Second-order CoSimRank for Similarity Measures in Social Networks

Xueting Liao, Yubao Wu and Xiaojun Cao

Department of Computer Science

Georgia State University, Atlanta, GA, USA

xliao3@student.gsu.edu, ywu28@gsu.edu, cao@gsu.edu

***Abstract*—Measuring the similarity between nodes is a fundamental challenge in social networks. The SimRank and CoSimRank are techniques widely used to calculate the similarity of two nodes in a graph. They can be applied to many applications such as recommending friends and detecting communities in social networks. Both SimRank and CoSimRank are based on random walk. They both consider first-order transition probabilities, which means that the next node to visit in random walk solely depends on the current node, like a Markov chain. However, in many real-world situations, simply considering the current node may not be enough. For example, to predict the next hyperlink a user will visit, it may not be optimal to use only the current website URL and completely ignore the history of the hyperlinks visited. In this paper, we propose a novel similarity measure technique by investigating CoSimRank to take advantage of the second-order information in a random walk process. Our extensive analysis and experiments show that the proposed second-order CoSimRank significantly outperforms the existing techniques.**

***Index Terms*—Second-order, Random Walk, CoSimRank, Social network**

I. Introduction

Social networks have received incredible growth in recent years. Facebook, Twitter and Instagram are among the most popular social networking platforms. Social graphs or networks are often used to depict personal relations of Internet users in these platforms. In a social network, the nodes represent individuals in the network, and the edges represent the social interactions between them [1]. It is crucial to mine useful knowledge from the huge and complex graph efficiently. For example, measuring the similarity is a fundamental step for many advanced applications like querying and ranking, community detection [2]–[4], link prediction [5], [6]. To measure the similarity of nodes, a common procedure is to measure the proximity between nodes. If two nodes have more and shorter paths between them, they are more likely similar than others [7]. Random walk has been proven to be a simple yet effective method to design a good proximity measure. The main idea of random walk is to allow a surfer to randomly traverse a graph based on the transition probability of the graph. Some of the popular random walk based proximity measure approaches include PageRank [8], random walk with restart [9] and SimRank [10].

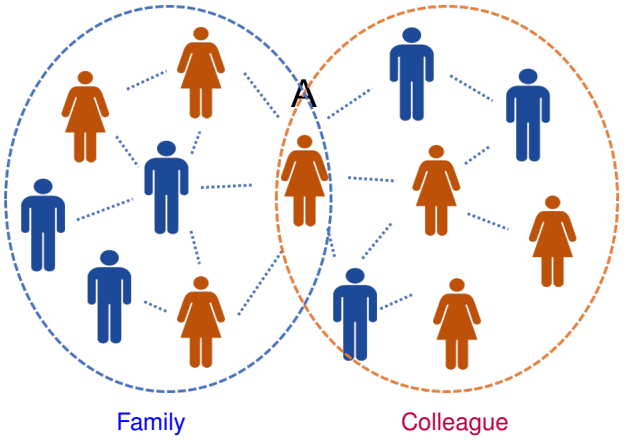


Figure 1. A subgraph of relationship

However, the existing random walk based proximity measure approaches were developed using the first-order Markov model, meaning that they do not consider any previous nodes other than the current node. This might not model real-world applications effectively.

Figure 1 shows user connections in a social network. Each node represents a user. If two nodes are connected by an edge, it means these two users are friends. Since the community is a group of nodes that are similar to each other while dissimilar with nodes in other groups, there are two communities in the figure. The left part of the figure is about families, while the right part of the figure is about colleagues. Most of the current existing random walk based measures do not consider where the surfer came from. Suppose that the random surfer came from A's family community. Based on the first-order proximity measure, the probability of visiting the nodes in colleague community is the same as the nodes in the family community because the surfer has no memory on the previous nodes. However, in real-world situations, people exploring family connections could have a higher probability of visiting another family user.

SimRank is one of the popular random walk based proximity measures. It was developed based on the intuitive assumption that two nodes are similar if they are referenced by similar nodes [10]. SimRank can be applied to any domain as long

as there are enough relevant relationships between objects. However, one of the major problems of SimRank is that it is computationally expensive since it requires calculation of all other SimRank values before determining any value between two nodes. Some researches have done work to improve its efficiency [11]–[13]. S. Rothe and H. Schütze designed CoSimRank [14], which is much faster than SimRank because it does not need to compute the similarity values of the whole graph in order to calculate the value of two nodes. W. Yu and J. McCann designed Co-Simmate to improve its speed, which reduces the time of computing all pairs of CoSimRank to just $\mathcal{O}(\log_2(\log(1/\epsilon))n^3)$ while still retaining the accuracy of the model [15]. Y. Wu et al. developed the second-order random walk in PageRank, random walk with restart, SimRank and SimRank* [16], [17]. In [18], the authors worked on the CoSimRank on dynamic graph. However, to our best knowledge, there is no work developing the second-order structure for CoSimRank.

In this paper, we propose the second-order CoSimRank, which will take advantages of the second-order structure such as trigrams to facilitate exploring the community structures and social networks. Standard adjacency matrices can only represent the first-order graph information and define node-to-node transition probabilities. If we consider the already visited node, we will need to build a tensor [19]. Using the tensor will increase the computational difficulty. Thus, we use edge-to-edge transition probabilities. Since the incidence matrices of the graph can represent such probabilities [20], we use simple matrix representations for the second-order measures for CoSimRank.

The rest of the paper is organized as follows. Section II introduces the related work. Section III presents the problem formulation. In Section IV, the second-order CoSimRank method and the necessary analysis are elaborated. Section V compares our method and with the baseline methods such as first-order CoSimRank. Finally, conclusions are drawn in Section VI.

II. Background and Problem Formulation

In this section, we introduce the background knowledge of CoSimRank. We first introduce first-order CoSimRank then the second-order random walk. Table I shows the main symbols we will use in this paper.

A. CoSimRank

CoSimRank starts with a probability distribution at each time point in the first-order random walk. Suppose the random surfer starts from node i and randomly explores the network following the first-order random walk. Let $\mathbf{r}^{t,\,i}$ represent the probability distribution vector at time point t when the random surfer starts from node i. Since the random surfer starts from node i at time point t (t = 0), we have that $\mathbf{r}^{0,i}$ = $\mathbf{q}^i$, where

$$[\mathbf{q}^i]_k = \begin{cases} 1 & \text{if } k = i, \\ 0 & \text{if } k \neq i. \end{cases}$$

Table I
Main symbols

symbols	definitions
$G(V, E)$	directed graph G with node set V and edge set E
I_i, O_i	set of in-/out-neighbor nodes of node i
n, m, σ	number of nodes; number of edges; $\sigma = \sum_{i \in V} \lvert I_i \rvert \cdot \lvert O_i \rvert$
$\mathbf{B}$	$n \times m$ incidence matrix, $[\mathbf{B}]_{i,u} = 1 : u$ is an out-edge of i
$\mathbf{E}$	$m \times n$ incidence matrix, $[\mathbf{E}]_{u,i} = 1 : u$ is an in-edge of i
$w_{i,j}$, w_i	weight of edge (i, j); out-degree of i: $w_i = \sum_{j \in O_i} w_{i,j}$
$\mathbf{W}$	$m \times m$ diagonal matrix, $[\mathbf{W}]_{u,u} = w_{i,j}$ if edge $u = (i, j)$
$\mathbf{D}$	$n \times n$ diagonal matrix, $[\mathbf{D}]_{i,i} = w_i$
$p_{i,j}$	transition probability from node i to j
$p_{i,j,k}$	transition prob. from j to k if the surfer came from i
$p_{u,v}$	transition prob. from edge u to v, $p_{(i,j),(j,k)} = p_{i,j,k}$
$\mathbf{P}$	$n \times n$ node-to-node transition matrix, $[\mathbf{P}]_{i,j} = p_{i,j}$
$\mathbf{H}$	$n \times m$ node-to-edge transition matrix, $[\mathbf{H}]_{i,(i,j)} = p_{i,j}$
$\mathbf{M}$	$m \times m$ edge-to-edge transition matrix, $[\mathbf{M}]_{u,v} = p_{u,v}$
$r_{i,j}$, r_i	$r_{i,j}$: proximity value of node i w.r.t. node j; $r_i = r_{i,q}$
$\mathbf{r}, \mathbf{R}$	$\mathbf{r} : n \times 1$ vector, $\mathbf{r}_i = r_i$; $\mathbf{R} : n \times n$ matrix, $[\mathbf{R}]_{i,j} = r_{i,j}$
s_u, $s_{(i,j)}$	proximity value of edge u or (i, j) w.r.t. query node q
$s_{u,v}$	proximity value between edges u and v
$\mathbf{s}, \mathbf{S}$	$\mathbf{s} : m \times 1$ vector, $\mathbf{s}_u = s_u$; $\mathbf{S} : m \times m$ matrix, $[\mathbf{S}]_{u,v} = s_{u,v}$

For time point t ($t \geq 1$), let $\mathbf{P}$ represent the transition matrix, we have

$$\mathbf{r}^{t,i} = \mathbf{P}^{\mathrm{T}} \mathbf{r}^{t-1,i} \tag{1}$$

Thus, $\mathbf{r}^{t,\,i}$ can be represented as

$$\mathbf{r}^{t,i} = \begin{cases} \mathbf{q}^i & \text{if } t = 0, \\ \mathbf{P}^{\mathrm{T}} \mathbf{r}^{t-1,i} & \text{if } t \geq 1. \end{cases}$$

CoSimRank is defined by using the probability distribution vectors $\mathbf{r}_{t,i}$. Specifically, the CoSimRank values $\mathbf{s}_{(i,j)}$, between node i and j is defined as

$$\mathbf{s}_{(i,j)} = \sum_{t=0}^{\infty} c^t \cdot \left\langle \mathbf{r}^{t,i}, \mathbf{r}^{t,j} \right\rangle \tag{2}$$

where the operator $\langle \cdot, \cdot \rangle$ represents the inner product

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{k=1}^{n} x_k \cdot y_k \tag{3}$$

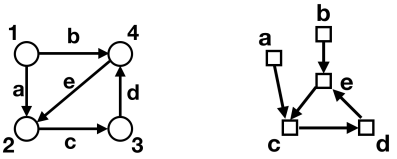
and $c (0 < c < 1)$ is a decay factor.

CoSimRank is similar to SimRank in terms of definition [14]. The experimental results demonstrated that CoSimRank is more accurate than both SimRank and personalized PageRank [14].

B. Second-order Random Walk

In first-order random walk, a node-to-node transition probability is used to determine where the surfer will go. In the second-order random walk, edge-to-edge transition probability is used [16]. It can be treated as the node-to-node transition in the line graph of the original graph.

Let $p_{i,j}$ be the probability of the surfer visiting node j from node i and r_j^t be the probability of the surfer visiting node j at time t. We have $r_j^t = \sum_{i \in I_j} p_{i,j} \cdot r_i^{t-1}$ where I_j is the



(a) original graph (b) the line graph

Figure 2. An example graph and its line graph

set of in-neighbors of j. When we deal with the second-order random walk, let $p_{i,j,k}$ be the probability that the surfer is currently at node j and on the way to node k, after previously visiting node i. Further assume that the probability of visiting j from i is u, and the probability of visiting k from j is v. The node to node probability $p_{i,j,k}$ can then be represented by edge-to-edge probability $p_{u,v}$. In Figure 2a, the probability of $p_{1,4,2}$ is the same as the probability $p_{b,e}$ in Figure 2b.

In the second-order random walk, the incidence matrix is used instead [16]. Incidence matrix is the node-to-edge matrix representing the relationship of nodes and edges. Let $\mathbf{B}$ and $\mathbf{E}$ be the out-edges and in-edges incidence matrices respectively. If element $[\mathbf{B}]_{i,u}$ is 1, the edge u is an out-edge of node i. If element $[\mathbf{E}]_{u,i}$ is 1, the edge u is an in-edge of node i. Otherwise the elements are 0. For example in Figure 2a, the incidence matrices are

$$\mathbf{B} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\!\!\begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}, \text{and} \quad \mathbf{E}^{\mathsf{T}} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\!\!\begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}.$$

Now let $\mathbf{H}$ be the node-to-edge transition probability matrix. $\mathbf{H}$ can also be represented by the equation $\mathbf{H} = \mathbf{D}^{-1}\mathbf{B}\mathbf{W}$, where $\mathbf{W}$ is the diagonal matrix with $[\mathbf{W}]_{u,u}$ being the weight of edge u. $\mathbf{P} = \mathbf{H}\mathbf{E}$ can then represent the node-to-node transition probability matrix.

III. Second-order CoSimRank

In this section, we present our second-order CoSimRank. The matrix form and convergence properties will be given, which is followed by a comparison of SimRank and CoSimRank.

A. Second-order CoSimRank

Suppose the random surfer starts from node i and randomly explore the network following the second-order random walk. Let $\mathbf{r}^{t,\,i}$ represent the probability distribution vector at time point t when the random surfer starts from node i. Since the random surfer starts from node i at time point t (t = 0), we have that

$$\mathbf{r}^{0,i} = \mathbf{q}^i \tag{4}$$

where

$$[\mathbf{q}^i]_k = \begin{cases} 1 & \text{if } k = i, \\ 0 & \text{if } k \neq i. \end{cases}$$

For the time point t (t = 1), we have that

$$\mathbf{r}^{t,i} = \mathbf{P}^{\mathsf{T}}\mathbf{q}^i \tag{5}$$

For the time point t ($t \geq 2$), we have that

$$\mathbf{r}^{t,i} = \mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{t-1}\mathbf{H}^{\mathsf{T}}\mathbf{q}^i \tag{6}$$

where $\mathbf{E}$ is in-edge incidence matrix. $\mathbf{M}$ is the edge-to-edge transition matrix. $\mathbf{H}$ is the node-to-edge transition matrix.

Thus, $\mathbf{r}^{t,\,i}$ for second-order CoSimRank can be represented by

$$\mathbf{r}^{t,i} = \begin{cases} \mathbf{q}^i & \text{if } t = 0, \\ \mathbf{P}^{\mathsf{T}}\mathbf{q}^i & \text{if } t = 1, \\ \mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{t-1}\mathbf{H}^{\mathsf{T}}\mathbf{q}^i & \text{if } t \geq 2. \end{cases}$$

CoSimRank can be defined using the probability distribution vectors $\mathbf{r}_{r,i}$. Specifically, the second-order CoSimRank values $\mathbf{s}_{(i,j)}$ between node i and j is defined as

$$\mathbf{s}_{(i,j)} = \sum_{t=0}^{\infty} c^t \cdot \left\langle \mathbf{r}^{t,i}, \mathbf{r}^{t,j} \right\rangle \tag{7}$$

where the operator $\langle \cdot, \cdot \rangle$ represents the inner product

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{k=1}^{n} x_k \cdot y_k \tag{8}$$

Let $\mathbf{R}^{(t)}$ represent the corresponding $\mathbf{R}$ in time point t. We know that $\mathbf{R}^{(0)} = \mathbf{I}$ where $\mathbf{I}$ is the identity matrix, and the series form can be written as

$$\mathbf{R} = \sum_{t=1}^{\infty} c^t \mathbf{H}\mathbf{M}^{t-1}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{t-1}\mathbf{H}^{\mathsf{T}} + \mathbf{I} \tag{9}$$

We can find:

$$\begin{aligned} \mathbf{R}^{(1)} &= c\mathbf{H}\mathbf{E}\mathbf{E}^{\mathsf{T}}\mathbf{H}^{\mathsf{T}} + \mathbf{R}^{(0)} \\ \mathbf{R}^{(2)} &= c^2\mathbf{H}\mathbf{M}\mathbf{E}\mathbf{E}^{\mathsf{T}}\mathbf{M}^{\mathsf{T}}\mathbf{H}^{\mathsf{T}} + \mathbf{R}^{(1)} \\ \mathbf{R}^{(3)} &= c^3\mathbf{H}\mathbf{M}^2\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^2\mathbf{H}^{\mathsf{T}} + \mathbf{R}^{(2)} \\ &\cdots \end{aligned} \tag{10}$$

Generally, for $t \geq 1$ we have

$$\mathbf{R}^{(t)} = c^t\mathbf{H}\mathbf{M}^{(t\text{-}1)}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{(t\text{-}1)}\mathbf{H}^{\mathsf{T}} + \mathbf{R}^{(t-1)} \tag{11}$$

B. Convergence Properties

We now show that the values of second-order CoSimRank exist and are unique.

Theorem 1. *There exists a unique solution to the second-order CoSimRank.*

Proof. We can see in Eq. (2) that the value of single pair CoSimRank is monotonically increasing as follows.

$$\sum_{t=0}^{\infty} c^t \cdot \left\langle \mathbf{r}^{t,i}, \mathbf{r}^{t,j} \right\rangle \leq \sum_{t=0}^{\infty} c^t$$

we consider the term $\left\langle \mathbf{r}^{t,i}, \mathbf{r}^{t,j} \right\rangle$. If $\left\langle \mathbf{r}^{t,i}, \mathbf{r}^{t,j} \right\rangle$ is 1 in every iteration, $\mathbf{s}_{(i,j)} = \sum_{t=0}^{\infty} c^t$. If $\left\langle \mathbf{r}^{t,i}, \mathbf{r}^{t,j} \right\rangle \leq 1$ in every iteration and $\mathbf{s}_{(i,j)} \leq \sum_{t=0}^{\infty} c^t$. By Cauchy-Schwarz inequality, the upper bound of the inner product of two vectors is

$$\langle u, v \rangle \leq ||u||\,||v||$$

In our case, $\mathbf{r}^{t,i}$ and $\mathbf{r}^{t,j}$ are two probability vectors, so $||\mathbf{r}^{t,i}||\,||\mathbf{r}^{t,j}|| = 1$ and $\left\langle \mathbf{r}^{t,i}, \mathbf{r}^{t,j} \right\rangle \leq 1$. Therefore,

$$\mathbf{s}_{(i,j)} \leq \sum_{t=0}^{\infty} c^t$$

As $\sum_{t=0}^{\infty} c^t = \frac{1}{1-c}$, we can express the above inequality as

$$\mathbf{s}_{(i,j)} \leq \frac{1}{1-c}$$

Together with the fact that $\mathbf{s}_{(i,j)}$ is monotonically increasing, it will finally converge.

As shown by the matrices in Eq. (9), a unique graph generates unique matrices $\mathbf{H}$, $\mathbf{M}$, $\mathbf{E}$. Given the uniqueness of the matrices, the value $\mathbf{R}$ is also unique. ■

C. Comparison of Second-order SimRank and Second-order CoSimRank

The first-order SimRank can be written as

$$\mathbf{R} = (c\mathbf{P}\mathbf{R}\mathbf{P}^{\mathsf{T}}) \vee \mathbf{I}$$

where $\mathbf{I}$ is the identity matrix, $\vee$ denotes the element-wise maximum, i.e., (i,j) entry of the matrix $\mathbf{A} \vee \mathbf{B}$ is given by $max\{\mathbf{A}_{i,j}, \mathbf{B}_{i,j}\}$, $\mathbf{P}$ is the transition matrix.

As the element-wise maximum is a non-linear operation, $\mathbf{R}$ can be rewritten as

$$\mathbf{R} = c\mathbf{P}\mathbf{R}\mathbf{P}^{\mathsf{T}} + \mathbf{D}$$

where $\mathbf{D}$ the diagonal correction matrix with $(1-c) \leq \mathbf{D}_{i,i} \leq 1$. The computation of $\mathbf{D}$ is also costly. So we usually use a simple approximation $\mathbf{D} \approx (1-c)\mathbf{I}$ [16] and we have

$$\mathbf{R} = c\mathbf{P}\mathbf{R}\mathbf{P}^{\mathsf{T}} + (1-c)\mathbf{I}$$

The series form can be represented by

$$\mathbf{R} = (1-c)\sum_{t=0}^{\infty} c^t \mathbf{P}^t (\mathbf{P}^{\mathsf{T}})^t \tag{12}$$

The series form of first-order CoSimRank can be written as

$$\mathbf{R} = \sum_{t=0}^{\infty} c^t \mathbf{P}^t (\mathbf{P}^{\mathsf{T}})^t \tag{13}$$

From the equations for SimRank and CoSimRank, we can find they differently initialize the similarities on the diagonal. SimRank sets each diagonal entries to one at each iteration while CoSimRank adds one.

In second-order, the recursive equation of second-order SimRank is as the following:

$$\begin{cases} \mathbf{S} = c\mathbf{M}\mathbf{S}\mathbf{M}^{\mathsf{T}} + (1-c)\mathbf{E}\mathbf{E}^{\mathsf{T}} \\ \mathbf{R} = c\mathbf{H}\mathbf{S}\mathbf{H}^{\mathsf{T}} + (1-c)\mathbf{I} \end{cases}$$

In series form, second-order SimRank can be represented by:

$$\mathbf{R} = (1-c)\sum_{t=1}^{\infty} c^t \mathbf{H}\mathbf{M}^{t-1}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{t-1}\mathbf{H}^{\mathsf{T}} + (1-c)\mathbf{I} \tag{14}$$

We have the element-wise form of second-order CoSimRank in Eq. (9). By comparing the Eq. in (9) and (14), we can observe that the two equations are similar since they are both in first-order.

IV. Computing Algorithm

In this section, we introduce the algorithm to calculate a single-source second-order CoSimRank score. Single-source second-order CoSimRank is querying from a node v_i requesting for the second-order CoSimRank score between v_i and every other node.

Given a recursive equation and the condition that the returned matrix or vector will converge, the power iteration can be used to find the final converged matrix or vector by running the equation over and over. Below are the stopping criterion for vector:

$$||\mathbf{r}^t - \mathbf{r}^{t-1}||_1 < \epsilon$$

and for matrix:

$$||\mathbf{A}^t - \mathbf{A}^{t-1}||_1 < \epsilon$$

where $||\mathbf{r}||_1 = \sum_i |r_i|$ and $||\mathbf{A}||_1 = \sum_{i,j} |A_{i,j}|$ denote the sum of absolute values, ϵ is a small positive value.

Power iteration is easy to implement when we know that the matrix or vector would finally converge. In our experiment, we use Eq. (11) to perform power iteration and use the value of $R^{(t)}$ to compute $R^{(t+1)}$. When the above criterion is met, we stop the iterations. However, the power iteration can take extended time before it terminates. Therefore, we develop another method to perform the computing experiments as shown in Algorithm 1.

Algorithm 1 Single-source algorithm for the second-order CoSimRank

Input : $G(V,E)$, query node $\mathbf{q}$, decay factor c, number of iterations β, transition matrices $\mathbf{M}$ and $\mathbf{H}$, incidence matrix $\mathbf{E}$

Output : proximity vector $\mathbf{r}$

```
r ← q
r' ← H^T q
for t ← 1 to β do
    r ← r + c^t H M^{t-1} E E^T r'
    r' ← M^T r'
return r
```

A. Single-source algorithm

The single-source values of second-order CoSimRank can be computed by the following equation:

$$\mathbf{r} = \sum_{t=1}^{\beta} c^t \mathbf{H}\mathbf{M}^{t-1}\mathbf{E}\mathbf{E}^{\mathsf{T}}(\mathbf{M}^{\mathsf{T}})^{t-1}\mathbf{H}^{\mathsf{T}}\mathbf{q} + \mathbf{q}$$

where β is the number of iteration, $\mathbf{q}$ is the query node. $\mathbf{r}$ is a vector containing second-order CoSimRank values of

query node $\mathbf{q}$ with any other nodes in the graph. As shown in Algorithm 1, it starts with initializing $\mathbf{r}$ with query node $\mathbf{q}$. The right hand side of the equation is calculated by maintaining $\mathbf{r}' = (\mathbf{M}^{\mathrm{T}})^{t-1}\mathbf{H}^{\mathrm{T}}\mathbf{q}$.

Time complexity: Time complexity of the single–source algorithm is $O(\beta\rho n)$. To calculate the values of all node in one iteration, it takes $O(\rho n)$. To iterate β times, it takes $O(\beta\rho n)$.

Space complexity: The space complexity for single–source algorithm is $O(\beta n^2)$. As mentioned earlier, single-pair value takes $O(\beta n)$ space. When we need to calculate all n nodes, it takes $O(\beta n^2)$.

B. Top-k nodes algorithm

The top-k nodes algorithm is based on the single-source algorithm. After Algorithm 1 being called, we get the CoSimRank value $\mathbf{r}$. Then, we sort the nodes and retrieve the top-k nodes in the list.

Time complexity: As we know, the time complexity of single-source algorithm is $O(\beta\rho n)$. After getting $\mathbf{r}$ from the algorithm, we sort the nodes. Sorting the nodes takes $O(n \log k)$. Overall, it takes $O(\beta\rho n + n \log k)$. However, $O(\beta\rho n)$ is much greater than $O(n \log k)$. So, to get top-k nodes, it takes $O(\beta\rho n)$.

Space complexity: As mentioned earlier, the space complexity of getting single-source is $O(\beta n^2)$. If we only get the top-k nodes, it will take $O(\beta n k)$ space.

V. Performance Evaluation

In this section, we present our experiments with the ground truth datasets and synthetic datasets. We use a normalized distributed cumulative cost gain (NDCG) statistical model to measure the accuracy of the CoSimRank algorithms. This necessitates ground truth data to be established. We use Facebook and Twitter API to get the real-world datasets. The datasets used for comparing the computing complexity of CoSimRank were created with a graph generator, GTGraph [21]. The model employed in the data analysis of this research is *Autoregression*.

A. Datasets

The experimental goals are to compare the accuracy and execution time of first-order and second-order CoSimRank with real-world datasets. For the Facebook data, we programmed a python project to access a random user and create an edge list of friends for the experimental dataset. Similarly, using the Python language and its libraries, we access a random Twitter user and create an edge list of followers for the Twitter experimental dataset.

One should note that in Facebook, the network is an undirected network because if user A is a friend of user B, it must be the same in the other way around. After we crawl the data on Facebook, we transform the directed Facebook dataset into undirected by copying the dataset and switch the two nodes in every edge. For example, if there is an edge (i, j) appearing in the original dataset, there should be another edge (j, i) appearing in the newly created dataset. In Twitter, the network is directed. If user A is a follower of user B, it is not necessary that user B is also a follower of A.

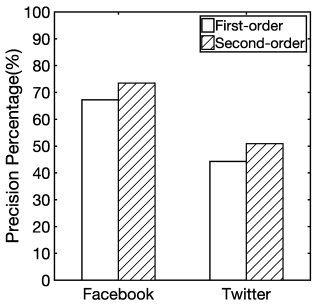


Figure 3. Link Prediction Precision for Facebook and Twitter data

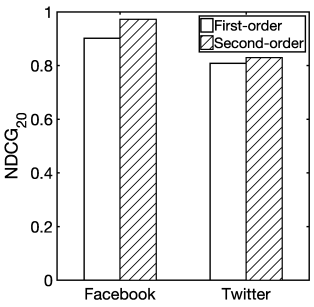


Figure 4. Facebook and Twitter Top-k Query Accuracy

B. Link Prediction

Facebook users and Twitter users are queried as the datasets for this experiment. A group of 46,008 users and 385,641 edges for Facebook, 12,150 users and 39,447 edges for Twitter are used. The data crawled before March 2018 is used for training, and the data crawled in September 2018 is used for testing. The autoregressive model is chosen to calculate the edge-to-edge transition matrix, where α is set at 0.5. To evaluate the accuracy, we define the precision as:

$$precision = \frac{k'}{k}$$

where k' is the number of correctly predicted edges that exactly appeared in the original dataset, k is the number of edges used as testing data. We randomly pick 10^3 query edges and generate the average. The training data is used to calculate the ranking values and the precision is shown in Figure 3.

In Figure 3, we can see that no matter in the Facebook or Twitter dataset, the second-order CoSimRank outperforms the first-order. The second-order CoSimRank improves the prevision by approximately 7% for Facebook data and 19% for Twitter data. This is due to the fact that the second-order CoSimRank considers the previously visited node, which can take advantages of the correlation among the visited nodes that are within two hops. The precision for Facebook data is higher than Twitter data, which shows a larger size of data can improve the precision value.

C. Top-k Query

In this experiment, a random user of Facebook and Twitter is selected, and the top twenty relevant friends are chosen. The technique used to rank importance is on a scale of one to five (five being the most relevant). We then observe how many friends that person has and whether that friend is another person or non-person (i.e., movie, company, product, etc.). A non-person decreases relevance ranking. From these observed friends, another list of twenty is created out of randomly selected friends. If a selected friend had no further friends, he is treated as a dangling node and his relevance is decreased.

Table II
SYNTHETIC DATA CHARACTERISTICS

Nodes $\times 2^{16}$	**Edges** $\times 10^5$
1	1
2	2
4	4
8	8

The computation of NDCG compared to the first and second-order CoSimRank are calculated. The discounted cumulative gain (DCG) measures the quality of the results in a ranked list, where items in that list are ranked by the scale of one to five. Typically, it is a relevance judgment for each randomly selected user or node. The DCG that is accumulated at a particular rank position p is defined as:

$$\mathrm{DCG}_p = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)} = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2(i+1)}$$

and the ideal DCG is defined as:

$$\mathrm{IDCG}_p = \sum_{i=1}^{|REL|} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

Normalized DCG (NDCG) is a way to calculate this measure across many independent queries, the result lists for which might all be of different length. NDCG is computed as:

$$\mathrm{NDCG}_p = \frac{DCG_p}{IDCG_p}$$

If more than one user is selected, the NDCG values are averaged to obtain an accuracy value, thus p is the number of chosen users or nodes. The results shown in Figure 4, indicate CoSimRank second-order is more accurate in ranking the top twenty friends of a friend on Facebook. It also indicates an improvement in ranking the top twenty Twitter followers. This is because the second-order measures can better capture the information among users and significantly improve the accuracy of the results.

D. Computational Runtime

The measure of computational complexity is taken using synthetic data in this experiment. The synthetic dataset is generated using GTGraph. Table II shows the generated dataset and its statistics. The runtime is collected for each process of CoSimRank with first and second-orders. Figure 5 shows the computational running time with respect to the number of nodes. We can find that with the increase in graph size, the computational time is increasing. For the second-order CoSimRank, it is slightly slower than the first-order CoSimRank. This is because we use the edge-to-edge information to calculate the value, and in a real-world application, the number of edges is usually larger than the number of nodes.

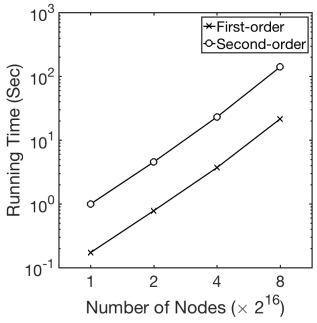


Figure 5. Computational Time Complexities

VI. CONCLUSION

Designing effective proximity measures for large graphs in the social network is a critical and computationally challenging task. With the recent introduction of CoSimRank, we can efficiently calculate the similarity values of two nodes. In this paper, we investigated the second-order CoSimRank measures which takes the previously visited node into consideration. We provided rigorous theoretical foundations for the second-order CoSimRank and developed second-order algorithms for second-order CoSimRank. The experimental results demonstrated that the second-order CoSimRank measures outperform the first-order CoSimRank in various applications. The simulation results of the random graph show that the time complexities for the second-order CoSimRank are slightly higher than the first-order CoSimRank.

REFERENCES

[1] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*. Cambridge university press, 1994, vol. 8.

[2] M. R. Bouadjenek, H. Hacid, and M. Bouzeghoub, "Sopra: A new social personalized ranking function for improving web search," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, 2013, pp. 861–864.

[3] P. Bedi and C. Sharma, "Community detection in social networks," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 6, no. 3, pp. 115–135, 2016.

[4] Y. Wu, R. Jin, J. Li, and X. Zhang, "Robust local community detection: On free rider effect and its elimination," *Proceedings of the VLDB Endowment*, vol. 8, no. 7, pp. 798–809, 2015.

[5] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A*, vol. 390, no. 6, pp. 1150–1170, 2011.

[6] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *JASIST*, vol. 58, no. 7, pp. 1019–1031, 2007.

[7] S. Cohen, B. Kimelfeld, and G. Koutrika, "A survey on proximity measures for social networks," in *Search computing*. Springer, 2012, pp. 191–206.

[8] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," 1999.

[9] H. Tong, C. Faloutsos, and J.-Y. Pan, "Fast random walk with restart and its applications," in *the 6th International Conference on Data Mining*, 2006, pp. 613–622.

[10] G. Jeh and J. Widom, "Simrank: a measure of structural-context similarity," in *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 538–543.

[11] L. Li, C. Li, H. Chen, and X. Du, "Mapreduce-based simrank computation and its application in social recommender system," in *Big Data (BigData Congress), 2013 IEEE International Congress on*, 2013, pp. 133–140.

[12] P. Li, H. Liu, J. X. Yu, J. He, and X. Du, "Fast single-pair simrank computation," in *Proceedings of the 2010 SIAM International Conference on Data Mining*, 2010, pp. 571–582.

[13] Y. Shao, B. Cui, L. Chen, M. Liu, and X. Xie, "An efficient similarity search framework for simrank over large dynamic graphs," *Proceedings of the VLDB Endowment*, vol. 8, no. 8, pp. 838–849, 2015.

[14] S. Rothe and H. Schütze, "Cosimrank: A flexible & efficient graph-theoretic similarity measure." in *ACL (1)*, 2014, pp. 1392–1402.

[15] W. Yu and J. A. McCann, "Co-simmate: Quick retrieving all pairwise co-simrank scores," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, vol. 2, 2015, pp. 327–333.

[16] Y. Wu, Y. Bian, and X. Zhang, "Remember where you came from: on the second-order random walk based proximity measures," *Proceedings of the VLDB Endowment*, vol. 10, no. 1, pp. 13–24, 2016.

[17] Y. Wu, X. Zhang, Y. Bian, Z. Cai, X. Lian, X. Liao, and F. Zhao, "Second-order random walk-based proximity measures in graph analysis: formulations and algorithms," *The VLDB Journal*, vol. 27, no. 1, pp. 127–152, 2018.

[18] W. Yu and F. Wang, "Fast exact cosimrank search on evolving and static graphs," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, 2018, pp. 599–608.

[19] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *Society for Industrial and Applied Mathematics review*, vol. 51, no. 3, pp. 455–500, 2009.

[20] A. N. Langville and C. D. Meyer, *Google's PageRank and beyond: The science of search engine rankings*. Princeton University Press, 2011, ch. The mathematics guide.

[21] K. Madduri and D. A. Bader. (2006) Gtgraph: A suite of synthetic random graph generators. [Online]. Available: http://www.cse.psu.edu/ kxm85/software/GTgraph/