(a,u)w(u,v)$ $w(v,b)$ to its weight $w(a,b)$; (2) for each $(a,b) \in I_i(u) \times O_i(u)$, if $e \in E_{G_i}$, we update $w(a,b)$ by accumulating $w(a,u)w(u,b)$, otherwise we build a new edge $e(a,b)$ and assign $w(a,u)w(u,b)$ to its weight $w(a,b)$. The time cost of this algorithm is $O(|V_{X_0}|(d_{O_0}d_{I_i}d_{O_i} + d_{I_i}d_{O_i}))$, where d_{O_i} and d_{I_i} are the average out- and in-degree correspond to X_i attribute type respectively. In the worst case, each object points to all other objects of the network, the time cost is $O(|V_{X_0}|^2|V_{X_i}|^2)$. Many real networks obey a power-law degree distribution (Faloutsos, Faloutsos, & Faloutsos, 1999), in this case, the average cost is $O(|V_{X_0}|)$.

5.2. Computing attribute similarity

Similarities among attributes can be computed by any existing methods on homogeneous networks. In this paper, we compute attribute similarity based on SimRank, where the transition probability is defined by considering the weights of links. The transition probability from a to b over G_i is formalized as $P_i(a,b) = \frac{w(a,b)}{\sum_{x \in O_{G_i}(a)} w(a,x)}$ if $e(a,b) \in E_{G_i}$, otherwise $P_i(a,b) = 0$. $M_i^l(a,b)$ denotes the similarity between a and b. For $l = 0$, we define $M_i^0(a,b) = 1$ if $a = b$, and otherwise $M_{i,0}(a,b) = 0$. For $l \neq 0$, we define $M_i^l(a,b) = 1$ if $a = b$, otherwise:

$$M_i^l(a,b) = c \sum_{x \in O_{G_i}(a)} \sum_{y \in O_{G_i}(b)} P_i(a,x)P_i(b,y)M_i^{l-1}(x,y) \tag{4}$$

where $c \in (0,1)$ is the decay factor. For attribute type X_i, the time cost for computing attribute similarity matrices of different types is derived as $O(ld_{G_i}^2|V_{X_i}|^2)$, where d_{G_i} is the average degree in G_i.

Lemma 1. *Given attributes $a, b \in V_{X_i}$, decay factor $c \in (0,1)$ and path length l, we hold that $M_i^l(a,b) = M_i^l(b,a)$.*

From Eq. (4) and the property of SimRank (Jeh & Widom, 2002), we can derive $M_i^l(a,b) = M_i^l(b,a)$.

Lemma 2. *Given attributes $a, b \in V_{X_i}$, decay factor $c \in (0,1)$ and path length l, we hold that $0 \leqslant M_i^l(a,b) \leqslant M_i^{l+1}(a,b) \leqslant 1$.*

Similarly, from Eq. (4) and the property of SimRank (Jeh & Widom, 2002), we can also derive $0 \leqslant M_i^l(a,b) \leqslant M_i^{l+1}(a,b) \leqslant 1$.

6. Query processing over X-star network

6.1. Center similarity under NetSim

Center similarity is computed based on the mentioned intuition that similar centers are linked with similar attributes. We firstly reformalize the link importance and the relation importance without considering relationship among centers, i.e., $w^*(u,x) = \frac{w(u,x)}{\sum_{p \in O_i(u)} w(u,p)}$, $rw^*(X_0X_i) = \frac{rw(X_0X_i)}{\sum_{j=1}^{t} rw(X_0X_j)}$. The similarity between u and v is defined as $S^l(u,v) = 1$ if $u = v$, otherwise:

$$S^l(u,v) = \sum_{i=1}^{t} rw^*(X_0X_i)S_i^l(u,v) \tag{5}$$

where

$$S_i^l(u,v) = \sum_{x \in O_i(u)} \sum_{y \in O_i(v)} w^*(u,x)w^*(v,y)M_i^l(x,y) \tag{6}$$

In Eq. (6), $S_i^l(u,v)$ is the similarity between u and v corresponds to X_i type attributes without considering other types. We use $rw^*(X_0X_i)$ in Eq. (5) to balance the similarity contributed by each X_i type, since different type attributes make very different contribution for computing center similarity. For example, venues (such as *EDBT*) usually make less contribution than authors (such as *Jiawei Han*) for computing paper similarity since the research topics of venues are more broader than authors.

Theorem 1. *Given centers $u, v \in V_{X_0}$, decay factor $c \in (0,1)$ and path length l, we hold that $S^l(u,v) = S^l(v,u)$.*

From Eq. (5) and Lemma 1, we can easily get $S^l(u,v) = S^l(v,u)$. Theorem 1 demonstrates the symmetry property of NetSim.

Theorem 2. *Given centers $u, v \in V_{X_0}$, decay factor $c \in (0,1)$ and path length l, we hold that $0 \leqslant S^l(u,v) \leqslant S^{l+1}(u,v) \leqslant 1$.*

Proof. According to Lemma 2, we get $0 \leqslant M_i^l(a,b) \leqslant M_i^{l+1}(a,b) \leqslant 1$, combine with Eq. (6), we can derive:

(1) $S_i^l(u,v) \geqslant 0$, by Eq. (5), we have $S^l(u,v) \geqslant 0$;
(2) $S_i^l(u,v) \leqslant \sum_{x \in O_i(u)} \sum_{y \in O_i(v)} w^*(u,x)w^*(v,y) = 1$, by Eq. (5), we have $S^l(u,v) = \sum_{i=1}^{t} rw^*(X_0X_i)S_i^l(u,v) \leqslant \sum_{i=1}^{t} rw^*(X_0X_i) = 1$, replace l by $l+1$, we get $S^{l+1}(u,v) \leqslant 1$;
(3) $S_i^{l+1}(u,v) - S_i^l(u,v) = \sum_{i=1}^{t} rw^*(X_0X_i)\left(S_i^{l+1}(u,v) - S_i^l(u,v)\right)$ and from $M_i^{l+1}(x,y) - M_i^{l+1}(x,y) \geqslant 0$, we get $S_i^{l+1}(u,v) - S_i^l(u,v) \geqslant 0$, then we have $S^{l+1}(u,v) - S^l(u,v) \geqslant 0$.

Thus, $0 \leqslant S^{l+1}(u,v) - S^l(u,v) \leqslant 1$. □

Theorem 2 shows the monotonicity property of NetSim that the iterative NetSim is non-decreasing with path length increasing.

6.2. NetSim-baseline

For a given query center, the straightforward baseline method for finding top-k similar centers is: firstly find the most k similar centers by computing similarity between query and each candidate using Eq. (5), then sort and return them. There are $|V_{X_0}|$ candidates need to be checked. The total time cost is $O\left(|V_{X_0}|\left(\sum_{i=1}^{t}|d_{O_i}|^2 + k\right) + \Gamma(k)\right)$, where $\Gamma(k)$ is the time cost for sorting k objects. Only the attribute similarities need to be pre-computed and stored, of which the space cost is $O\left(\sum_{i=1}^{t}|V_{X_i}|^2\right)$.

In SimRank, similarities among all type objects are pre-computed, we only retrieve the similarity between query and each candidate from similarity matrix, then the time cost is $O(|V_{X_0}|(1+k)+\Gamma(k))$. The similarities are computed based on an unified matrix of the whole network, of which the space cost is $O\left(\left(\sum_{i=1}^{t}|V_{X_i}|\right)^2\right)$.

6.3. NetSim-pruning

In the baseline algorithm, two factors that increase the computational cost are involved. First, the more candidates to check, the more time the algorithm take; and second, when computing the similarity between the query and each candidate, the more attributes link to the candidate, the more time the algorithm take. Therefore, the intuition to speed up the search is to prune unpromising candidates and unimportant attributes.

6.3.1. Pruning-index

We propose an index-based pruning method, in which the closeness between each attribute and each center is pre-computed and stored in an index, called pruning-index, denoted by D. One example of pruning-index is shown as Fig. 2, where *ATid* denotes the attribute type ID, *Aid* denotes attribute ID, *Cid* denotes center ID and $\langle Cid, Closeness\rangle$ denotes the closeness from *Cid* to *Aid*.

We define the pruning-index by using nested set $D^{\varepsilon,l} = \cup_{i=1}^{t}\{D_i^{\varepsilon_i,l}\}$ corresponds to the pruning-index D, where $\varepsilon = \cup_{i=1}^{t}\{\varepsilon_i\}$ is the set of the closeness thresholds for different attribute types which satisfies $\varepsilon_i \geqslant 0$ and $\sum_{i=0}^{t}\varepsilon_i \leqslant 1$ for different i. Of which, $D_i^{\varepsilon_i,l} = \left\{D_i^{\varepsilon_i,l}(x)|x \in V_{X_i}\right\}$ corresponds to *ATid* set, where $D_i^{\varepsilon_i,l}(x) = \left\{\left\langle v, D_i^{\varepsilon_i,l}(x,v)\right\rangle \middle| v \in V_{X_0} \wedge D_i^{\varepsilon_i,l}(x,v) \neq 0\right\}$ corresponds to sets of $\langle Cid, Closeness\rangle$. $\left\langle v, D_i^{\varepsilon_i,l}(x,v)\right\rangle$ is a tuple that corresponds to $\langle Cid, Closeness\rangle$. $D_i^{\varepsilon_i,l}(x,v)$ measures the closeness from x to v, which is defined as:

$$D_i^{\varepsilon_i,l}(x,v) = \sum_{y\in O_i(v)} rw^*(X_0X_i)w^*(v,y)M_i^l(x,y) \tag{7}$$

if right-hand side of Eq. (7) is bigger than ε_i, otherwise $D_i^{\varepsilon_i,l}(x,v) = 0$, where ε_i is the closeness threshold for X_i type attributes. Then Eq. (5) can be defined approximately as $S_\varepsilon^l(u,v) = 1$ if $u = v$, otherwise:

$$S_\varepsilon^l(u,v) = \sum_{i=1}^{t}\sum_{x\in O_i(u)} w^*(u,x)D_i^{\varepsilon_i,l}(x,v) \tag{8}$$

In fact, the informative attributes closed to one center are usually not too many, for example, a paper focus on similarity search topic may have higher closeness to terms *SimRank*, *P-Rank*, etc., but have lower closeness to term *physics*, *sports*, etc. Lower closeness attributes usually make little contribution for similarity computation and decrease computation efficiency. In order to decrease searching cost, we tune ε_i to limit the closed X_i type attribute set size corresponds to centers, by which too low closeness attributes can be pruned, and hence the cost for computing center similarity can be decreased. We compare the query u with only the centers in set $D_i^\varepsilon(x)$ for its all type attributes x, therefore, the unpromising candidates can also be pruned by turning closeness thresholds. Note that only the elements corresponding to non-zero closeness are stored.

For each attribute type X_i, we can compute the closeness between the centers and the attributes using Eq. (7), and store them in the pruning index, the time cost is $O(d_{O_i})$. And there are $|V_{X_0}||V_{X_i}|$ items need to be computed. So the time cost is $O(d_{O_i}|V_{X_0}||V_{X_i}|)$. The time cost for building pruning index is the time for computing the closeness between all centers and attributes, which can be derived as $O(\sum_{i=1}^{t}(d_{O_i}|V_{X_0}||V_{X_i}|))$.

Lemma 3. *Given center $v \in V_{X_0}$, attribute $x \in V_{X_i}$, path length l, decay factor $c \in (0,1)$ and closeness threshold set $\varepsilon = \cup_{i=1}^{t}\{\varepsilon_i\}$, we hold that $0 \leqslant D_i^{\varepsilon_i,l}(x,v) \leqslant D_i^{\varepsilon_i,l+1}(x,v) \leqslant rw^*(X_0X_i)$.*

Proof. By Eq. (7), we can derive:

(1) If $D_i^{0,l}(x,v) \leqslant \varepsilon_i$, we get $D_i^{\varepsilon_i,l}(x,v) = 0$, otherwise, $D_i^{\varepsilon_i,l}(x,v) > 0$, which gives $D_i^{\varepsilon_i,l}(x,v) \geqslant 0$.

(2) If $D_i^{0,l}(x,v) > \varepsilon_i$ and $D_i^{0,l+1}(x,v) > \varepsilon_i$, we have

$$D_i^{\varepsilon_i,l+1}(x,v) - D_i^{\varepsilon_i,l}(x,v) = \sum_{y\in O_i(v)} rw^*(X_0X_i)w^*(v,y)\Big(M_i^{l+1}(x,y) - M_i^l(x,y)\Big),$$

by Lemma 2, we get $D_i^{\varepsilon_i,l+1}(x,v) - D_i^{\varepsilon_i,l}(x,v) \geqslant 0$, which gives $D_i^{\varepsilon_i,l}(x,v) \leqslant D_i^{\varepsilon_i,l+1}(x,v)$;
If $D_i^{0,l}(x,v) > \varepsilon_i$ and $D_i^{0,l+1}(x,v) \leqslant \varepsilon_i$, we can get $M_i^{l+1}(x,y) < M_i^l(x,y)$, which conflicts with Lemma 2;
If $D_i^{0,l}(x,v) \leqslant \varepsilon_i$ and $D_i^{0,l+1}(x,v) < \varepsilon_i$, we have $D_i^{\varepsilon_i,l+1}(x,v) - D_i^{\varepsilon_i,l}(x,v) = 0 - 0 = 0$, which gives $D_i^{\varepsilon_i,l}(x,v) = D_i^{\varepsilon_i,l+1}(x,v)$;
If $D_i^{0,l}(x,v) \leqslant \varepsilon_i$ and $D_i^{0,l+1}(x,v) > \varepsilon_i$, we have

$$D_i^{\varepsilon_i,l+1}(x,v) - D_i^{\varepsilon_i,l}(x,v) = \sum_{y\in O_i(v)} rw^*(X_0X_i)w^*(v,y)M_i^l(x,y) - 0 \geqslant 0,$$

which gives $D_i^{\varepsilon_i,l}(x,v) \leqslant D_i^{\varepsilon_i,l+1}(x,v)$.
Then we get $D_i^{\varepsilon_i,l}(x,v) \leqslant D_i^{\varepsilon_i,l+1}(x,v)$.

(3) If $D_i^l(x,v) \leqslant \varepsilon_i$, we have $D_i^{\varepsilon_i,l+1}(x,v) = 0$;
If $D_i^{0,l+1}(x,v) > \varepsilon_i$, we have

$$\begin{aligned} D_i^{\varepsilon_i,l+1}(x,v) &= \sum_{y\in O_i(v)} rw^*(X_0X_i)w^*(v,y)sim_i^{l+1}(x,y) \\ &\leqslant \sum_{y\in O_i(v)} rw^*(X_0X_i)w^*(v,y) = \sum_{y\in O_i(v)} rw^*(X_0X_i)w^*(v,y) \\ &= rw^*(X_0X_i)\sum_{y\in O_i(v)} w^*(v,y) = rw^*(X_0X_i). \end{aligned}$$

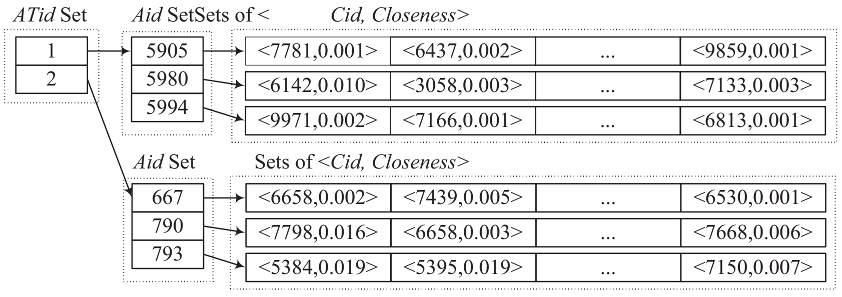


Fig. 2. Example of pruning-index.

Then we get $D_i^{\varepsilon_i,l+1}(x,v) \leqslant rw^*(X_0X_i)$.

Thus, $0 \leqslant D_i^{\varepsilon_i,l}(x,v) \leqslant D_i^{\varepsilon_i,l+1}(x,v) \leqslant rw^*(X_0X_i)$. □

Algorithm 2. NetSim-pruning algorithm.

```
Input:
  Query $u$ of $X_0$ type, $G(V,E,W)$, $k$ and $D^{\varepsilon,l}$;
Output:
  Top-$k$ most similar sorted centers;
1: Initialize $Q(\mathcal{C},\mathcal{S})$ by setting $\mathcal{C}$ and $\mathcal{S}$ as $\emptyset$;
2: For $X_i \in \Lambda$ do
3:    For $x \in O_i(u)$ do
4:       For $v \in \left\{v' \middle| \left\langle v', D_i^{\varepsilon_i,l}(x,v')\right\rangle \in D_i^{\varepsilon_i,l}(x)\right\}$ do
5:          if $v \in \mathcal{C}$ then
6:             $\mathcal{S}(v) \leftarrow \mathcal{S}(v) + w^*(u,x)D_i^{\varepsilon_i,l}(x,v)$;
7:          else
8:             $\mathcal{S}(v) \leftarrow w^*(u,x)D_i^{\varepsilon_i,l}(x,v)$;
9:             $\mathcal{C} \leftarrow \mathcal{C} \cup \{v\}$;
10:            $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathcal{S}(v)\}$;
11:         end if
12:      end for
13:   end for
14: end for
15: return $GetSortedCenter(k,Q)$;
```

6.3.2. NetSim-pruning algorithm

Algorithm 2 shows the online query processing of NetSim-pruning. For a given query center u, we firstly initialize $Q(\mathcal{C},\mathcal{S})$ by setting $\mathcal{C}$ and $\mathcal{S}$ as $\emptyset$, where $\mathcal{C}$ is the candidate set, and $\mathcal{S}$ is the set of similarities between query and candidates. For attribute type X_i, we find $x \in O_i(u)$, and get $D_i^{\varepsilon_i,l}(x,v)$ from $D^{\varepsilon_i}(x)$ for candidate v corresponds to x. The similarity between u and v is updated by accumulating $w^*(v,x)D_i^{\varepsilon_i,l}(x,v)$. $GetSortedCenter(k,Q)$ is the function for getting the k most similar centers using Q, the basic process of which is that firstly get the k most similar centers from $\mathcal{C}$ according to their corresponding similarities in $\mathcal{S}$, then sort and return them. The time cost of Algorithm 2 is $O\left(\sum_{i=1}^{t}\sum_{x\in O_i(u)} l_i^{\varepsilon_i}(x) + k|\mathcal{C}| + \Gamma(k)\right)$, where $l_i^{\varepsilon_i}(x)$ is the size of $D_i^{\varepsilon_i,l}(x)$.

In the NetSim-pruning, for a query u, we first find its corresponding attributes in the pruning index $D^{\varepsilon,l}$. For X_i attribute type, the candidate set corresponds to attribute $x \in V_{X_i}$ is $\mathcal{C}_i(x) = \left\{v' \middle| \left\langle v', D_i^{\varepsilon_i,l}(x,v')\right\rangle \in D_i^{\varepsilon_i,l}(x)\right\}$, and hence the candidate set corresponds to $O_i(u)$ is $\mathcal{C}_i = \cup_{x\in O_i(u)}\mathcal{C}_i(x)$, then the candidate set is formalized as $\mathcal{C} = \cup_{i=1}^{t}\mathcal{C}_i = \cup_{i=1}^{t}\cup_{x\in O_i(u)}\mathcal{C}_i(x) = \cup_{i=1}^{t}\cup_{x\in O_i(u)} \left\{v' \middle| \left\langle v', D_i^{\varepsilon_i,l}(x,v')\right\rangle \in D_i^{\varepsilon_i,l}(x)\right\}$. When giving a higher ε_i for each $X_i \in \Lambda$, the accumulation operations for computing similarities would become fewer, which makes the time cost lower. In this case, the size of $\mathcal{C}_i(x)$ would become smaller, which gives $\mathcal{C}_i$ a downward trend, and hence the size of $\mathcal{C}$ would have a downward trend as well. Then the time cost for choosing the k centers from $\mathcal{C}$ would become lower as well. Note that the size of $\mathcal{C}_i(x)$ is equal to the size of $D_i^{\varepsilon_i,l}(x)$.

Theorem 3. *Given centers u, $v \in V_{X_0}$, decay factor $c \in (0,1)$, path length l and closeness threshold set $\varepsilon = \cup_{i=1}^{t}\{\varepsilon_i\}$, we hold that $0 \leqslant S^l(u,v) - S_\varepsilon^l(u,v) \leqslant \sum_{i=1}^{t}\varepsilon_i$.*

Proof. By Eqs. (5), (7) and (8), we can derive that $S^l(u,v) - S_\varepsilon^l(u,v) = \sum_{i=1}^{t}\sum_{x\in O_i(u)} w^*(u,x)D_i^0(x,v) - \sum_{i=1}^{t}\sum_{x\in O_i(u)} w^*(u,x)D_i^{\varepsilon_i,l}(x,v) = \sum_{i=1}^{t}\sum_{x\in O_i(u)} w^*(u,x)(D_i^0(x,v) - D_i^{\varepsilon_i,l}(x,v))$, by Eq. (7), if $D_i^0(x,v) > \varepsilon_i$ we derive $D_i^0(x,v) - D_i^{\varepsilon_i,l}(x,v) = 0$, otherwise $0 < D_i^0(x,v) - D_i^{\varepsilon_i,l}(x,v) \leqslant \varepsilon_i$, then we have $0 \leqslant D_i^0(x,v) - D_i^{\varepsilon_i,l}(x,v) \leqslant \varepsilon_i$, which gives $S^l(u,v) - S_\varepsilon^l(u,v) \geqslant \sum_{i=1}^{t}\sum_{x\in O_i(u)} w^*(u,x)\cdot 0 = 0$, and $S^l(u,v) - S_\varepsilon^l(u,v) \leqslant \sum_{i=1}^{t}\sum_{x\in O_i(u)} w^*(u,x)\varepsilon_i = \sum_{i=1}^{t}\varepsilon_i\sum_{x\in O_i(u)} w^*(u,x) = \sum_{i=1}^{t}\varepsilon_i$.

Thus, $0 \leqslant S^l(u,v) - S_\varepsilon^l(u,v) \leqslant \sum_{i=1}^{t}\varepsilon_i$. □

Theorem 3 gives the maximal difference between NetSim-baseline and NetSim-pruning. The maximal accuracy loss of NetSim-pruning is the sum of the different type thresholds.

Theorem 4. *Given centers u, $v \in V_{X_0}$, decay factor $c \in (0,1)$, path length l and closeness threshold set $\varepsilon = \cup_{i=1}^{t}\{\varepsilon_i\}$, we hold that $\left|S_\varepsilon^l(v,u) - S_\varepsilon^l(u,v)\right| \leqslant \sum_{i=1}^{t}\varepsilon_i$.*

Proof. By Theorem 3, we have $0 \leqslant S^l(u,v) - S_\varepsilon^l(u,v) \leqslant \sum_{i=1}^{t}\varepsilon_i$, and $0 \leqslant S^l(v,u) - S_\varepsilon^l(v,u) \leqslant \sum_{i=1}^{t}\varepsilon_i$, which gives $-\sum_{i=1}^{t}\varepsilon_i \leqslant S_\varepsilon^l(v,u) - S^l(v,u) \leqslant 0$, then $0 + (-\sum_{i=1}^{t}\varepsilon_i) \leqslant \left(S^l(u,v) - S_\varepsilon^l(u,v)\right) + \left(S_\varepsilon^l(v,u) - S^l(v,u)\right) \leqslant \sum_{i=1}^{t}\varepsilon_i + 0$. By Theorem 1, we have $S^l(u,v) = S^l(v,u)$, we get $-\sum_{i=1}^{t}\varepsilon_i \leqslant S_\varepsilon^l(v,u) - S_\varepsilon^l(u,v) \leqslant \sum_{i=1}^{t}\varepsilon_i$, which gives $|S_\varepsilon^l(v,u) - S_\varepsilon^l(u,v)| \leqslant \sum_{i=1}^{t}\varepsilon_i$. □

Theorem 4 shows that the similarity under NetSim-pruning is not symmetrical and gives the maximal difference between $S_\varepsilon^l(u,v)$ and $S_\varepsilon^l(u,v)$. Note that $S^l(u,v) = S_\varepsilon^l(u,v)$ if $\varepsilon_i = 0$ for all $X_i \in \Lambda$.

Theorem 5. *Given centers u, $v \in V_{X_0}$, decay factor $c \in (0,1)$, path length l and closeness threshold set $\varepsilon = \cup_{i=1}^{t}\{\varepsilon_i\}$, we hold $0 \leqslant S_\varepsilon^l(u,v) \leqslant S_\varepsilon^{l+1}(u,v) \leqslant 1$.*

Proof. From Eq. (8), we can derive $S_\varepsilon^l(u,v) = S_\varepsilon^{l+1}(u,v) = 1$ for $u = v$, and if $u \neq v$, we have $D_i^{\varepsilon_i,l}(x,v) \geqslant 0$, which gives

$S_\varepsilon^l(u,v) = \sum_{i=1}^{t}\sum_{x\in O_i(u)} w^*(u,x)D_i^{\varepsilon_i,l}(x,v) \geqslant 0$, by Lemma 3, we have $0 \leqslant D_i^{\varepsilon_i,l}(x,v) \leqslant D_i^{\varepsilon_i,l+1}(x,v) \leqslant rw^*(X_0X_i)$, which gives

$$
\begin{aligned}
0 \leqslant S_\varepsilon^l(u,v) &= \sum_{i=1}^{t}\sum_{x\in O_i(u)} w^*(u,x)D_i^{\varepsilon_i,l}(x,v)\\
&\leqslant \sum_{i=1}^{t}\sum_{x\in O_i(u)} w^*(u,x)D_i^{\varepsilon_i,l+1}(x,v) = S_\varepsilon^{l+1}(u,v)\\
&\leqslant \sum_{i=1}^{t}\sum_{x\in O_i(u)} w^*(u,x)rw^*(X_0X_i)\\
&= \sum_{i=1}^{t} rw^*(X_0X_i)\sum_{x\in O_i(u)} w^*(u,x) = 1.
\end{aligned}
$$

Therefore, $0 \leqslant S_\varepsilon^l(u,v) \leqslant S_\varepsilon^{l+1}(u,v) \leqslant 1$. □

Theorem 5 shows the monotonicity property of NetSim-pruning that the iterative NetSim-pruning is non-decreasing with path length increasing as well as NetSim-baseline.

7. Experimental study

Experiments are done on a 2.39 GHz Intel(R) Xeon(R) CPU with 64 GB main memory, running Windows Server 2008 R2 Enterprise. All algorithms are implemented in C++ and compiled using VS 2010.

Table 1
Queries that are picked from DBLP and Amazon.

	DBLP	Amazon
1	Software-reliability engineering: technology for the 1990s	Alpha teach yourself grammar and style in 24 hours (Book)
2	ARIAL: rapid prototyping for mixed and parallel platforms	Clinical handbook of psychotropic drugs (Book)
3	Everyday computer graphics	Spanish for life textbook with atajo CD-ROM (Book)
4	Movement Problems for 2-dimensional linkages	Negotiating skills for managers (Book)
5	Floating search methods in feature selection	Distributed systems: principles and paradigms (Book)
6	Genetic local search algorithms for the travelling salesman problem	Managerial psychology: managing behavior in organizations (Book)
7	Software engineering in Asia	A first course in stochastic processes (Book)
8	Optimal detection of sequence similarity by local alignment	Principles of operations management (Book)
9	Teaching data base systems using date and computing surveys	First course in Fourier analysis, A (Book)
10	Kernel methods for pattern analysis	Distributed systems (2nd Edition) (Book)
11	People, organizations, and process improvement	Research design in clinical psychology (4th Edition) (Book)
12	A Stochastic approach to parsing	Behavior and medicine (Book)
13	Semantics and conversations for an agent communication language	Understanding Shakespeare: Hamlet (Video)
14	Generalizing the notion of schema in genetic algorithms	Lyric language Spanish/English combo2 (Video)
15	A data flow processor array system: design and analysis	Handbook of home health nursing procedures (Book)
16	A sublinear time approximation scheme for clustering in metric spaces	Chiropractic technique: principles and procedures (Book)

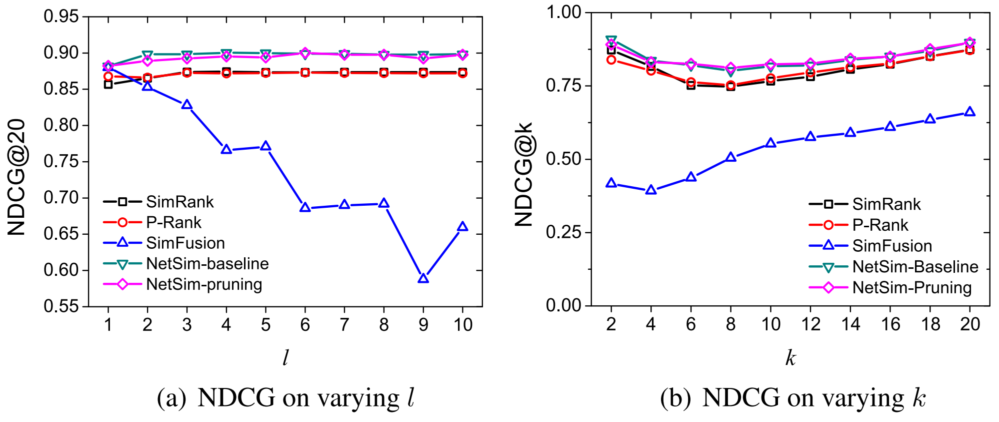


Fig. 3. Effectiveness comparison on DBLP.

7.1. Setup

7.1.1. Datasets

- **DBLP** We extract a bibliographic network from DBLP citation dataset[1] (Tang, Yao, Zhang, & Zhang, 2010; Tang et al., 2011; Tang et al., 2008). It contains 314,753 papers with 2,376,644 citations, 2942 venues and 180,322 authors. After removing stop words in the papers' titles, we get around 49,688 terms appearing more than once. We consider the paper as center that *is published in* one venue, *contains* several terms. We choose 13,837 objects with 68,984 different type links, specifically, there are 5952 papers, 6929 terms and 956 venues.
- **Amazon** Co-purchased network dataset is extracted from Amazon dataset[2] (Leskovec, Adamic, & Huberman, 2007). There are 355,601 products with 2,359,584 co-purchased relationships, 36,591 categories, and 42,890 terms appearing more than once. We consider the product as the center that *contains* several terms and *belongs to* some categories. We choose 17,570 objects with 94,112 different type links, specifically, there are 5819 products, 7267 terms and 4484 categories.

7.1.2. Comparison methods and evaluation

NetSim is compared with SimRank, P-Rank and SimFusion. All the comparison methods are implemented strictly following their papers. The decay factor c is set as 0.8. Unnecessary connections that can decrease the evaluation accuracy are removed from attribute networks. On DBLP, we set term type closeness threshold ε_T as 0.003 and venue type closeness threshold ε_V as 0.002; and on Amazon, we set term type closeness threshold ε_T as 0.003 and category type closeness threshold ε_C as 0.0015.

We randomly pick 16 query centers from both DBLP and Amazon datasets, all of them are shown in Table 1. On DBLP, we pick 16 papers, we list only the titles of them; and on Amazon, we pick 16 products, and we list their titles and the groups they belong to in the followed brackets.

We use Normalized Discounted Cumulative Gain (NDCG) to evaluate the effectiveness of similarity ranking lists (Järvelin & Kekäläinen, 2002). The NDCG value at the kth position (NDCG@k) of the ranking result is computed according to the similarity levels which are set as: 2 (highly relevant), 1 (marginally relevant), and 0 (irrelevant). And the similarity levels are labeled by 8 persons in a double-blind fashion.

Efficiency comparison includes the running time for pre-computing similarity matrices and execution time of online query processing. As mentioned, all optimization methods for fast computing SimRank can be applied to NetSim for fast pre-computing attribute similarity matrices by Eq. (4). For supporting our opinion, we use one optimization technique of SimRank (Lizorkin et al., 2010) to speed up NetSim, and test the time cost of NetSim, SimRank, optimized NetSim (Opt-NetSim) and optimized SimRank (Opt-SimRank), more details are shown in Section 7.5.

[1] <http://www.arnetminer.org/citation>.
[2] <http://snap.stanford.edu/data/amazon-meta.html>.

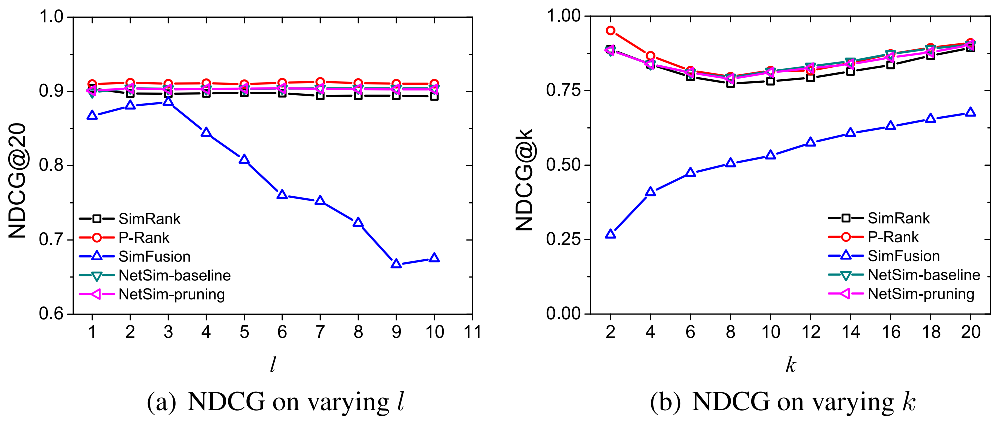


Fig. 4. Effectiveness comparison on Amazon.

Table 2
Case study on DBLP for query of title: *People, Organizations, and Process Improvement*.

Rank	SimRank	P-Rank	SimFusion	NetSim-baseline	NetSim-pruning
1	People, organizations, and process improvement	People, organizations, and process improvement	Porting OpenVMS from VAX to alpha AXP	People, organizations, and process improvement	People, organizations, and process improvement
2	Software process improvement at Raytheon	Software process improvement at Raytheon	HCI for older and disabled people in the Queen Mother Research Centre at Dundee University, Scotland	Software process improvement at Raytheon	Software process improvement at Raytheon
3	Software process improvement at Hughes aircraft	Top-down vs. bottom-up process improvement	NEATER2: a PL/I source statement reformatter	Top-down vs. bottom-up process improvement	Top-down vs. bottom-up process improvement
4	Top-down vs. bottom-up process improvement	Software process improvement at Hughes aircraft	The panopticon and the performance arena: hci reaches within	Software process improvement at Hughes aircraft	Software process improvement at Hughes aircraft
5	A parallel-programming process model	Software process improvement: blueprints versus recipes	Alpha AXP architecture	Software process improvement: blueprints versus recipes	Software process improvement: blueprints versus recipes

7.2. Effectiveness

7.2.1. NDCG Value

Fig. 3(a) shows the NDCG@20 value versus path length l on Amazon. We observe that NetSim performs better than other methods. The NDCG value becomes stable as l increases on each method except SimFusion, which suggests the convergence of these methods. The result on NetSim is consistent with Theorems 2 and 5. SimFusion shows a significant downward NDCG value with l increasing, this is because SimFusion is not a convergent algorithm (Xi et al., 2005). The NDCG value of NetSim-baseline and NetSim-pruning are close, which is consistent with Theorem 3. Fig. 3(b) shows the NDCG value versus rank k, where $l = 10$. From $k = 2$ to 8, there are downward NDCG value except SimFusion, this is because each center are similar to itself, so the value of NDCG value at $k = 1$ is 1, which affects the value from $k = 2$ to 8.

Fig. 4(a) shows the NDCG@20 value change versus l on Amazon; and Fig. 4(b) shows the NDCG value change versus k, where $l = 10$. P-Rank gives the best ranking quality. NDCG value of NetSim-baseline and NetSim-pruning is lower than P-Rank and higher than other methods, which is different from the result on DBLP, this is because different methods may be suit for different datasets, which is common for similarity search algorithms. In Fig. 4(a), the NDCG value of SimRank does not increase strictly with l

Table 3
Case study on Amazon for query of title: *First Course in Fourier Analysis, A (Book)*.

Rank	SimRank	P-Rank	SimFusion	NetSim-baseline	NetSim-pruning
1	First course in fourier analysis, A (Book)	First course in fourier analysis, A (Book)	The topology of fibre bundles (Book)	First course in Fourier analysis, A (Book)	First course in Fourier analysis, A (Book)
2	Fourier analysis and boundary value problems (Book)	Fourier analysis and boundary value problems (Book)	Riemannian geometry (Graduate texts in mathematics) (Book)	Fourier analysis and boundary value problems (Book)	Fourier analysis and boundary value problems (Book)
3	The Fourier transform & Its applications (Book)	The Fourier transform & Its applications (Book)	Fundamentals of Fourier transform infrared spectroscopy (Book)	The Fourier transform & Its applications (Book)	The Fourier transform & Its applications (Book)
4	An introduction to harmonic analysis (Book)	An introduction to harmonic analysis (Book)	Fourier transform infrared spectrometry (Book)	Functions of one complex variable II (Graduate texts in mathematics) (Book)	Functions of one complex variable II (Graduate texts in mathematics) (Book)
5	Elementary classical analysis (Book)	Complex analysis in one variable (Book)	Differential geometry of curves and surfaces (Book)	Real and complex analysis (Higher mathematics series) (Book)	Real and complex analysis (Higher mathematics series) (Book)

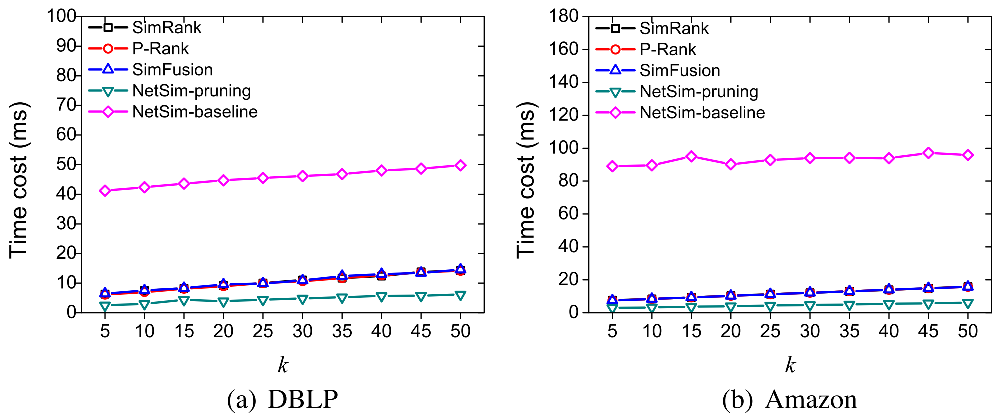


Fig. 5. Execution time of online query processing.

Table 4
Pre-computation time (min).

Dataset	SimRank	P-Rank	SimFusion	NetSim
DBLP	177.78	369.06	162.85	106.42
Amazon	356.99	681.56	353.77	149.58

increasing, this is probably because the ranking lists are not 100% coherent with data.

7.2.2. Case study

Some case studies have been done on both DBLP and Amazon. Although the judgement of similarity might be subjective and difficult for human beings, we still find some interesting results by making use of different algorithms. On DBLP, the top 5 similar papers returned for the given query of paper titled "People, Organizations, and Process Improvement" by different algorithms are shown in Table 2. This query is a paper related with "software development". From which we found that most papers in the returned lists are highly relevant to the given query except the list of SimFusion. For example, "First Course in Fourier Analysis, A (Book)" is same to the given query, which is similar to itself; "Software Process Improvement At Raytheon", "Software Process Improvement at Hughes Aircraft" and "Software Process Improvement: Blueprints versus Recipes" are concerned with "Software Process Improvement", which is the sub-topic of "software development", and they are obviously two similar topics; "A Parallel-Programming Process Model" is a paper on the topic "Parallel-Programming", which is an important software development model, therefore, the paper is also relevant to the given query; "Top-Down vs. Bottom-Up Process Improvement" is concerned with "Top-Down" and "Bottom-Up", which is relevant to the topic "software development" as well. We also found that lists returned by NetSim-baseline and NetSim-pruning are same, besides, which is consistent with the comparisons on NDCG value in subsubSection 7.2.1.

Table 3 shows the case study on Amazon for the given query of a book product titled "First Course in Fourier Analysis, A". This query is a book concerned with "Fourier analysis". We found that most books in the returned lists are highly relevant to the given query as well. "First Course in Fourier Analysis, A" is certainly similar to itself; "The Fourier Transform & Its Applications", "Fourier Analysis and Boundary Value Problems", "Fundamentals of Fourier Transform Infrared Spectroscopy" are all concerned with the topic "Fourier analysis" or "Fourier Transform", which are obviously similar to "Fourier analysis", therefore, these books are similar to the given query; "An Introduction to Harmonic Analysis" is on the topic "Harmonic Analysis", which is in fact the extended form of topic "Fourier analysis", therefore, this book is similar to the given paper as well; Similar cases can be found in other results as well, such as "Fourier Transform Infrared Spectrometry" and "Differential Geometry of Curves and Surfaces".

From the case studies on Amazon as well as DBLP, we conclude that NetSim can also really reflect the reality to single out similar centers as well as the state-of-the-art methods. We also conclude that NetSim-pruning is effective as well as NetSim-baseline.

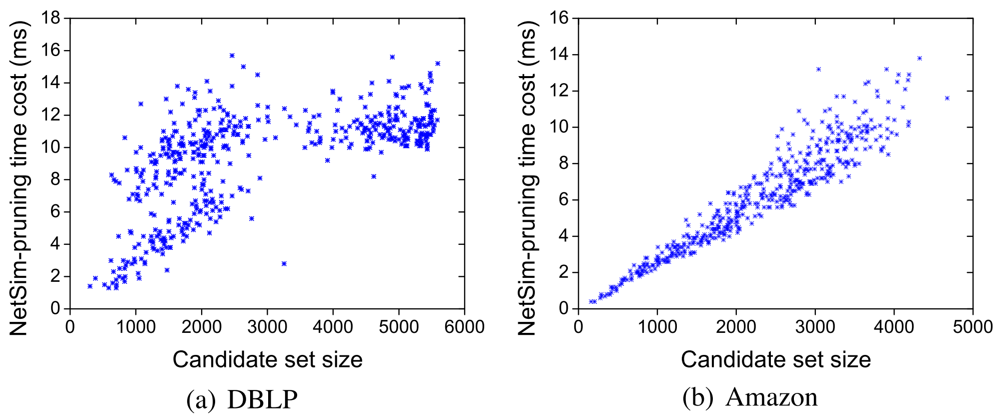
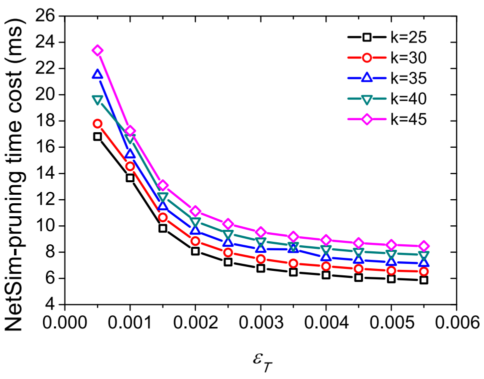
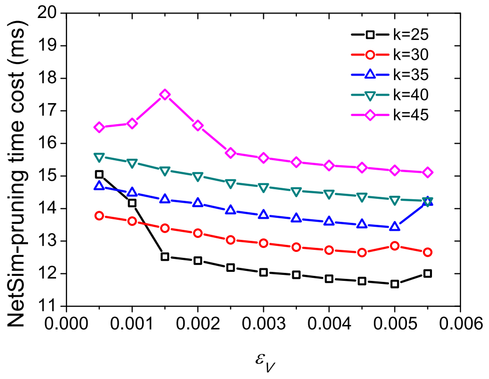


Fig. 6. Candidate set size v.s. NetSim-pruning.



(a) On varying ε_T



(b) On varying ε_V

Fig. 7. Attribute closeness threshold v.s. NetSim-pruning on DBLP.

7.3. Efficiency

7.3.1. Comparing NetSim with other methods

Fig. 5(a) and (b) show the execution time of online query processing with k increasing on DBLP and Amazon, where $l = 10$. Generally, the time cost becomes high as k increases. The lines of SimRank, P-Rank, Simfusion are almost overlapped, this is because the size of their similarity matrix is almost same. NetSim-baseline is the most time-consuming method, and NetSim-pruning improves the efficiency evidently. These results demonstrate that NetSim-pruning is more efficient than other methods.

Table 4 shows the recorded running time for pre-computing similarity matrices. On both of the two datasets, P-Rank is the most time-consuming, since it considers out- and in-links for computing similarity, which decreases the efficiency. On DBLP, the time cost of NetSim are 60.00% of SimRank, 28.83% of P-Rank and 65.35% of SimFusion respectively; and on Amazon are $41.90\%, 21.95\%$ and 42.28% respectively.

7.3.2. NetSim-pruning efficiency

Fig. 6(a) and (b) show the candidate set size v.s. NetSim-pruning time cost on DBLP and Amazon respectively. We can observe that more candidates give more time cost. On Amazon, time cost is almost linear to the candidate set size, and such phenomenon is not so evident on DBLP.

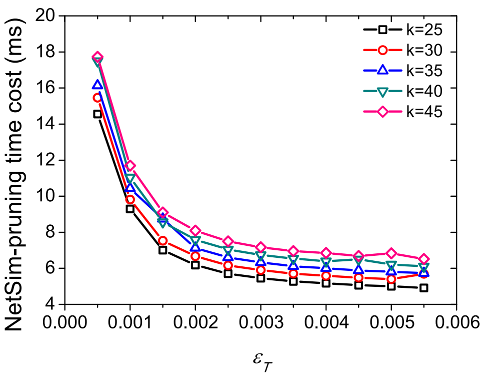
Fig. 7(a) shows the time cost on varying term type closeness threshold ε_T on DBLP, where venue type closeness threshold ε_V is set as 0.001. The downward curves on different k show the decrease on time cost as ε_T increases, this is because lower closeness centers correspond to the X_T type attributes are pruned. Fig. 7(b) shows the time cost on varying ε_V, where ε_T is set as 0.001. The downward trend of the curves is not so evident as above, this is because there are only 956 venues in our test dataset, the venue type closeness threshold between 0.0005 and 0.005 is not high enough for pruning the unpromising candidates.

Fig. 8(a) shows the time cost on varying term type closeness threshold ε_T on Amazon, where category type closeness threshold ε_C is set as 0.001. The time cost on different k decreases as ε_T increases as well. Fig. 8(b) shows the time cost on varying ε_C, where ε_T is set as 0.001. The downward trend of the curves is not so evident as Fig. 8(a), which can be explained similar to Fig. 7(b).

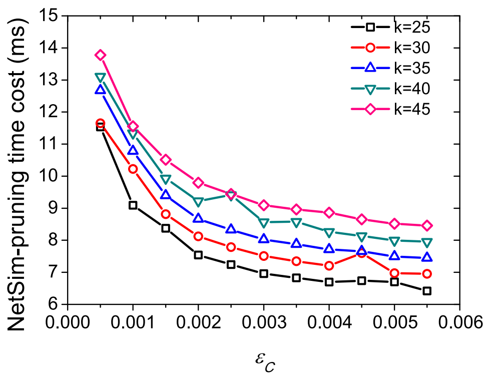
We also recorded the time cost for building pruning-index on DBLP and Amazon respectively. The time cost is 31,687 ms on DBLP, and on Amazon is 33,248 ms. From which we can see that the time cost on both of the two datasets is less than 1 min. The result demonstrate that the off-line process time cost of the NetSim-pruning is lower.

7.4. Similarity matrix size

Fig. 9(a) shows the size of non-zero similarity matrices of different methods on varying l on DBLP. We find that the matrix size becomes bigger as l increases, and suggests a stable state finally. At $l = 10$, there are 98.55% non-zero elements among the full matrix of NetSim, and 100% of other methods. The matrix size of SimRank, P-Rank and SimFusion are close. The matrix size of NetSim is 25.18% of all other methods. The similarity matrix size

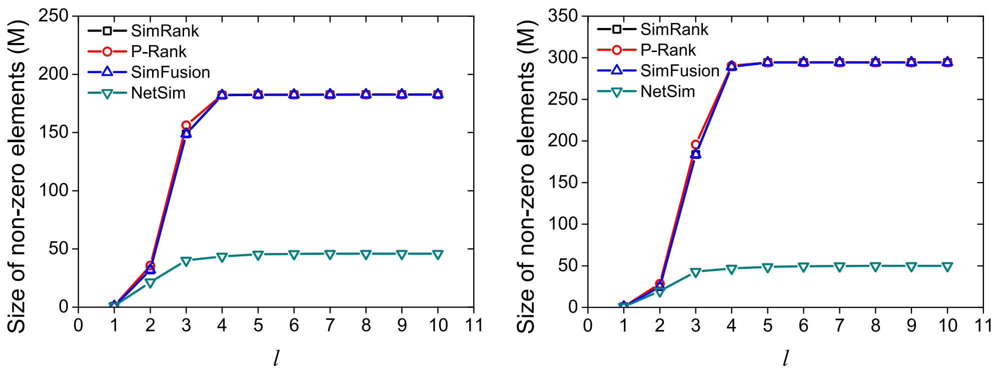


(a) On varying ε_T



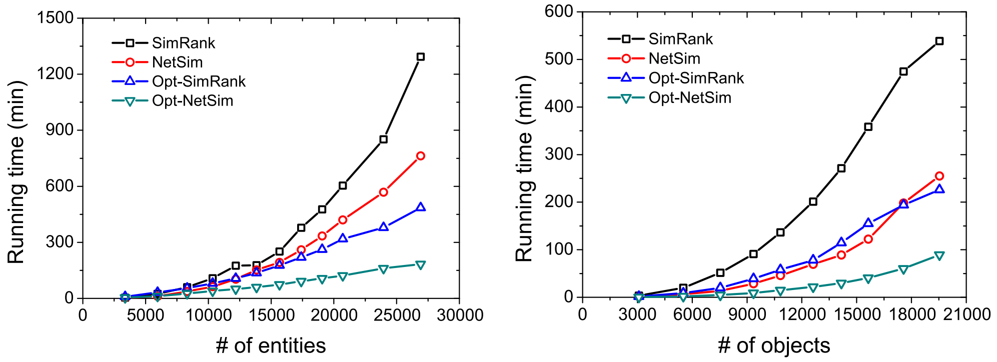
(b) On varying ε_C

Fig. 8. Attribute closeness threshold v.s. NetSim-pruning on Amazon.



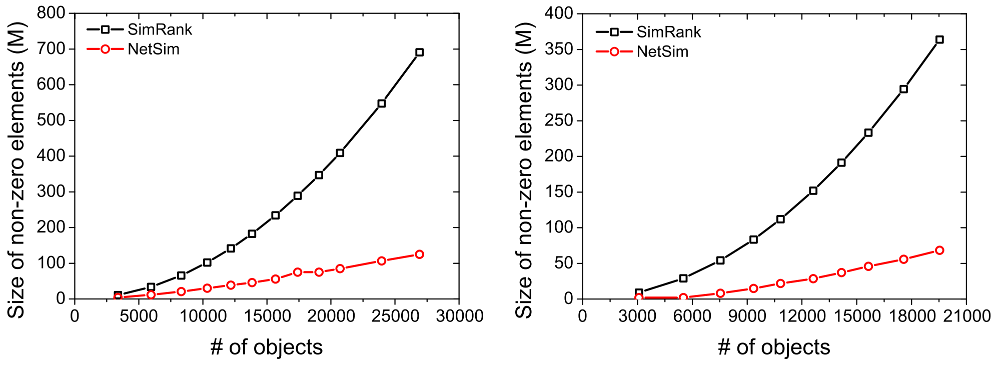
(a) DBLP (b) Amazon

Fig. 9. Similarity matrix size on varying *l*.



(a) DBLP (b) Amazon

Fig. 10. Pre-computation time v.s. # of objects on Optimized SimRank and NetSim.



(a) DBLP (b) Amazon

Fig. 11. Similarity matrix size v.s. # of objects.

change on Amazon is shown in Fig. 9(b). At $l = 10$, there are 98.32% non-zero elements among the full matrix of NetSim, and 100% of other methods. The matrix size of NetSim is 23.62% of other methods.

7.5. Test on increasing objects

Next we test the performance of NetSim by comparing with SimRank on increasing objects, where $l = 10$ and $k = 50$. Fig. 10 shows the running time comparison for pre-computing similarity matrices in off-line stage on varying object number on DBLP and Amazon. From which we can find that: (1) NetSim gives evident efficiency improvement with objects increasing by comparing with SimRank, and the improvement is more evident on Amazon; (2) the time cost of Opt-NetSim is lower than NetSim with objects increasing, and the time cost Opt-SimRank is lower than SimRank, which shows the optimization technique can really applied to both NetSim and SimRank for improving the computation efficiency; and (3) the time cost of Opt-NetSim is lower than Opt-SimRank with objects increasing, which shows the optimized NetSim is more efficient compared with the optimized SimRank.

The similarity matrix size on varying object number is shown in Fig. 11. On both of the two datasets, the matrix size of NetSim is evidently smaller than SimRank since the number of the corre-

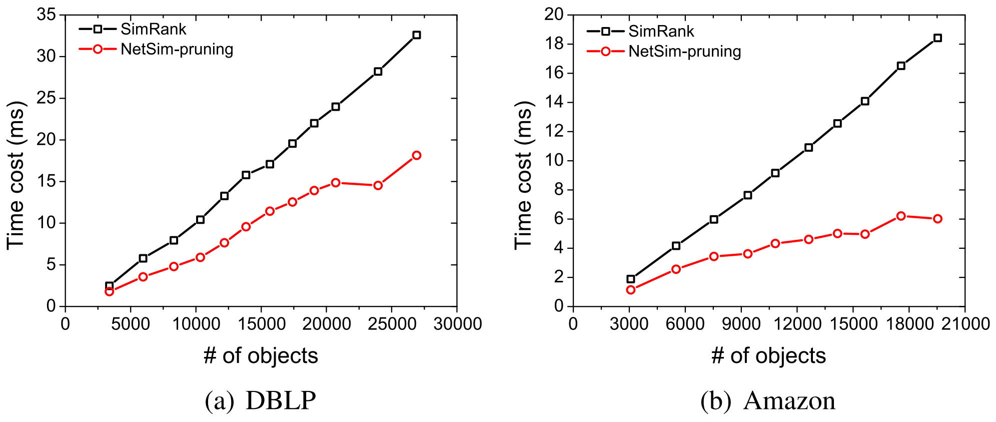


Fig. 12. Query execution time v.s. # of objects.

sponding attributes of all types is significantly less than the number of all type objects in the whole x-star network. Fig. 12 shows the execution time of online query processing of NetSim-pruning and SimRank with objects increasing. On both DBLP and Amazon, the NetSim-pruning performs more efficient than SimRank. And on Amazon, the efficiency improvement is more evident, and the time increment of NetSim-pruning is very small with objects increasing. The optimization technique of NetSim and SimRank reduces only the pre-computation time cost, and have no affects on the matrix size and query processing time, so the repeated results are not shown here.

8. Conclusion and future work

The research presented in this paper tackled the problem of similarity search in x-star networks. A structural-based similarity measure NetSim is proposed for efficiently computing similarity between centers. Compared to traditional methods, the computation cost would be decreased significantly, since NetSim computes center similarities on-line according to the attribute similarities which are computed in the off-line stage over small attribute networks. Besides, the global structure information is utilized as well since the global structure information is integrated into attribute networks, and then the effectiveness of NetSim and the state-of-the-art methods are close. In order to support fast query processing, we developed a pruning algorithm NetSim-pruning, by which the unpromising candidates can be pruned by setting the user-controlled thresholds.

The strengths of this paper can be summarized as follows. Firstly, NetSim requires less time and space cost than existing methods. This is because the scale of attribute network is significantly smaller than the whole x-star network, the total cost of similarity computation over the attribute networks would be significantly decreased. Secondly, the pruning algorithm is efficient for fast query processing, since the unpromising candidates can be pruned by setting thresholds. Thirdly, our approach is in fact a general framework on similarity search, since the existing optimization techniques on similarity computation can be easily taken into attribute similarity computation process for further speeding up NetSim.

This work has theoretical and practical implications. In theory, this is the first attempt for a solution to the efficiency problem of similarity search in information networks according to attribute similarity matrices that have not been previously explored. Our proposed NetSim is a new similarity search framework, which can be easily combined with the existing optimization techniques, and we hope that it will help begin a fruitful discussion among scholars. As far as practical implications are concerned, NetSim can be easily applied to many real applications, including query expansion, clustering and web search engine. NetSim provides an effective and efficient evaluation of underlying similarity, which can satisfy the requirements of above applications and map human intuition under different real settings of information networks. By NetSim, the system performance would be improved significantly.

Our approach also has some limitations which are summarized below. Firstly, our approach focuses on reducing the computational cost of similarity search in static information network, and the case of dynamic information networks is not considered. Secondly, although this paper considers the relation types for computing similarity, the semantic of relations is not explored thoroughly. Finally, NetSim is proposed for measuring similarity in x-star networks, the information networks beyond x-star network schema are not addressed.

Accordingly, in future work, we will focus on the followed aspects to continue our study. First is to study problem of increment computation on NetSim for adapting dynamic information networks, which may directly benefit from increment computation methods (Li et al., 2010a; Yu & Lin, 2013). Secondly, we will give much attention on semantic of relationships among objects and for developing a new semantic similarity measure by fusing NetSim and existing computation techniques on semantic similarity (Sánchez & Batet, 2013; Sánchez, Batet, Isern, & Valls, 2012). Finally, we plan to extend NetSim to the information networks beyond x-star network schema for handling more complex data.

Acknowledgments

This work was supported by National Science Foundation of China Grants 61170007, 61170006 and 61202376; Bidding Program of Shanghai Research Institute of Publishing and Media Grants SAYB1410; and Innovation Program of Shanghai Municipal Education Commission Grants 13YZ075 and 13ZZ111.

References

Amsler, R., (1972). December. *Application of citation-based automatic classification. Technical report*, The University of Texas at Austin Linguistics Research Center.
Cai, Y., Zhang, M., Ding, C., & Chakravarthy, S. (2010). Closed form solutions of similarity algorithms. In *SIGIR* (pp. 709–710).
Cheng, H., Zhou, Y., & Yu, J. X. (2011). Clustering large attributed graphs: A balance between structural and attribute similarities. *TKDD, 5*(2), 12.
Faloutsos, M., Faloutsos, P., & Faloutsos, C. (1999). On power-law relationships of the internet topology. In *SIGCOMM* (pp. 251–262).
Fogaras, D., & Rácz, B. (2005). Scaling link-based similarity search. In *WWW* (pp. 641–650).

Ganesan, P., Garcia-Molina, H., & Widom, J. (2003). Exploiting hierarchical domain structure to compute similarity. *ACM Transactions on Information Systems, 21*(1), 64–93.

Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems, 20*(4).

Jeh, G., & Widom, J. (2002). Simrank: A measure of structural-context similarity. In *KDD* (pp. 538–543).

Kessler, M. M. (1963). Bibliographic coupling between scientific papers. *American Documentation, 14*, 10–25.

Lee, P., Lakshmanan, L. V. S., & Yu, J. X. (2012). On top-k structural similarity search. In *ICDE* (pp. 774–785).

Leskovec, J., Adamic, L. A., & Huberman, B. A. (2007). The dynamics of viral marketing. *TWEB, 1*(1).

Li, C., Han, J., He, G., Jin, X., Sun, Y., Yu, Y., & Wu, T. (2010a). Fast computation of simrank for static and dynamic information networks. In *EDBT* (pp. 465–476).

Li, P., Liu, H., Yu, J. X., He, J., & Du, X. (2010b). Fast single-pair simrank computation. In *SDM* (pp. 571–582).

Lin, Z., King, I., & Lyu, M. R. (2006). Pagesim: A novel link-based similarity measure for the world wide web. In *WI* (pp. 687–693).

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

Lizorkin, D., Velikhov, P., Grinev, M. N., & Turdakov, D. (2010). Accuracy estimate and optimization techniques for simrank computation. *VLDB J., 19*(1), 45–66.

Maguitman, A. G., Menczer, F., Erdinc, F., Roinestad, H., & Vespignani, A. (2006). Algorithmic computation and approximation of semantic similarity. *World Wide Web, 9*(4), 431–456.

Pan, J. -Y., Yang, H. -J., Faloutsos, C., & Duygulu, P. (2004). Automatic multimedia cross-modal correlation discovery. In *KDD* (pp. 653–658).

Sánchez, D., & Batet, M. (2013). A semantic similarity method based on information content exploiting multiple ontologies. *Expert Systems with Applications, 40*(4), 1393–1399.

Sánchez, D., Batet, M., Isern, D., & Valls, A. (2012). Ontology-based semantic similarity: A new feature-based approach. *Expert Systems with Applications, 39*(9), 7718–7728.

Sarkar, P., & Moore, A. W. (2010). Fast nearest-neighbor search in disk-resident graphs. In *SIGKDD* (pp. 513–522).

Satuluri, V., & Parthasarathy, S. (2011). Symmetrizations for clustering directed graphs. In *EDBT* (pp. 343–354).

Small, H. G. (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.

Sun, Y., Yu, Y., & Han, J. (2009). Ranking-based clustering of heterogeneous information networks with star network schema. In *KDD* (pp. 797–806).

Sun, Y., Han, J., Yan, X., Yu, P. S., & Wu, T. (2011). Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB, 4*(11), 992–1003.

Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., & Su, Z. (2008). Arnetminer: Extraction and mining of academic social networks. In *KDD* (pp. 990–998).

Tang, J., Yao, L., Zhang, D., & Zhang, J. (2010). A combination approach to web user profiling. *ACM TKDD, 5*(1), 1–44.

Tang, J., Zhang, J., Jin, R., Yang, Z., Cai, K., Zhang, L., et al. (2011). Topic level expertise search over heterogeneous networks. *Machine Learning Journal, 82*(2), 211–237.

Xi, W., Fox, E. A., Fan, W., Zhang, B., Chen, Z., Yan, J., & Zhuang, D. (2005). Simfusion: Measuring similarity using unified relationship matrix. In *SIGIR* (pp. 130–137).

Yin, X., Han, J., & Yu, P. S. (2006). Linkclus: Efficient clustering via heterogeneous semantic links. In *VLDB* (pp. 427–438).

Yu, W., & Lin, X. (2013). Irwr: Incremental random walk with restart. In *SIGIR* (pp. 1017–1020).

Yu, W., Lin, X., Zhang, W., Chang, L., & Pei, J. (2013). More is simpler: Effectively and efficiently assessing node-pair similarities based on hyperlinks. *PVLDB, 7*(1), 13–24.

Zhao, P., Han, J., & Sun, Y. (2009). P-rank: A comprehensive structural similarity measure over information networks. In *CIKM* (pp. 553–562).

Zhao, X., Xiao, C., Lin, X., Liu, Q., & Zhang, W. (2013). A partition-based approach to structure similarity search. *PVLDB, 7*(3), 169–180.

Zhou, Y., Cheng, H., & Yu, J. X. (2009). Graph clustering based on structural/attribute similarities. *PVLDB, 2*(1), 718–729.